Handbook of Philosophical Logic 17

Dov M. Gabbay
Franz Guenthner *Editors*

Handbook of Philosophical Logic

Second Edition

Springer

Handbook of Philosophical Logic

HANDBOOK OF PHILOSOPHICAL LOGIC

SECOND EDITION

Volume 17

edited by Dov M. Gabbay and Franz Guenthner

Volume 1 – ISBN 0-7923-7018-X
Volume 2 – ISBN 0-7923-7126-7
Volume 3 – ISBN 0-7923-7160-7
Volume 4 – ISBN 1-4020-0139-8
Volume 5 – ISBN 1-4020-0235-1
Volume 6 – ISBN 1-4020-0583-0
Volume 7 – ISBN 1-4020-0599-7
Volume 8 – ISBN 1-4020-0665-9
Volume 9 – ISBN 1-4020-0699-3
Volume 10 – ISBN 1-4020-1644-1
Volume 11 – ISBN 1-4020-1966-1
Volume 12 – ISBN 1-4020-3091-6
Volume 13 – ISBN 978-1-4020-3520-3
Volume 14 – ISBN 978-1-4020-6323-7
Volume 15 – ISBN 978-94-007-0484-8
Volume 16 – ISBN 978-94-007-0478-7

For further volumes:
http://www.springer.com/series/6024

Dov M. Gabbay • Franz Guenthner
Editors

Handbook of Philosophical Logic

Volume 17

Springer

Editors
Dov M. Gabbay
Department of Computer Science
King's College London
London
United Kingdom

Franz Guenthner
Centrüm für Informations- und Sprachvera
University of Munich
Munich
Germany

ISBN 978-94-007-6599-3 ISBN 978-94-007-6600-6 (eBook)
DOI 10.1007/978-94-007-6600-6
Springer Dordrecht Heidelberg New York London

Library of Congress Control Number: 2013939602

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface to the Second Edition

It is with great pleasure that we are presenting to the community the second edition of this extraordinary Handbook. It has been over 15 years since the publication of the first edition and there have been great changes in the landscape of philosophical logic since then.

The first edition has proved invaluable to generations of students and researchers in formal philosophy and language, as well as to consumers of logic in many applied areas. The main logic article in the *Encyclopaedia Britannica 1999* has described the first edition as 'the best starting point for exploring any of the topics in logic'. We are confident that the second edition will prove to be just as good!

The first edition was the second Handbook published for the logic community. It followed the North Holland one-volume *Handbook of Mathematical Logic*, published in 1977, edited by the late Jon Barwise. The four-volume *Handbook of Philosophical Logic*, published in 1983–1989, came at a fortunate temporal junction at the evolution of logic. This was the time when logic was gaining ground in computer science and artificial intelligence circles.

These areas were under increasing commercial pressure to provide devices which help and/or replace the human in his daily activity. This pressure required the use of logic in the modelling of human activity and organisation on the one hand and to provide the theoretical basis for the computer program constructs on the other. The result was that the *Handbook of Philosophical Logic*, which covered most of the areas needed from logic for these active communities, became their bible.

The increased demand for philosophical logic from computer science and artificial intelligence and computational linguistics accelerated the development of the subject directly and indirectly. It directly pushed research forward, stimulated by the needs of applications. New logic areas became established and old areas were enriched and expanded. At the same time, it socially provided employment for generations of logicians residing in computer science, linguistics and electrical engineering departments which of course helped keep the logic community thriving. In addition to that, it so happens (perhaps not by accident) that many of the Handbook contributors became active in these application areas and took their place

as time passed on, among the most famous leading figures of applied philosophical logic of our times. Today we have a Handbook with a most extraordinary collection of famous people as authors!

The table below will give our readers an idea of the landscape of logic and its relation to computer science and formal language and artificial intelligence. It shows that the first edition is very close to the mark of what was needed. Two topics were not included in the first edition, even though they were extensively discussed by all authors in a 3-day Handbook meeting. These are:

- A chapter on non-monotonic logic
- A chapter on combinatory logic and λ-calculus

We felt at the time (1979) that non-monotonic logic was not ready for a chapter yet and that combinatory logic and λ-calculus was too far removed.[1] Non-monotonic logic is now a very major area of philosophical logic, alongside default logics, labelled deductive systems, fibring logics, and multi-dimensional, multimodal and substructural logics. Intensive re-examinations of fragments of classical logic have produced fresh insights, including at times decision procedures and equivalence with non-classical systems.

Perhaps the most impressive achievement of philosophical logic as arising in the past decade has been the effective negotiation of research partnerships with fallacy theory, informal logic and argumentation theory, attested to by the Amsterdam Conference in Logic and Argumentation in 1995, and the two Bonn Conferences in Practical Reasoning in 1996 and 1997.

These subjects are becoming more and more useful in agent theory and intelligent and reactive databases.

Finally, 15 years after the start of the Handbook project, I would like to take this opportunity to put forward my current views about logic in computer science, computational linguistics and artificial intelligence. In the early 1980s, the perception of the role of logic in computer science was that of a specification and reasoning tool and that of a basis for possibly neat computer languages. The computer scientist was manipulating data structures and the use of logic was one of his options.

My own view at the time was that there was an opportunity for logic to play a key role in computer science and to exchange benefits with this rich and important application area and thus enhance its own evolution. The relationship between logic and computer science was perceived as very much like the relationship of applied mathematics to physics and engineering. Applied mathematics evolves through its use as an essential tool, and so we hoped for logic. Today my view has changed. As computer science and artificial intelligence deal more and more

[1] I am really sorry, in hindsight, about the omission of the non-monotonic logic chapter. I wonder how the subject would have developed, if the AI research community had had a theoretical model, in the form of a chapter, to look at. Perhaps the area would have developed in a more streamlined way!

with distributed and interactive systems, processes, concurrency, agents, causes, transitions, communication and control (to name a few), the researcher in this area is having more and more in common with the traditional philosopher who has been analysing such questions for centuries (unrestricted by the capabilities of any hardware).

The principles governing the interaction of several processes, for example, are abstract and similar to principles governing the cooperation of two large organisations. A detailed rule based effective but rigid bureaucracy is very much similar to a complex computer program handling and manipulating data. My guess is that the principles underlying one are very much the same as those underlying the other.

I believe the day is not far away in the future when the computer scientist will wake up one morning with the realisation that he is actually a kind of formal philosopher!

The projected number of volumes for this Handbook is about 18. The subject has evolved and its areas have become interrelated to such an extent that it no longer makes sense to dedicate volumes to topics. However, the volumes do follow some natural groupings of chapters.

I would like to thank our authors and readers for their contributions and their commitment in making this Handbook a success. Thanks also to our publication administrator Mrs J. Spurr for her usual dedication and excellence and to Kluwer Academic Publishers for their continuing support for the Handbook.

King's College London, and
Bar Ilan University, Israel, and
University of Luxembourg

Dov M. Gabbay

Logic	**IT**			
	Natural language processing	**Program control specification, verification, concurrency**	**Artificial intelligence**	**Logic programming**
Temporal logic	Expressive power of tense operators. Temporal indices. Separation of past from future	Expressive power for recurrent events. Specification of temporal control. Decision problems. Model checking	Planning. Time dependent data. Event calculus. Persistence through time—the Frame Problem. Temporal query language. Temporal transactions	Extension of Horn clause with time capability. Event calculus. Temporal logic programming
Modal logic. Multi-modal logics	Generalised quantifiers	Action logic	Belief revision. Inferential databases	Negation by failure and modality
Algorithmic proof	Discourse representation. Direct computation on linguistic input	New logics. Generic theorem provers	General theory of reasoning. Non-monotonic systems	Procedural approach to logic
Non-monotonic reasoning	Resolving ambiguities. Machine translation. Document classification. Relevance theory	Loop checking. Non-monotonic decisions about loops. Faults in systems	Intrinsic logical discipline for AI. Evolving and communi-cating databases	Negation by failure. Deductive databases
Probabilistic and fuzzy logic	Logical analysis of language	Real time systems	Expert systems. Machine learning	Semantics for logic programs
Intuitionistic logic	Quantifiers in logic	Constructive reasoning and proof theory about specification design	Intuitionistic logic is a better logical basis than classical logic	Horn clause logic is really intu-itionistic. Extension of logic pro-gramming languages
Set theory, higher-order logic, λ-calculus, types	Montague semantics. Situation semantics	Non-well-founded sets	Hereditary finite predicates	λ-calculus extension to logic programs

(continued)

(continued)

Imperative vs. declarative languages	**Database theory**	**Complexity theory**	**Agent theory**	**Special comments: A look to the future**
Temporal logic as a declarative programming language. The changing past in databases. The imperative future	Temporal databases and temporal transactions	Complexity questions of decision procedures of the logics involved	An essential component	Temporal systems are becoming more and more sophisticated and extensively applied
Dynamic logic	Database updates and action logic	Ditto	Possible actions	Multimodal logics are on the rise. Quantification and context becoming very active
Types. Term rewrite systems. Abstract interpretation	Abduction, relevance	Ditto	Agent's implementation rely on proof theory	
	Inferential databases. Non-monotonic coding of databases	Ditto	Agent's reasoning is non-monotonic	A major area now. Important for formalising practical reasoning
	Fuzzy and probabilistic data	Ditto	Connection with decision theory	Major area now
Semantics for programming languages. Martin-Löf theories	Database transactions. Inductive learning	Ditto	Agent's constructive reasoning	Still a major central alternative to classical logic
Semantics for programming languages. Abstract interpretation. Domain recursion theory.		Ditto		More central than ever!

(continued)

(continued)

Classical logic. Classical fragments	Basic background language	Program synthesis	A basic tool	
Labelled deductive systems	Extremely useful in modelling		A unifying framework. Context theory	Annotated logic programs
Resource and substructural logics	Lambek calculus		Truth maintenance systems	
Fibring and combining logics	Dynamic syntax	Modules. Combining languages	Logics of space and time	Combining features
Fallacy theory				
Logical dynamics	Widely applied here			
Argumentation theory games		Game semantics gaining ground		
Object level/ metalevel **Mechanisms: Abduction, default relevance**			Extensively used in AI Ditto	
Connection with neural nets				
Time-action-revision models			Ditto	
	Relational databases	Logical complexity classes	The workhorse of logic	The study of fragments is very active and promising.
	Labelling allows for context and control		Essential tool	The new unifying framework for logics
Linear logic			Agents have limited resources	

(continued)

(continued)

	Linked databases. Reactive databases		Agents are built up of various fibred mecha-nisms	The notion of self-fibring allows for self-reference
				Fallacies are really valid modes of reasoning in the right context
			Potentially applicable	A dynamic view of logic
				On the rise in all areas of applied logic. Promises a great future
			Important feature of agents	Always central in all areas
			Very important for agents	Becoming part of the notion of a logic
				Of great im-portance to the future. Just starting
			A new theory of logical agent	A new kind of model

Contents

Chapter 1
Hybrid Logic

Torben Braüner

The starting point of this chapter[1] is the remarkable fact that proof procedures for wide classes of hybrid logics can be given in a uniform way, and moreover, this encompasses proof procedures like natural deduction and tableau systems which are suitable for actual[2] reasoning. A focus of the chapter is such proof procedures. Axiom systems, which are not meant for actual reasoning, are only mentioned in passing. We present a relatively small selection of procedures rather than trying to be encyclopedic. This allows us to give a reasonably detailed treatment of the selected procedures. Another focus of the chapter is the origin of hybrid logic in Arthur Prior's philosophical work.

In the first section of the chapter, Sect. 1.1, we give the basics of hybrid logic. In Sect. 1.2 we discuss the work of Arthur Prior and describe how hybrid logic has its origin in his work. In Sect. 1.3 we outline the development of hybrid logic since Prior. In Sect. 1.4 we introduce a natural deduction system for hybrid logic and in Sect. 1.5 we introduce tableau systems and tableau-based decision procedures for hybrid logic. In Sect. 1.6 we try to give an answer to the following question: Why does the proof-theory of hybrid logic behave so well compared to the proof-theory of ordinary modal logic?

[1]The chapter is composed of material adapted from the author's book (Braüner 2011). The author wishes to acknowledge the financial support received from The Danish Natural Science Research Council as funding for the projects HyLoMOL (2004–2008) and HYLOCORE (2009–2013).

[2]The word "actual" has here a broad meaning, not restricted to actual human reasoning. The logic does not care whether it is a human that carries out the reasoning, or the reasoning takes place in a computer, or in some other medium.

T. Braüner (✉)
Programming, Logic and Intelligent Systems Research Group, Roskilde University,
DK-4000 Roskilde, Denmark
e-mail: torben@ruc.dk

D.M. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic: Volume 17*,
Handbook of Philosophical Logic 17, DOI 10.1007/978-94-007-6600-6_1,

1.1 The Basics of Hybrid Logic

In this section we give the basics of hybrid logic. We first give an informal motivation of hybrid logic. We then give the formal syntax and semantics and we give translations forwards and backwards between hybrid logic and first-order logic.

1.1.1 Informal Motivation

The term "hybrid logic" covers a number of logics obtained by adding further expressive power to ordinary modal logic.[3] The history of what now is known as hybrid logic goes back to Arthur Prior's work in the 1960s, which we shall come back to in Sect. 1.2. The term "hybrid logic" was coined in Patrick Blackburn and Jerry Seligman's paper published in 1995. The most basic hybrid logic is obtained by adding nominals, which are propositional symbols of a new sort interpreted in a restricted way that enables reference to individual points in a Kripke model. In what follows we shall give a more detailed explanation.

In the standard Kripke semantics for modal logic, the truth-value of a formula is relative to points in a set, that is, a formula is evaluated "locally" at a point. Usually, the points are taken to represent possible worlds, times, locations, epistemic states, states in a computer, or something else. Thus, in the Kripke semantics, a propositional symbol might have different truth-values at different points. This allows us to formalize natural language statements whose truth-values are relative to, for example times, like the statement

> it is raining

which has clearly different truth-values at different times. Such statements can be formalized in ordinary modal logic using ordinary propositional symbols. Now, certain natural language statements are true at exactly one time, possible world, or something else. An example is the statement

> it is 5 o'clock 10 May 2007

which is true at the time 5 o'clock 10 May 2007, but false at all other times. While the first kind of statement can be formalized in ordinary modal logic, the second kind of statement cannot, the reason being that there is only one sort of propositional symbol available, namely ordinary propositional symbols, which are not restricted to being true at exactly one point in the Kripke semantics.

A major motivation for hybrid logic is to add further expressive power to ordinary modal logic with the aim of being able to formalize the second kind of statement. This is obtained by adding to ordinary modal logic a second sort of propositional

[3]This should not be confused with the term "hybrid systems" which in the computer science community is used for systems that combine discrete and continuous features.

symbol called a nominal such that in the Kripke semantics each nominal is true at exactly one point. In other words, a nominal is interpreted with the restriction that the set of points at which it is true is a singleton set, not an arbitrary set. A natural language statement of the second kind (like the example statement with the time 5 o'clock 10 May 2007) is then formalized using a nominal, not an ordinary propositional symbol (which is used to formalize the example statement with rainy weather). The fact that a nominal is true at exactly one point implies that a nominal can be considered a term referring to a point, for example, if *a* is a nominal that stands for "it is 5 o'clock 10 May 2007", then the nominal *a* can be considered a term referring to the time 5 o'clock 10 May 2007.[4] Thus, in hybrid logic a term is a specific sort of propositional symbol whereas in first-order logic it is an argument to a predicate.

Most hybrid logics involve further additional machinery than nominals. There is a number of options for adding further machinery; here we shall consider a kind of operator called satisfaction operators. The motivation for adding satisfaction operators is to be able to formalize a statement being true at a particular time, possible world, or something else. For example, we want to be able to formalize that the statement "it is raining" is true at the time 5 o'clock 10 May 2007, that is, that

> at 5 o'clock 10 May 2007, it is raining.

[4]Considering a nominal as a symbol that refers to something is not the only way to view nominals. Two different views on nominals can be identified in the works of Arthur Prior, as is clear from the quotation below where Prior discusses the addition of nominals to a temporal version of modal logic called tense logic.

> We might . . . equate the instant *a* with a conjunction of all those propositions which would ordinarily be said to be true at that instant, or we might equate it with some proposition which would ordinarily be said to be true at that instant only, and so could serve as an index of it (Hasle et al. 2003, p. 124).

In the second half of the sentence, the nominal *a* is viewed as a proposition that can serve as an index of an instant, which is clearly in line with considering a nominal as a symbol that refers to an instant. On the other hand, in the first half of the sentence, the nominal *a* is viewed as a description of the content of an instant. The alternative view on nominals expressed in the first half of the sentence quoted above can also be found in a number of other places in Prior's works, for example the following.

> The essential trick is to treat the instant variables as a special sort of *propositional* variables, by identifying an 'instant' with the totality of what would ordinarily be said to be true *at* that instant, . . . (Hasle et al. 2003, p. 141).

See the discussion of Prior's work in Sect. 1.2 of the present handbook chapter, in particular Footnote 8 of that section. Moreover, see the discussion in Patrick Blackburn's paper (2006), the last paragraph of page 353, including Footnote 7, and the first complete paragraph of page 362, in particular Footnote 11. Incidentally, note that the description of the content of an instant as the conjunction of all propositions true at that instant is similar to a maximal consistent set of formulas.

This is formalized by the formula $@_a p$ where the nominal a stands for "it is 5 o'clock 10 May 2007" as above and where p is an ordinary propositional symbol that stands for "it is raining". It is the part $@_a$ of the formula $@_a p$ that is called a satisfaction operator. In general, if a is a nominal and ϕ is an arbitrary formula, then a new formula $@_a \phi$ can be built (in some literature the notation $a : \phi$ is used instead of $@_a \phi$). A formula of the form $@_a \phi$ is called a satisfaction statement. The satisfaction statement $@_a \phi$ expresses that the formula ϕ is true at one particular point, namely the point to which the nominal a refers.

To sum up, we have now added further expressive power to ordinary modal logic in the form of nominals and satisfaction operators. Informally, the nominal a has the truth-condition

a is true relative to a point w
if and only if
the reference of a is identical to w

and the satisfaction statement $@_a \phi$ has the truth-condition

$@_a \phi$ is true relative to a point w
if and only if
ϕ is true relative to the reference of a

Observe that actually the point w does not matter in the truth-condition for $@_a \phi$ since the satisfaction operator $@_a$ moves the point of evaluation to the reference of a whatever the identity of w. Note that the addition of nominals and satisfaction operators does not disturb the local character of the Kripke semantics: The truth-value of a formula is still relative to points in a set and the added machinery only involves reference to particular points, not all points in the set.

It is worth noting that nominals together with satisfaction operators allow us to express that two points are identical: If the nominals a and b refer to the points u and v, then the formula $@_a b$ expresses that u and v are identical. The following line of reasoning shows why.

$@_a b$ is true relative to a point w
if and only if
b is true relative to the reference of a
if and only if
b is true relative to u
if and only if
the reference of b is identical to u
if and only if
v is identical to u

The identity relation on a set has the well-known properties reflexivity, symmetry, and transitivity, which is reflected in the fact that the formulas

$@_a a$
$@_a b \rightarrow @_b a$
$(@_a b \wedge @_b c) \rightarrow @_a c$

are valid formulas of hybrid logic. To see that these hybrid-logical formulas correspond to the properties reflexivity, symmetry, and transitivity, read $@_a b$ as $a = b$ etc. Also the formula

$$(@_a b \wedge @_a \phi) \rightarrow @_b \phi$$

is valid. This hybrid-logical formula corresponds to the rule of replacement.

Beside nominals and satisfaction operators, in what follows we shall consider the binders $\forall$ and $\downarrow$, which allow us to build formulas $\forall a\phi$ and $\downarrow a\phi$. The binders bind nominals to points in two different ways: The $\forall$ binder quantifies over all points analogous to the standard first-order universal quantifier, that is, $\forall a\phi$ is true relative to w if and only if whatever point the nominal a refers to, ϕ is true relative to w. The $\downarrow$ binder binds a nominal to the point of evaluation, that is, $\downarrow a\phi$ is true relative to w if and only if ϕ is true relative to w when a refers to w. It turns out that the $\downarrow$ binder is definable in terms of $\forall$.

Above we noted that nominals and satisfaction operators do not disturb the local character of the Kripke semantics. Also the $\downarrow$ binder leaves the local character of the semantics undisturbed since this binder just binds a nominal to the point of evaluation. Things are more complicated with the $\forall$ binder. This binder has a non-local character in the sense that it involves reference to all points in the Kripke semantics. Moreover, together with nominals and satisfaction operators, the $\forall$ binder gives rise to non-local expressivity in the form of full first-order expressive power (which we shall show in Sect. 1.1.3). However, the $\forall$ binder does not give rise to full first-order expressive power just together with nominals, that is, in the absence of satisfaction operators (or some similar machinery). Thus, it is really the interaction between the $\forall$ binder and satisfaction operators that gives rise to full first-order expressive power, and hence, non-local expressivity.[5]

To conclude, extending ordinary modal logic with hybrid-logical machinery (disregarding the extreme case involving both $\forall$ and satisfaction operators), gives us a more expressive logic without sacrificing the local character of the Kripke semantics.[6]

[5] In fact, the paper (Blackburn and Seligman 1995) gives a result (Proposition 4.5 on p. 264) indicating that the $\forall$ binder has a surprisingly local character when it is not accompanied by satisfaction operators or some similar machinery. Informally, this result says that the $\forall$ binder is then insensitive to the information at points outside the submodel generated by the point of evaluation, that is, it cannot detect the truth-values of formulas at such points.

[6] Further discussion of this point can be found in a number of places, notably the paper Blackburn (2006). This paper also discusses hybrid-logical versions of *bisimulations*, which give a mathematical way to illustrate the local character of the Kripke semantics. See also the paper Simons (2006) which discusses a number of logics of location involving what we here call satisfaction operators.

1.1.2 Formal Syntax and Semantics

In what follows we give the formal syntax and semantics of hybrid logic. In many cases we will adopt the terminology of Blackburn et al. (2001) and Areces et al. (2001a). The hybrid logic we consider is obtained by adding a second sort of propositional symbol, called *nominals*, to ordinary modal logic, that is, propositional logic extended with a modal operator $\Box$.[7] It is assumed that a set of ordinary propositional symbols and a countably infinite set of nominals are given. The sets are assumed to be disjoint. The metavariables p, q, r, ... range over ordinary propositional symbols and a, b, c, ... range over nominals. Besides nominals, an operator $@_a$ called a *satisfaction operator* is added for each nominal a. Sometimes the operator $@_a$ is called an *at operator*. Moreover, we shall consider the *binders* $\forall$ and $\downarrow$. The formulas of hybrid modal logic are defined by the grammar

$$S ::= p \mid a \mid S \wedge S \mid S \rightarrow S \mid \bot \mid \Box S \mid @_a S \mid \forall a S \mid \downarrow a S$$

where p ranges over ordinary propositional symbols and a ranges over nominals. In what follows, the metavariables ϕ, ψ, θ, ... range over formulas. Formulas of the form $@_a\phi$ are called *satisfaction statements* (cf. Blackburn 2000a). The notions of free and bound occurrences of nominals are defined as in first-order logic with the addition that the free nominal occurrences in $@_a\phi$ are the free nominal occurrences in ϕ together with the occurrence of a, and moreover, the free nominal occurrences in $\downarrow a\phi$ are the free nominal occurrences in ϕ except for occurrences of a. Also, if $\overline{a}$ is a list of pairwise distinct nominals and $\overline{c}$ is a list of nominals of the same length as $\overline{a}$, then $\psi[\overline{c}/\overline{a}]$ is the formula ψ where the nominals $\overline{c}$ have been simultaneously substituted for all free occurrences of the nominals $\overline{a}$. If a nominal a_i in $\overline{a}$ occurs free in ψ within the scope of $\forall c_i$ or $\downarrow c_i$, then the nominal c_i in ψ is renamed as appropriate (this can be done since there are infinitely many nominals). The connectives negation, nullary conjunction, disjunction, and bi-implication are defined by the conventions that $\neg\phi$ is an abbreviation for $\phi \rightarrow \bot$, $\top$ is an abbreviation for $\neg\bot$, $\phi \vee \psi$ is an abbreviation for $\neg(\neg\phi \wedge \neg\psi)$, and $\phi \leftrightarrow \psi$ is an abbreviation for $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$. Similarly, $\Diamond\phi$ is an abbreviation for $\neg\Box\neg\phi$ and $\exists a\phi$ is an abbreviation for $\neg\forall a\neg\phi$.

We now define models and frames.

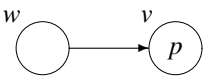
Definition 1.1. A *model* for hybrid logic is a tuple $(W, R, \{V_w\}_{w \in W})$ where

1. W is a non-empty set;
2. R is a binary relation on W; and
3. for each w, V_w is a function that to each ordinary propositional symbol assigns an element of $\{0, 1\}$.

[7] All results in the present handbook chapter can be generalized to cover an arbitrary, finite number of modal operators, but in the interest of simplicity, we shall stick to one modal operator unless otherwise is specified.

The pair (W,R) is called a *frame* and the model is said to be *based* on this frame. The elements of W are called *worlds* and the relation R is called the *accessibility relation*. A propositional symbol p is said to be *true* at w if $V_w(p) = 1$ and it is said to be *false* at w if $V_w(p) = 0$.

Note that a model for hybrid logic is the same as a model for ordinary modal logic. To give an extremely simple example of a model, we let $W = \{w,v\}$ and $R = \{(w,v)\}$, and moreover, we let $V_w(p) = 0$ and $V_v(p) = 1$. All other propositional symbols than p are ignored. This model can be depicted as



where circles represent worlds and an arrow indicates that two worlds are related by the accessibility relation. A propositional symbol in a circle means that the symbol is true and the absence of a propositional symbol means that it is false.

Given a model $\mathfrak{M} = (W,R,\{V_w\}_{w\in W})$, an *assignment* is a function g that to each nominal assigns an element of W. Given assignments g' and g, $g' \overset{a}{\sim} g$ means that g' agrees with g on all nominals save possibly a. The relation $\mathfrak{M},g,w \models \phi$ is defined by induction, where g is an assignment, w is an element of W, and ϕ is a formula.

$$
\begin{array}{rll}
\mathfrak{M},g,w \models p & \text{iff} & V_w(p) = 1 \\
\mathfrak{M},g,w \models a & \text{iff} & w = g(a) \\
\mathfrak{M},g,w \models \phi \wedge \psi & \text{iff} & \mathfrak{M},g,w \models \phi \text{ and } \mathfrak{M},g,w \models \psi \\
\mathfrak{M},g,w \models \phi \rightarrow \psi & \text{iff} & \mathfrak{M},g,w \models \phi \text{ implies } \mathfrak{M},g,w \models \psi \\
\mathfrak{M},g,w \models \bot & \text{iff} & \text{falsum} \\
\mathfrak{M},g,w \models \Box\phi & \text{iff} & \text{for any } v \in W \text{ such that } wRv,\ \mathfrak{M},g,v \models \phi \\
\mathfrak{M},g,w \models @_a\phi & \text{iff} & \mathfrak{M},g,g(a) \models \phi \\
\mathfrak{M},g,w \models \forall a\phi & \text{iff} & \text{for any } g' \overset{a}{\sim} g,\ \mathfrak{M},g',w \models \phi \\
\mathfrak{M},g,w \models \downarrow a\phi & \text{iff} & \mathfrak{M},g',w \models \phi \text{ where } g' \overset{a}{\sim} g \text{ and } g'(a) = w
\end{array}
$$

A formula ϕ is said to be *true* at w if $\mathfrak{M},g,w \models \phi$; otherwise it is said to be *false* at w. By convention $\mathfrak{M},g \models \phi$ means $\mathfrak{M},g,w \models \phi$ for every element w of W and $\mathfrak{M} \models \phi$ means $\mathfrak{M},g \models \phi$ for every assignment g. A formula ϕ is *valid* in a frame if and only if $\mathfrak{M} \models \phi$ for any model $\mathfrak{M}$ that is based on the frame. A formula ϕ is *valid* in a class of frames if and only if ϕ is valid in any frame in the class of frames in question. A formula ϕ is *valid* if and only if ϕ is valid in the class of all frames.

Now, let $\mathcal{O} \subseteq \{\downarrow,\forall\}$. In what follows $\mathcal{H}(\mathcal{O})$ denotes the fragment of hybrid logic in which the only binders are the binders in the set $\mathcal{O}$. If $\mathcal{O} = \emptyset$, then we simply write $\mathcal{H}$, and if $\mathcal{O} = \{\downarrow\}$, then we write $\mathcal{H}(\downarrow)$, etc. It is assumed that the set $\mathcal{O}$ of binders is fixed.

Note that $\downarrow$ is definable in terms of $\forall$ since the formula $\downarrow a\phi \leftrightarrow \forall a(a \rightarrow \phi)$ is valid. The fact that hybridizing ordinary modal logic actually does give more expressive power can for example be seen by considering the formula $\downarrow c\Box\neg c$. It is

straightforward to check that this formula is valid in a frame if and only if the frame is irreflexive. Thus, irreflexivity can be expressed by a hybrid-logical formula, but it is well known that it cannot be expressed by any formula of ordinary modal logic. Irreflexivity can actually be expressed just by adding nominals to ordinary modal logic, namely by the formula $c \to \Box\neg c$. It is clear that if a frame is irreflexive, then $c \to \Box\neg c$ is valid in the frame. On the other hand, if $c \to \Box\neg c$ is valid in a frame, then the frame is irreflexive: Let (W,R) be a frame in which $c \to \Box\neg c$ is valid and let w be an element of W, then $\mathfrak{M},g,w \models c \to \Box\neg c$ where $\mathfrak{M}$ is an arbitrarily chosen model based on (W,R) and g is an arbitrarily chosen assignment such that $g(c) = w$, and from this it follows that wRw is false. Hence, the formula $c \to \Box\neg c$ expresses irreflexivity. Other examples of properties expressible in hybrid logic, but not in ordinary modal logic, are asymmetry (expressed by $c \to \Box\neg\Diamond c$), antisymmetry (expressed by $c \to \Box(\Diamond c \to c)$), and universality (expressed by $\Diamond c$).

1.1.3 Translation into First-Order Logic

Hybrid logic can be translated into first-order logic with equality and (a fragment of) first-order logic with equality can be translated back into (a fragment of) hybrid logic. The translation from hybrid logic into first-order logic we consider in this subsection is an extension of the well-known *standard translation* from modal logic into first-order logic (see Areces et al. 2001a and van Benthem 1983).

The first-order language under consideration has a 1-place predicate symbol corresponding to each ordinary propositional symbol of modal logic, a 2-place predicate symbol corresponding to the modality, and a 2-place predicate symbol corresponding to equality. The language does not have constant or function symbols. It is assumed that a countably infinite set of first-order variables is given. The metavariables a, b, c, ... range over first-order variables. There are no function symbols or constants. So the formulas of the first-order language we consider are defined by the grammar

$$S ::= p^*(a) \mid R(a,b) \mid a = b \mid S \wedge S \mid S \to S \mid \bot \mid \forall a S$$

where p ranges over ordinary propositional symbols of hybrid logic, and a and b range over first-order variables. Note that according to the grammar above, for each ordinary propositional symbol p of the modal language there is a corresponding 1-place predicate symbol p^* in the first-order language. The predicate symbol p^* will be interpreted such that it relativises the interpretation of the corresponding modal propositional symbol p to worlds. In the grammar above, R is a designated predicate symbol which will be interpreted using the accessibility relation (with the same name). In what follows, we shall identify first-order variables with nominals of hybrid logic. Note in this connection that the set of metavariables ranging over first-order variables is identical to the set of metavariables ranging over nominals. Free and bound occurrences of variables are defined as usual for first-order logic.

Also, $\psi[c/a]$ is the formula ψ where the variable c has been substituted for all free occurrences of the variable a. As usual, if the variable a occurs free in ψ within the scope of $\forall c$, then the variable c in ψ is renamed as appropriate. It is assumed that a does not occur free in ψ within the scope of $\forall c$. The connectives $\neg$, $\top$, $\vee$, $\leftrightarrow$, and $\exists$ are defined in one of the usual ways.

We first translate the hybrid logic $\mathcal{H}(\downarrow,\forall)$ into first-order logic with equality. It is assumed that two nominals a and b are given which do not occur in the formulas to be translated. The translations ST_a and ST_b are defined by mutual induction. We just give the translation ST_a.

$$\begin{aligned}
ST_a(p) &= p^*(a)\\
ST_a(c) &= a = c\\
ST_a(\phi \wedge \psi) &= ST_a(\phi) \wedge ST_a(\psi)\\
ST_a(\phi \rightarrow \psi) &= ST_a(\phi) \rightarrow ST_a(\psi)\\
ST_a(\bot) &= \bot\\
ST_a(\Box\phi) &= \forall b(R(a,b) \rightarrow ST_b(\phi))\\
ST_a(@_c\phi) &= ST_a(\phi)[c/a]\\
ST_a(\forall c\phi) &= \forall c ST_a(\phi)\\
ST_a(\downarrow c\phi) &= ST_a(\phi)[a/c]
\end{aligned}$$

The definition of ST_b is obtained by exchanging a and b. As an example, we demonstrate step by step how the hybrid-logical formula $\downarrow c\Box\neg c$ is translated into a first-order formula:

$$\begin{aligned}
ST_a(\downarrow c\Box\neg c) &= ST_a(\Box\neg c)[a/c]\\
&= \forall b(R(a,b) \rightarrow ST_b(\neg c))[a/c]\\
&= \forall b(R(a,b) \rightarrow \neg ST_b(c))[a/c]\\
&= \forall b(R(a,b) \rightarrow \neg b = c)[a/c]\\
&= \forall b(R(a,b) \rightarrow \neg b = a)
\end{aligned}$$

The resulting first-order formula is equivalent to $\neg R(a,a)$ which shows that $\downarrow c\Box\neg c$ indeed does correspond to the accessibility relation being irreflexive, cf. above. What has been done in the translation is that the semantics of hybrid logic has been formalised in terms of first-order logic; note how each clause in the translation formalizes a clause in the definition of the semantics, that is, the relation $\mathfrak{M}, g, w \models \phi$.

The translation ST_a is truth-preserving. To state this formally, we make use of the well-known observation that a model for hybrid logic can be considered as a model for first-order logic and vice versa.

Definition 1.2. Given a model $\mathfrak{M} = (W, R, \{V_w\}_{w\in W})$ for hybrid logic, a model $\mathfrak{M}^* = (W, V^*)$ for first-order logic is defined by letting

- $V^*(p^*) = \{w \mid V_w(p) = 1\}$ and
- $V^*(R) = R$.

It is straightforward to see that the map $(\cdot)^*$ which maps $\mathfrak{M}$ to $\mathfrak{M}^*$ is bijective. Moreover, an assignment in the sense of classical hybrid logic can be considered as an assignment in the sense of classical first-order logic and vice versa.

Given a model $\mathfrak{M}$ for first-order logic, the relation $\mathfrak{M}, g \models \phi$ is defined by induction in the standard way, where g is an assignment and ϕ is a first-order formula.

$$\begin{array}{rll}
\mathfrak{M}, g \models p^*(a) & \text{iff} & g(a) \in V(p^*) \\
\mathfrak{M}, g \models R(a,b) & \text{iff} & g(a) V(R) g(b) \\
\mathfrak{M}, g \models a = b & \text{iff} & g(a) = g(b) \\
\mathfrak{M}, g \models \phi \wedge \psi & \text{iff} & \mathfrak{M}, g \models \phi \text{ and } \mathfrak{M}, g \models \psi \\
\mathfrak{M}, g \models \phi \to \psi & \text{iff} & \mathfrak{M}, g \models \phi \text{ implies } \mathfrak{M}, g \models \psi \\
\mathfrak{M}, g \models \bot & \text{iff} & \text{falsum} \\
\mathfrak{M}, g \models \forall a \phi & \text{iff} & \text{for any } g' \overset{a}{\sim} g,\ \mathfrak{M}, g' \models \phi
\end{array}$$

The formula ϕ is said to be *true* if $\mathfrak{M}, g \models \phi$; otherwise it is said to be *false*. By convention $\mathfrak{M} \models \phi$ means $\mathfrak{M}, g \models \phi$ for every assignment g. We shall later make use of the first-order semantics in connection with the interpretation of geometric theories.

It can now be stated formally that the translation is truth-preserving.

Proposition 1.3. *Let $\mathfrak{M}$ be a model for hybrid logic and let ϕ be a hybrid-logical formula in which the nominals a and b do not occur. For any assignment g, it is the case that $\mathfrak{M}, g, g(a) \models \phi$ if and only if $\mathfrak{M}^*, g \models ST_a(\phi)$ (and the same for ST_b).*

Proof. Induction on the structure of ϕ. ■

Thus, hybrid logic, considered as a language for talking about models, has the same expressive power as the fragment of first-order logic obtained by taking the image of hybrid logic under the translation ST_a.

First-order logic with equality can be translated into the hybrid logic $\mathcal{H}(\forall)$ by the translation HT given below.

$$\begin{array}{rcl}
HT(p^*(a)) & = & @_a p \\
HT(R(a,c)) & = & @_a \Diamond c \\
HT(a = c) & = & @_a c \\
HT(\phi \wedge \psi) & = & HT(\phi) \wedge HT(\psi) \\
HT(\phi \to \psi) & = & HT(\phi) \to HT(\psi) \\
HT(\bot) & = & \bot \\
HT(\forall a \phi) & = & \forall a HT(\phi)
\end{array}$$

The translation HT is truth-preserving.

Proposition 1.4. *Let $\mathfrak{M}$ be a model for hybrid logic. For any first-order formula ϕ and any assignment g, it is the case that $\mathfrak{M}^*, g \models \phi$ if and only if $\mathfrak{M}, g \models HT(\phi)$.*

Proof. Induction on the structure of ϕ. ■

Thus, in the sense above the hybrid logic $\mathcal{H}(\forall)$ has the same expressive power as first-order logic with equality. It is implicit in the proposition above that the first-order formula ϕ is a formula of the first-order language defined by the grammar given earlier in the previous subsection. The history of the above observations goes back to the work of Arthur Prior, which we shall come back to in the next section.

In a way similar to the above translation, a fragment of first-order logic with equality which is called the *bounded fragment* can be translated into the hybrid logic $\mathcal{H}(\downarrow)$. This was pointed out in Areces et al. (2001a). The bounded fragment is obtained from the above grammar for first-order logic by replacing the clause $\forall aS$ by the new clause $\forall c(R(a,c) \rightarrow S$ where it is required that the variables a and c are distinct. In Areces et al. (2001a) a number of independent semantic characterizations of the bounded fragment are given. A translation from the bounded fragment to the hybrid logic $\mathcal{H}(\downarrow)$ can be obtained by replacing the last clause in the translation HT above by the following.

$$HT(\forall c(R(a,c) \rightarrow \phi)) = @_a \Box \downarrow c HT(\phi)$$

It is straightforward to check that Proposition 1.4 still holds, hence, the hybrid logic $\mathcal{H}(\downarrow)$ has the same expressive power as the bounded fragment of first-order logic (note that for any formula ϕ of $\mathcal{H}(\downarrow)$, the formula $ST_a(\phi)$ is in the bounded fragment).

1.2 The Origin of Hybrid Logic in Prior's Work

In this section we discuss the work of Arthur Prior, and we describe how hybrid logic has its origin in his work. The precise origin of hybrid logic is Prior's hybrid tense logic, which is a hybridized version of ordinary tense logic. Arthur Prior (1914–1969) is usually considered the founding father of modern temporal logic, his main contribution being the formal logic of tenses. In his memorial paper on Prior (Kenny 1970), A.J.P. Kenny summed up Prior's life and work as follows.

> Prior's greatest scholarly achievement was undoubtedly the creation and development of tense-logic. But his research and reflection on this topic led him to elaborate, piece by piece, a whole metaphysical system of an individual and characteristic stamp. He had many different interests at different periods of his life, but from different angles he constantly returned to the same central and unchanging themes. Throughout his life, for instance, he worked away at the knot of problems surrounding determinism: first as a predestinarian theologian, then as a moral philosopher, finally as a metaphysician and logician (Kenny 1970, p. 348).

Prior's reflections on determinism and other issues related to the philosophy of time were a major motivation for his formulation of tense logic. With the aim of discussing tense logic and hybrid tense logic further, we shall give a formal definition of hybrid tense logic: The language of hybrid tense logic is simply the language of hybrid logic defined above except that there are two modal operators,

namely G and H, instead of the single modal operator $\Box$. The two new modal operators are called *tense operators*. The semantics of hybrid tense logic is the semantics of hybrid logic, cf. earlier, with the clause for $\Box$ replaced by clauses for the tense operators G and H.

$$\begin{array}{lll} \mathfrak{M}, g, w \models G\phi & \text{iff} & \text{for any } v \in W \text{ such that } wRv, \mathfrak{M}, g, v \models \phi \\ \mathfrak{M}, g, w \models H\phi & \text{iff} & \text{for any } v \in W \text{ such that } vRw, \mathfrak{M}, g, v \models \phi \end{array}$$

Thus, there are now two modal operators, namely one that "looks forwards" along the accessibility relation R and one that "looks backwards". In tense logic the elements of the set W are called *moments* or *instants* and the accessibility relation R is now also called the *earlier-later relation*.

It is straightforward to modify the translations ST_a and HT in the previous section such that translations are obtained between a tense-logical version of $\mathcal{H}(\forall)$ and first-order logic with equality. The first-order logic under consideration is what Prior called *first-order earlier-later logic*. Given the translations, it follows that Prior's first-order earlier-later logic has the same expressive power as the tense-logical version of $\mathcal{H}(\forall)$, that is, hybrid tense-logic.

Now, Prior introduced hybrid tense logic in connection with what he called four grades of tense-logical involvement. The four grades were presented in the book Prior (1968), Chapter XI (also Chapter XI in the new edition (Hasle et al. 2003)). Moreover, see the book Prior (1967), Chapter V.6 and Appendix B.3–4. For a more general discussion of the four grades, see the posthumously published book Fine and Prior (1977). The stages progress from pure first-order earlier-later logic to what can be regarded as a pure tense logic, where the second grade is a "neutral" logic encompassing first-order earlier-later logic and tense logic on the same footing. The motivation for Prior's four grades of tense-logical involvement was philosophical. Prior considered instants to be "artificial" entities which due to their abstractness should not be taken as primitive concepts.

> ... my desire to sweep 'instants' under the metaphysical table is not prompted by any worries about their punctual or dimensionless character but purely by their abstractness. ... 'instants' as literal objects, or as cross-sections of a literal object, go along with a picture of 'time' as a literal object, a sort of snake which either eats its tail or doesn't, either has ends or doesn't, either is made of separate segments or isn't; and this picture I think we must drop (Prior 1967, p. 189).

Given the explicit reference to instants in first-order earlier-later logic, Prior found that first-order earlier-later logic gives rise to undesired ontological import. Instead of first-order earlier-later logic, he preferred tense logic.

> Some of us at least would prefer to see 'instants', and the 'time-series' which they are supposed to constitute, as mere logical constructions out of tensed facts (Hasle et al. 2003, p. 120).

This is why Prior's goal was to extend tense logic such that it could be considered as encompassing first-order earlier-later logic. Technically, the goal was to extend

tense logic such that first-order earlier-later logic could be translated into it. It was with this goal in mind Prior introduced what he called *instant-propositions*.

> What I shall call the third grade of tense-logical involvement consists in treating the instant-variables a, b, c, etc. as also representing propositions (Hasle et al. 2003, p. 124).

In the context of modal logic, Prior called such propositions *possible-world-propositions*. Of course, this is what we here call nominals. Prior also introduced the binder $\forall$ and what we here call satisfaction operators (he used the notation $T(a, \phi)$ instead of $@_a\phi$ for satisfaction operators). The extended tense-logic thus obtained is the logic he called third grade tense logic, hence, the third grade tense logic is identical to the tense-logical version of $\mathcal{H}(\forall)$, hybrid tense logic, which has the same expressive power as first-order earlier-later logic, as remarked above.

Prior gave an alternative, but equivalent, formulation of the third grade tense logic in which the satisfaction operator is replaced by a modal operator A called the *universal* modality (some authors call it the *global* modality). The universal modality has a fixed interpretation: The truth-condition is that a formula $A\phi$ is true (at any world) if and only if the formula ϕ is true at all worlds. Thus, the universal modality is interpreted using the universal binary relation. Formally, the clause for the satisfaction operator in the semantics is replaced by a clause for the modal operator A.

$$\mathfrak{M}, g, w \models A\phi \quad \text{iff} \quad \text{for any } v \in W, \mathfrak{M}, g, v \models \phi.$$

Thus, besides the tense operators G and H, the language under consideration here also contains the modal operator A. The two formulations of the third degree are equivalent since the satisfaction operator and the universal modality are interdefinable in the presence of nominals and the $\forall$ binder, this being the case as the formulas $A\phi \leftrightarrow \forall a(@_a\phi)$ and $@_a\phi \leftrightarrow A(a \rightarrow \phi)$ are valid.

Prior's fourth grade tense logic is obtained from the third grade tense logic by replacing the satisfaction operator (or the universal modality in the alternative formulation of the third grade) by a defined modal operator L such that

$$\mathfrak{M}, g, w \models L\phi \quad \text{iff} \quad \text{for any } v \in W \text{ such that } wR^*v, \mathfrak{M}, g, v \models \phi$$

where the binary relation R^* is the reflective, symmetric, and transitive closure of the earlier-later relation R. Prior considered two ways to define the operator L in what he took to be purely tense-logical terms. In the first case he allowed what amounts to infinite conjunctions of formulas. If infinite conjunctions are allowed, the operator L can be defined by the conventions that

$$L\phi = L^0\phi \wedge L^1\phi \wedge \ldots$$

and

$$\begin{aligned} L^0\phi &= \phi \\ L^{n+1}\phi &= GL^n\phi \wedge HL^n\phi \end{aligned}$$

Note that for any given natural number k, $L^k\phi$ is a formula in the object language (which does not involve natural numbers). For example, if $k = 1$ and $\phi = p$, then $L^1\phi = Gp \wedge Hp$. In the second case Prior assumed time to have a structure making $L\phi$ equivalent to

$$L^0\phi \wedge L^1\phi \wedge \ldots \wedge L^k\phi$$

for some fixed natural number k whereby infinite conjunctions are avoided. If for example time is linear, that is, transitive, backwards linear, and forwards linear, then $k = 1$ will do. If time is branching, that is, transitive and backwards linear, then $k = 2$ will do. In whichever way the operator L is defined, the fourth grade tense logic has the same expressive power as first-order earlier-later logic if it is assumed that the time-series is unique, that is, if it is assumed that any two instants are connected by some number of steps in either direction along the earlier-later relation R. For Prior it was natural to assume that the time-series is unique, as is witnessed by the following quotation.

> For is not the question as to whether 'our' time-series (whatever its structure) is unique, a genuine one? I would urge the following consideration against saying that it is, or at all events against saying it too hurriedly: It is only if we have a more-or-less 'Platonistic' conception of what a time-series is, that we can raise this question. If, as I would contend, it is only by tensed statements that we can give the cash-value of assertions which purport to be about 'time', the question as to whether there are or could be unconnected time-series is a senseless one. We think we can give it a sense because it is as easy to draw unconnected lines and networks as it is to draw connected ones; but these diagrams cannot represent *time*, as they cannot be translated into the basic non-figurative temporal language (Prior 1967, pp. 198–199).

The reason why the fourth grade tense logic has first-order expressive power when the time-series is unique, is that the fourth-grade modality L then has the same effect as the universal modality A which is used in (the alternative formulation of) the third-grade logic, and the third-grade logic has first-order expressive power, as we argued above. This is discussed in more detail in the paper Braüner (2002b).

To sum up, Prior obtained tense logics having the same expressive power as first-order earlier-later logic, namely the third and fourth grade tense logics, by adding to ordinary tense logic further expressive power in the form of hybrid-logical machinery (and in the case of the fourth grade tense logic by making appropriate assumptions about the structure of time, including an assumption that the time-series is unique). So Prior clearly reached his technical goal. Prior also found that he reached his philosophical goal, namely that of avoiding an ontology including instants.

> The 'entities' which we 'countenance' in our 'ontology' ... depend on what variables we take seriously as individual variables in a first-order theory, i.e. as subjects of predicates rather than as *assertibilia* which may be qualified by modalities. If we prefer to handle instant-variables, for example, or person-variables, as subjects of predicates, then we may be taken to believe in the existence of instants, or of persons. If, on the other hand, we prefer to treat either of these as *propositional* variables, i.e. as arguments of truth-functions and of modal functions, then we may be taken as *not* believing in the existence of instants, etc. (they don't exist; rather, they are or are not the case) (Hasle et al. 2003, p. 220).

However, it has been debated whether or not Prior managed to avoid an instant ontology. We shall return to this later in Sect. 1.2.1 (where we also return to the person-variables mentioned in the quotation above).

The discussion on Prior's third grade tense logic and first-order earlier-later logic is closely related to the discussion on two different conceptions of time, namely the *A-series* and *B-series* conceptions, a terminology introduced in 1908 by the philosopher McTaggart (cf. McTaggart 1908). According to the A-series conception, also called the *dynamic* view, the past, present, and future tenses are primitive concepts from which other temporal concepts, in particular instants and the earlier-later relation, are to be derived. On the other hand, according to the B-series conception, also called the *static* view, instants and the earlier-later relation are primitive. The A-series conception embodies the local way in which human beings experience the flow of time whereas the B-series conception embodies a Gods-eye-view of time, where time is a sequence of objectively and tenselessly existing instants. It is notable that representations of both the A-series and B-series conceptions can be found in natural language (the A-series conception of course in the form of tense inflection of verbs and the B-series conception in particular in the form of nominal constructions like "5 o'clock 10 May 2007"). Of course, first-order earlier-later logic is associated with the B-series conception and Prior's third grade tense logic is associated with the A-series conception, which was Prior's own view, as succinctly expressed in the following quotation.

> So far, then, as I have anything that you could call a philosophical creed, its first article is this: I believe in the reality of the distinction between past, present, and future. I believe that what we see as a progress of events *is* a progress of events, a *coming to pass* of one thing after another, and not just a timeless tapestry with everything stuck there for good and all (Prior 1996, p. 47).

The discussion of A-series and B-series is reflected in discussions of time in Artificial Intelligence, see the paper Galton (2006). The paper by Patrick Blackburn (2006) discusses all the above issues as well as a number of other issues in hybrid logic and their origin in Prior's work. The above issues are also discussed in many papers of the collection (Copeland 1996), in particular in Richard Sylvan's paper (1996). See the paper Øhrstrøm and Hasle (1993), the book Øhrstrøm and Hasle (1995), and the handbook chapters Øhrstrøm and Hasle (2005b) and Øhrstrøm and Hasle (2005a) for general accounts of Prior's work. See also the encyclopedia article Copeland (2007). A recent assessment of Prior's philosophical and logical views can be found in Müller (2007).

1.2.1 Did Prior Reach His Philosophical Goal?

It has been debated whether Prior reached his philosophical goal with the third and fourth grade logics, namely that of avoiding an ontology including instants.

According to one criticism, the ontological import of the third and fourth grade logics is the same as the ontological import of first-order earlier-later logic since the third and fourth grade logics involve what are considered direct analogies to first-order primitives, in particular, nominals are considered a direct analogy to first-order variables and the $\forall$ binder is considered a direct analogy to the first-order $\forall$ quantifier.[8] Such a criticism can be found in Sylvan's paper (1996).[9] Note that this is a philosophical, not a technical, discussion. The technical, to be precise, mathematical, result that first-order earlier-later logic and the third grade logic (as well as the fourth grade logic in the light of appropriate assumptions on the structure of time) have the same expressive power, in the sense that there are truth-preserving translations in both directions between the logics, does not itself give an answer to the philosophical question as to whether the logics have the same ontological import.

Clearly, Prior's view on logic differs in a number of ways from the views held by most contemporary logicians, in particular logicians inclined towards model-theory. A criticism from the perspective of contemporary model-theory has been raised by Blackburn in the paper (2006).

> If the fundamental unit of logical modeling is a formal language *together with* a set-theoretical interpretation, then it makes little sense to claim, for example, that first-order logic automatically brings greater ontological commitment than (say) propositional modal logic. Under the model-theoretic conception, both make use of the same set-theoretic structures, so their ontological commitments are at least prima facie identical. Perhaps arguments could be mounted (based, perhaps, on the fact that modal logic is decidable and has the finite model property) that modal logic commits us to less. But such arguments

[8] It appears that this criticism presupposes a view on nominals according to which a nominal is a symbol that refers to something, like a first-order variable does. As remarked in Footnote 4 in Sect. 1.1.1, there is an alternative view on nominals according to which a nominal is viewed as a description of the content of an instant. It is not clear whether the criticism applies if this alternative view on nominals is adopted.

[9] Sylvan actually argues that it is not necessary to reduce first-order earlier-later logic (the B-series conception of time) to tense logic (the A-series conception), or vice versa. Sylvan points out that Prior regarded tense-logical postulates as being capable of giving the meaning of statements like 'time is continuous' and 'time is infinite both ways' (cf. Prior 1967, p. 74). To this Sylvan responds as follows.

> Time is an item, a theoretical object, which bears both the tensed and the temporally ordered properties which the item in question genuinely has. . . .
>
> Part of the elegance of such a simple characterization of Time is that it neatly decouples the stable sense of 'time' . . . from various vexed issues as to exactly which properties the item genuinely has (and so from what Time is 'really' like). Whichever it should have, under evolving or under alternative theories, the item can remain abstractly one and the same. Naturally, tight coupling remains between the item and its properties; but it is not a meaning connection, it is a theory-dependent linkage (Sylvan 1996, p. 114).

Sylvan sees Prior's goal to reduce the B-series talk to A-series talk as part of a more general, and in Sylvan's view overdeveloped, reductionist inclination of analytic philosophers, which also encompasses philosophers having the converse reduction as a goal, that is, having the goal of reducing A-series to B-series (cf. Sylvan 1996, p. 112).

> would have to be carefully constructed. In the light of modern correspondence theory, simple knockdown arguments based on the presence or absence of explicit quantifiers in the object language are unconvincing (Blackburn 2006, pp. 358–359).

Another criticism raised by Blackburn in the paper (Blackburn 2006) has to do with a logic Prior called *egocentric logic*. We now briefly describe this logic. Egocentric logic is technically the same as the third grade tense logic, but the points in the Kripke semantics are now taken to represent persons, not instants, and the accessibility relation relates two persons if and only if the second person is taller than the first one. Of course, just as the third grade tense logic is a modal-logical counterpart to first-order earlier-later logic, egocentric logic is a modal-logical counterpart to a first-order logic which technically is the same as first-order earlier-later logic, but where the points in a model are taken to represent persons. What was observed by Prior is that egocentric logic is just an instance of a general relationship: Any first-order logic has a modal-logical counterpart, whatever signature the first-order logic has and whatever the points in a model are taken to represent. Thus, in this sense there is nothing special about tense logic. But Prior considered tense logic to have a privileged status that distinguishes it from other logics, in particular egocentric logic.

> Tense logic is for me, if I may use the phrase, *metaphysically fundamental*, and not just an artificially torn-off fragment of the first-order theory of the earlier-later relation. Egocentric logic is a different matter; I find it hard to believe that individuals really are just propositions of a certain sort, or just 'points of view', or that the real world of individuals is just a logical construction out of such points of view (Hasle et al. 2003, p. 232).

Thus, as the quotation indicates, Prior considered tense logic to have a special philosophical status, but in the sense described above, there is nothing special about tense logic. This calls for an explanation. Prior concluded the following.

> So far as I can see, there is nothing philosophically disreputable in saying that (i) persons just *are* genuine individuals, so that their figuring as individual variables in a first-order theory needs no explaining (*this* first-order theory being, on the contrary, the only way of giving sense to its 'modal' counterpart), whereas (ii) instants are *not* genuine individuals, so that *their* figuring as values of individual variables *does* need explaining, and it is the related 'modal' logic (tense logic) which gives the first-order theory what sense it has (Hasle et al. 2003, pp. 219–220).

However, Prior's conclusion is criticized in Blackburn's paper for being unsatisfactorily justified, which is in line with the other criticism expressed in the above quotation from Blackburn's paper.

1.3 The Development of Hybrid Logic Since Prior

In this section we outline the development of hybrid logic since Prior. We shall present a selection of works rather than trying to be encyclopedic. See also the handbook chapter Areces and ten Cate (2007).

The first completely rigorous definition of hybrid logic was given in Robert Bull's paper (1970) which in 1970 appeared in a special issue of the journal *Theoria* in memory of Prior. Bull introduces a third sort of propositional symbols where a propositional symbol is assumed to be true exactly at one branch ("course of events") in a branching time model. This idea of sorting propositional symbols according to restrictions on their interpretations has later been developed further by a number of authors, see Section 5 of the paper Blackburn and Tzakova (1999) as well as Section 9 of the paper Blackburn (2000a) for discussions. The idea of sorting is also discussed in the unpublished manuscript (Goranko 2000).

The hybrid logical machinery originally invented by Prior in the late 1960s was reinvented in the 1980s by Solomon Passy and Tinko Tinchev from Bulgaria, see the paper Passy and Tinchev (1985) as well as Passy and Tinchev (1991). Rather than ordinary modal logic, this work took place in connection with the much more expressive Propositional Dynamic Logic.

A major contribution in the 1990s was the introduction of the $\downarrow$ binder by Valentin Goranko, see the papers Goranko (1994, 1996).[10] Since then, hybrid logic with the $\downarrow$ binder has been extensively studied by a number of people, notably Patrick Blackburn and his co-authors, for example, in the paper Blackburn and Seligman (1995) it is shown that this logic does not have the finite model property and that the logic is undecidable.[11] Also, various expressivity results are given in Blackburn and Seligman (1995). See the paper Areces et al. (2001a) for a number of model-theoretic aspects. A very comprehensive study of the model-theory of hybrid logic is the PhD thesis of Balder ten Cate (2004).

Also the weaker hybrid logic obtained by omitting both of the binders $\downarrow$ and $\forall$ has been the subject of extensive exploration. An early work on the binder-free hybrid logic (but including the strong universal modality) is the paper by Gargov and Goranko (1993). It turns out that the binder-free logic and a number of variants of it are decidable. In the paper Areces et al. (1999), a number of complexity results are given for hybrid modal and tense logics over various classes of frames, for example arbitrary, transitive, linear, and branching. It is remarkable that the satisfiability problem of the binder-free hybrid logic over arbitrary frames is decidable in Polynomial Space (PSPACE), which is the same as the complexity of deciding satisfiability in ordinary modal logic. Thus, hybridizing ordinary modal logic gives more expressive power, but the complexity stays the same.

[10] A variation of the $\downarrow$ binder (called the "freeze" quantifier) was actually introduced already in 1989 in connection with real-time logics, see the paper by Alur and Henzinger (1989). See also the survey Alur and Henzinger (1992). The $\downarrow$ binder and the freeze quantifier were discovered independently of each other.

[11] To prove these results, the paper Blackburn and Seligman (1995) introduces a proof technique called the spy-point technique, which later has been used in many other connections.

It is remarkable that first-order hybrid logic offers precisely the features needed to prove interpolation theorems.[12] While interpolation fails in a number of well-known first-order modal logics, their hybridized counterparts have this property, see the paper Areces et al. (2003) as well as Blackburn and Marx (2003). The first paper gives a model-theoretic proof of interpolation whereas the second paper gives an algorithm for calculating interpolants based on a tableau system.[13]

A number of papers have dealt with axioms for hybrid logic (for example Blackburn (1993), Blackburn and Tzakova (1999), Blackburn and ten Cate (2006)). The paper Blackburn and Tzakova (1999) gives an axiom system for hybrid logic and shows the remarkable result that if the axiom system is extended with a set of additional axioms which are *pure* formulas (that is, formulas where all propositional symbols are nominals), then the extended axiom system is complete with respect to the class of frames validating the axioms in question.[14] Pure formulas correspond to first-order conditions on the accessibility relation (cf. the translation ST_a in Sect. 1.1.3), so axiom systems for new hybrid logics with first-order conditions on the accessibility relation can be obtained in a uniform way simply by adding axioms as appropriate. So, if for example the formula $\downarrow c \Box \neg c$ is added as an axiom, then the resulting system is complete with respect to irreflexive frames, cf. Sect. 1.1.2. The paper Blackburn and ten Cate (2006) investigates orthodox proof-rules (which are proof-rules without side-conditions) in axiom systems, and it is shown that if one requires extended completeness using pure formulas, then unorthodox proof-rules are indispensable in axiom systems for binder-free hybrid logic. However, an axiom system can be given only involving orthodox proof-rules for the stronger hybrid logic including the $\downarrow$ binder. Another axiom system for hybrid logic is given in the paper (Braüner 2006) by the present author and an axiom system for first-order hybrid logic is given in the paper Braüner (2005) also by the present author. In the paper Gabbay and Malod (2002) an axiom system is given for a logic similar to hybrid logic, obtained by extending ordinary modal logic with first-order machinery for naming worlds.

Besides giving an axiom system for standard classical hybrid logic, the paper Braüner (2006) also gives axiom systems for intuitionistic and paraconsistent hybrid logic. The paper ten Cate and Litak (2007) gives hybrid-logical axiom systems that are sound and complete with respect to topological semantics, that is, generalisations of the standard Kripke semantics where the modal operator is interpreted in terms of a topology on the set of possible worlds. Strictly speaking, the topological

[12] The interpolation theorem for propositional logic says that for any valid formula $\phi \rightarrow \psi$ there exists a formula θ containing only the common propositional symbols of ϕ and ψ such that the formulas $\phi \rightarrow \theta$ and $\theta \rightarrow \psi$ are valid. Interpolation theorems for other logics are formulated in an analogous fashion.

[13] An unexplored line of work is to find out whether interpolation can be proved in other ways, for example using proof systems like the linear reasoning systems which in Fitting (1984) are used to prove interpolation for some particular propositional and first-order modal logics.

[14] See ten Cate (2004) for semantic characterizations of frame classes definable by pure hybrid-logical formulas.

semantics only generalise Kripke semantics where the frames are reflexive and transitive, but the paper ten Cate and Litak (2007) also considers an even more general kind of semantics called neighbourhood semantics which generalises all Kripke semantics. Topological semantics is interesting for a number of reasons, one being that it is applicable for spatial reasoning (topological spaces are abstractions from metric spaces which in turn are abstractions from Euclidean space). A major reason for the interest in neighbourhood semantics is that it does not validate the formula $\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$ and nor does it validate the standard modal-logical rule called necessitation, that is, from ϕ derive $\Box\phi$ (the Kripke semantics validates both, but for some applications this is undesirable, for example, if the modal operator represents an agent's knowledge, then these two validities together imply that the agent is logical omniscient, that is, the agent's knowledge is closed under logical consequence, which at least for human agents is implausible). Further work on topological semantics for hybrid logic can be found in the paper Sustretov (2009).

Work in resolution calculi and model-checking for hybrid logic is in the early phases, see the papers Areces et al. (2001b) and Areces and Heguiabehere (2002) for resolution calculi and see the paper Franceschet and de Rijke (2006) and Lange (2009) for results on model-checking.

Tableau, Gentzen, and natural deduction style proof-theory for hybrid logic work very well compared to ordinary modal logic. Usually, when a modal tableau, Gentzen, or natural deduction system is given, it is for one particular modal logic and it has turned out to be problematic to formulate such systems for modal logics in a uniform way without introducing metalinguistic machinery. This can be remedied by hybridization, that is, hybridization of modal logics enables the formulation of uniform tableau, Gentzen, and natural deduction systems for wide classes of logics. Blackburn's paper (2000a) introduces a tableau system for hybrid logic that has this desirable feature: Analogous to the axiom systems of Blackburn and Tzakova (1999) and Blackburn and ten Cate (2006), completeness is preserved if the tableau system is extended with a set of pure axioms, that is, a set of pure formulas that are allowed to be added to a tableau during the tableau construction. See Hansen (2007) for another tableau system for hybrid logic.

The tableau system of Blackburn (2000a) is the basis for a decision procedure for the binder-free fragment of hybrid logic given in the paper by the present author together with Thomas Bolander (2006). The tableau-based decision procedures of Bolander and Braüner (2006) have been further developed in the papers by Bolander and Blackburn (2007, 2009). The paper Mayer and Cerrito (2010) presents another tableau-based decision procedure for hybrid logic. Other decision procedures for hybrid logics, which also are based on proof-theory, are given in the papers Kaminski and Smolka (2007, 2009. The procedures of these two papers are based on the higher-order formulation of hybrid logic (involving the simply typed λ-calculus) given in Hardt and Smolka (2007).

The paper by Hansen et al. (2008) gives a tableau-based decision procedure for many-valued hybrid logic, that is, hybrid logic where the two-valued classical logic basis has been generalized to a many-valued logic basis involving a truth-value space having the structure of a finite Heyting algebra (this many-valued hybrid

logic can also be seen as a hybridized version of the many-valued modal logic given in the papers Fitting (1992a), Fitting (1992b), and Fitting (1995)). The paper Hansen (2010) gives a tableau-based decision procedure for a hybridized version of a dynamic epistemic logic called public announcement logic.

Natural deduction style proof-theory of propositional and first-order hybrid logic has been explored in the papers by the present author Braüner (2004a, 2005) (see Sect. 1.4 for the propositional natural deduction system). The paper Braüner (2004a) also gives a Gentzen system for hybrid logic. These natural deduction and Gentzen systems can be extended with additional proof-rules corresponding to first-order conditions on the accessibility relations expressed by geometric theories; this is analogous to extending tableau and axiom systems with pure axioms.[15] The paper (Braüner and de Paiva 2006) by the present author together with Valeria de Paiva gives a natural deduction system for intuitionistic hybrid logic. In the context of situation theory, Gentzen and natural deduction systems for logics similar to hybrid logics were explored in the early 1990s by Jerry Seligman, see the overview in Seligman (2001). In the paper (Braüner 2004b) by the present author, a natural deduction system given in Seligman's paper (1997) is compared to the system of Braüner (2004a).

The fact that hybridization of modal logics enables the formulation of uniform tableau, Gentzen, and natural deduction systems for wide classes of logics is discussed in detail in the paper (Braüner 2007) by the present author (see Sect. 1.6).

The development of hybrid logic is only outlined above, in particular, we have only outlined hybrid-logical proof-theory. The proof-theory of hybrid logic will be the main issue in what follows.

1.4 Natural Deduction for Hybrid Logic

In this section we shall give a sound and complete natural deduction system for hybrid logic. Moreover, we shall show how to extend the natural deduction system with additional derivation rules corresponding to first-order conditions on the accessibility relation. The conditions we consider are expressed by geometric theories. Different geometric theories give rise to different hybrid logics, so natural deduction systems for new hybrid logics can be obtained in a uniform way simply by adding derivation rules as appropriate. Furthermore, we prove a normalisation theorem which says that any derivation can be rewritten to a normal derivation by repeated applications of reduction steps. Normal derivations satisfy a version of the subformula property called the quasi-subformula property. The natural deduction system given in the present section was originally published in the paper Braüner (2004a). Only central theorems and proofs are included in the present section, more details can be found in that paper as well as the book Braüner (2011).

[15] Like frame classes definable by pure formulas, frame classes definable by geometric theories can be given a semantic characterization, see the remark at the end of Sect. 1.4.3.

1.4.1 The Basics of Natural Deduction Systems

Before giving our hybrid-logical natural deduction system, we shall sketch the basics of natural deduction and fix terminology.

Natural deduction style derivation rules for ordinary classical first-order logic were originally introduced by Gerhard Gentzen in 1969 and later on developed much further by Dag Prawitz in 1965; 1971. See Troelstra and Schwichtenberg (1996) for a general introduction to natural deduction systems. With reference to Gentzen's work, Prawitz made the following remarks on the significance of natural deduction.

> ... the essential logical content of intuitive logical operations that can be formulated in the languages considered can be understood as composed of the atomic inferences isolated by Gentzen. It is in this sense that we may understand the terminology *natural* deduction.
>
> Nevertheless, Gentzen's systems are also natural in the more superficial sense of corresponding rather well to informal practices; in other words, the structure of informal proofs are often preserved rather well when formalised within the systems of natural deduction (Prawitz 1971, p. 245).

The method of reasoning in natural deduction systems is called "forwards" reasoning: When you want to find a derivation of a certain formula you start with the rules and try to build a derivation of the formula you have in mind. This is contrary to tableau systems which are backward reasoning systems since you explicitly start with a particular formula and try to build a proof of it using tableau rules, cf. Sect. 1.5.1.

A natural deduction derivation has the form of a finite tree where the nodes are labelled with formulas such that for any formula occurrence ϕ in the derivation, either ϕ is a leaf of the derivation or the immediate successors of ϕ in the derivation are the premises of a rule-instance which has ϕ as the conclusion. In what follows, the metavariables π, τ, ... range over derivations. A formula occurrence that is a leaf but is not the conclusion of a rule-instance with zero premises, is called an *assumption* of the derivation. The root of a derivation is called the *end-formula* of the derivation. All assumptions are annotated with numbers. An assumption is either *undischarged* or *discharged*. If an assumption is discharged, then it is discharged at one particular rule-instance and this is indicated by annotating the assumption and the rule-instance with identical numbers. We shall often omit this information when no confusion can occur. A rule-instance annotated with some number discharges all undischarged assumptions that are above it and are annotated with the number in question, and moreover, are occurrences of a formula determined by the rule-instance.

Two assumptions in a derivation belong to the same *parcel* if they are annotated with the same number and are occurrences of the same formula, and moreover, either are both undischarged or have both been discharged at the same rule-instance. Thus, in this terminology rules discharge parcels. We shall make use of the standard notations

$$\begin{array}{c} [\phi^r] \\ \vdots\,\pi \\ \psi \end{array} \qquad\qquad \begin{array}{c} \vdots\,\tau \\ \phi \\ \vdots\,\pi \\ \psi \end{array}$$

which mean a derivation π where ψ is the end-formula and $[\phi^r]$ is the parcel consisting of all undischarged assumptions that have the form ϕ^r, and moreover, a derivation π where ψ is the end-formula and a derivation τ with end-formula ϕ has been substituted for each of the undischarged assumptions indicated by $[\phi^r]$.

We shall make use of the following conventions. The metavariables Γ, Δ, ... range over sets of formulas. A derivation π is called a *derivation of* ϕ if the end-formula of π is an occurrence of ϕ, and moreover, π is called a *derivation from* Γ if each undischarged assumption in π is an occurrence of a formula in Γ (note that numbers annotating undischarged assumptions are ignored). If there exists a derivation of ϕ from the empty set $\emptyset$, then we shall simply say that ϕ is *derivable*.

A characteristic feature of natural deduction is that there are two different kinds of rules for each connective; there are rules called introduction rules which introduce a connective (that is, the connective occurs in the conclusion of the rule, but not in the premises) and there are rules called elimination rules which eliminate a connective (the connective occurs in a premiss of the rule, but not in the conclusion). Introduction rules traditionally have names in the form $(\ldots I \ldots)$, and similarly, elimination rules traditionally have names in the form $(\ldots E \ldots)$.

The introduction and elimination rules for a connective are expected to satisfy Dag Prawitz' *inversion principle*.

> ... a proof of the conclusion of an elimination is already "contained" in the proofs of the premises when the major premise is inferred by introduction (Prawitz 1971, pp. 246–247).

(The major premise of an elimination rule is the premise that exhibits the connective being eliminated.) The history of the inversion principle goes back to Prawitz (1965). In the above formulation of the inversion principle it is not made explicit what is meant by requiring that some derivations (called proofs by Prawitz) "contain" a derivation of a certain formula, but it means that a derivation of the formula in question can be obtained by composition of derivations, that is, by substitution of derivations for undischarged assumptions. In the case of first-order logic, not only substitution of derivations for undischarged assumptions is allowed, but also substitution of variables for variables in derivations.

The inversion principle refers to a particular kind of formula occurrence in a derivation, namely a formula occurrence which is both introduced by an introduction rule and eliminated by an elimination rule. Such a formula occurrence is called a maximum formula. According to the inversion principle, a maximum formula can be considered a "detour" in the derivation, and it follows from the principle that the maximum formula can be removed by rewriting the derivation. This rewrite process is formalized in a kind of rewrite rules that are called proper reduction rules.

Some natural deduction systems also involve other kinds of reduction rules than proper reduction rules. A derivation is called normal if no reduction rules can be applied to it, and for most natural deduction systems a normalisation theorem can be proved which says that any derivation can be rewritten to such a normal derivation by repeated applications of reductions. In most natural deduction systems, normal derivations satisfy the subformula property which says that any formula in a derivation, save possibly some exceptions, is a subformula of the end-formula or one of the undischarged assumptions.

As is clear from the above, the inversion principle can be seen as a prerequisite for formulating a normalisation theorem since such a theorem is relative to a set of reduction rules. See in Sect. 1.6 for further discussion of the inversion principle.

1.4.2 Natural Deduction Rules for Hybrid Logic

In this subsection we give the natural deduction system for the hybrid logic $\mathcal{H}(\mathcal{O})$, where $\mathcal{O}$ is any subset of the set of binders $\{\forall, \downarrow\}$. The derivation rules for the system are given in Figs. 1.1 and 1.2. All formulas in the rules are satisfaction statements. Note that the rules $(\bot 1)$ and $(\bot 2)$ are neither introduction rules nor elimination rules (recall that $\neg\phi$ is an abbreviation for $\phi \to \bot$). Our natural deduction system for $\mathcal{H}(\mathcal{O})$ is obtained from the rules given in Figs. 1.1 and 1.2 by leaving out the rules for the binders that are not in the set $\mathcal{O}$. The system thus obtained will be denoted $\mathbf{N}_{\mathcal{H}(\mathcal{O})}$. So, for example, the system $\mathbf{N}_{\mathcal{H}(\forall)}$ is obtained by leaving out the rules $(\downarrow I)$ and $(\downarrow E)$. We shall discuss the side-condition on the rules $(\bot 1)$ and $(Nom1)$ in Sect. 1.4.4. Below we give an example of a derivation in the system $\mathbf{N}_{\mathcal{H}}$. The end-formula of the example derivation is the standard modal axiom K prefixed by a satisfaction operator (it is assumed that the nominal b in the derivation is new).

$$\dfrac{\dfrac{\dfrac{\dfrac{@_a\Box(\phi \to \psi)^3 \quad @_a\Diamond b^1}{@_b(\phi \to \psi)}(\Box E) \qquad \dfrac{@_a\Box\phi^2 \quad @_a\Diamond b^1}{@_b\phi}(\Box E)}{@_b\psi}(\to E)}{@_a\Box\psi}(\Box I)^1}{@_a(\Box\phi \to \Box\psi)}(\to I)^2}{@_a(\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi))}(\to I)^3$$

The natural deduction system $\mathbf{N}_{\mathcal{H}(\mathcal{O})}$ corresponds to the class of all frames, that is, the class of frames where no conditions are imposed on the accessibility conditions (this is formalized in the soundness and completeness result, Theorem 1.12). Hence, it is a hybrid version of the standard modal logic K.

$$\frac{@_a\phi \qquad @_a\psi}{@_a(\phi \wedge \psi)}\,(\wedge I) \qquad \frac{@_a(\phi \wedge \psi)}{@_a\phi}\,(\wedge E1) \qquad \frac{@_a(\phi \wedge \psi)}{@_a\psi}\,(\wedge E2)$$

$$\frac{\begin{array}{c}[@_a\phi]\\ \vdots\\ @_a\psi\end{array}}{@_a(\phi \to \psi)}\,(\to I) \qquad \frac{@_a(\phi \to \psi) \qquad @_a\phi}{@_a\psi}\,(\to E)$$

$$\frac{\begin{array}{c}[@_a\neg\phi]\\ \vdots\\ @_a\bot\end{array}}{@_a\phi}\,(\bot 1)^a \qquad \frac{@_a\bot}{@_c\bot}\,(\bot 2)$$

$$\frac{@_a\phi}{@_c@_a\phi}\,(@I) \qquad \frac{@_c@_a\phi}{@_a\phi}\,(@E)$$

$$\frac{\begin{array}{c}[@_a\Diamond c]\\ \vdots\\ @_c\phi\end{array}}{@_a\Box\phi}\,(\Box I)^b \qquad \frac{@_a\Box\phi \qquad @_a\Diamond e}{@_e\phi}\,(\Box E)$$

$$\frac{@_a\phi[c/b]}{@_a\forall b\phi}\,(\forall I)^c \qquad \frac{@_a\forall b\phi}{@_a\phi[e/b]}\,(\forall E)$$

$$\frac{\begin{array}{c}[@_a c]\\ \vdots\\ @_c\phi[c/b]\end{array}}{@_a\downarrow b\phi}\,(\downarrow I)^d \qquad \frac{@_a\downarrow b\phi \qquad @_a e}{@_e\phi[e/b]}\,(\downarrow E)$$

[a] ϕ is a propositional symbol (ordinary or a nominal)
[b] c does not occur free in $@_a\Box\phi$ or in any undischarged assumptions other than the specified occurrences of $@_a\Diamond c$
[c] c does not occur free in $@_a\forall b\phi$ or in any undischarged assumptions
[d] c does not occur free in $@_a\downarrow b\phi$ or in any undischarged assumptions other than the specified occurrences of $@_a c$

Fig. 1.1 Natural deduction rules for connectives

$$\frac{}{@_a a}\,(Ref) \qquad \frac{@_a c \qquad @_a\phi}{@_c\phi}\,(Nom1)^a \qquad \frac{@_a c \qquad @_a\Diamond b}{@_c\Diamond b}\,(Nom2)$$

[a] ϕ is a propositional symbol (ordinary or a nominal)

Fig. 1.2 Natural deduction rules for nominals

1. Symmetry	$\forall a \forall c(R(a,c) \to R(c,a))$
2. Antisymmetry	$\forall a \forall c((R(a,c) \wedge R(c,a)) \to a = c)$
3. Irreflexivity	$\forall a(R(a,a) \to \bot)$
4. Directedness	$\forall a \forall b \forall c((R(a,b) \wedge R(a,c)) \to \exists d(R(b,d) \wedge R(c,d)))$

Fig. 1.3 A sample of conditions on the accessibility relation

1.4.3 Conditions on the Accessibility Relation

In what follows we shall consider natural deduction systems obtained by extending $\mathbf{N}_{\mathcal{H}(\mathcal{O})}$ with additional derivation rules corresponding to first-order conditions on the accessibility relations. The conditions we consider are expressed by geometric theories. A first-order formula is *geometric* if it is built out of atomic formulas of the forms $R(a,c)$ and $a = c$ using only the connectives $\bot$, $\wedge$, $\vee$, and $\exists$. See Vickers (1988) for an introduction to geometric logic.

In what follows, the metavariables S_k and S_{jk} range over atomic first-order formulas of the forms $R(a,c)$ and $a = c$. Using the translation HT given in Sect. 1.1.3, atomic formulas of the mentioned forms are translated into hybrid logic as follows.

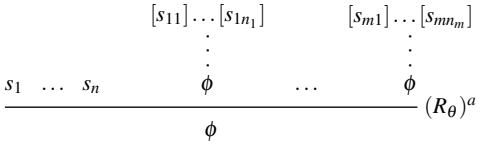
$$HT(R(a,c)) = @_a \Diamond c$$
$$HT(a = c) = @_a c$$

A *geometric theory* is a finite set of closed first-order formulas, each having the form $\forall \overline{a}(\phi \to \psi)$, where the formulas ϕ and ψ are geometric, $\overline{a}$ is a list $a_1, \ldots, a_l$ of variables, and $\forall \overline{a}$ is an abbreviation for $\forall a_1 \ldots \forall a_l$. It can be proved, cf. Simpson (1994), that any geometric theory is equivalent to a *basic geometric theory* which is a geometric theory in which each formula has the form

$$(*) \qquad \forall \overline{a}((S_1 \wedge \ldots \wedge S_n) \to \exists \overline{c} \bigvee_{j=1}^{m} (S_{j1} \wedge \ldots \wedge S_{jn_j}))$$

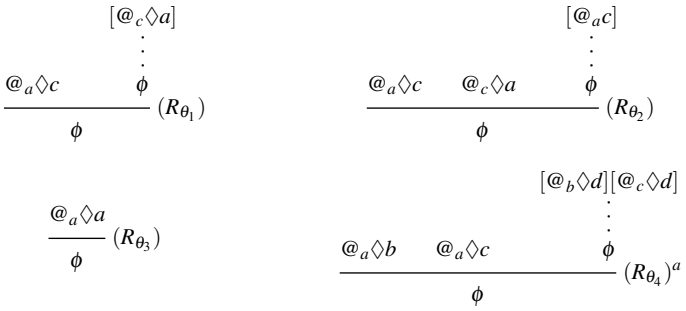
where $n, m \geq 0$ and $n_1, \ldots, n_m \geq 1$. For simplicity, we assume that the variables in the list $\overline{a}$ are pairwise distinct, that the variables in $\overline{c}$ are pairwise distinct, and that no variable occurs in both $\overline{c}$ and $\overline{a}$. A sample of formulas of the form $(*)$ displayed above is given in Fig. 1.3. Note that such a formula is a Horn clause if $\overline{c}$ is empty, $m = 1$, and $n_m = 1$. Thus, the first two formulas in Fig. 1.3 are Horn clauses. Also, note that the third formula in Fig. 1.3 is identical to $\forall a \neg R(a,a)$.

In what follows, the metavariables s_k and s_{jk} range over hybrid-logical formulas of the forms $@_a \Diamond c$ and $@_a c$. It turns out that basic geometric theories correspond



[a] None of the nominals in $\overline{c}$ occur free in ϕ or in any of the undischarged assumptions other than the specified occurrences of s_{jk}. (Recall that nominals are identified with first-order variables and that $\overline{c}$ are the first-order variables existentially quantified over in the formula θ)

Fig. 1.4 Natural deduction rules for geometric theories



[a] d does not occur free in ϕ or in any undischarged assumptions other than the specified occurrences of $@_c \Diamond d$ and $@_b \Diamond d$

Fig. 1.5 Rules corresponding to conditions on the accessibility relation

to straightforward natural deduction rules for hybrid logic: With a formula θ of the form displayed above, we associate the natural deduction rule (R_θ) given in Fig. 1.4, where s_k is of the form $HT(S_k)$ and s_{jk} is of the form $HT(S_{jk})$. If $\theta_1, \ldots, \theta_4$ are formulas of the forms given in Fig1.3, then the associated natural deduction rules $(R_{\theta_1}), \ldots, (R_{\theta_4})$ are the rules in Fig. 1.5. Note that the rule (R_{θ_3}) has zero non-relational premises. Now, let $\mathbf{T}$ be any basic geometric theory. The natural deduction system obtained by extending $\mathbf{N}_{\mathcal{H}(\mathcal{O})}$ with the set of rules $\{(R_\theta) \mid \theta \in \mathbf{T}\}$ will be denoted $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$. We shall assume that we are working with a fixed basic geometric theory $\mathbf{T}$ unless otherwise specified.

It is straightforward to check that if a formula θ of the form displayed above is a Horn clause, then the rule (R_θ) given in Fig. 1.4 can be replaced by the simpler rule below (which we have called (R_θ) too).

$$\frac{s_1 \quad \ldots \quad s_n}{s_{11}} \, (R_\theta)$$

Natural deduction rules corresponding to Horn clauses were discussed already in Prawitz (1971).

Remark: A semantic characterisation of frame classes definable by geometric theories can be found in the book Chang and Keisler (1990, p. 322, Exercise 5.2.24).[16] We briefly summarize this exercise. A *homomorphism* from a frame (W,R) to a frame (W',R') is a function f from W to W' such that for any $w,v \in W$, if wRv then $f(w)R'f(v)$. A *direct system* is a sequence of frames $\mathfrak{F}_1, \mathfrak{F}_2, \ldots$ together with a sequence of functions $f_1, f_2, \ldots$ such that each f_i is a homomorphism from $\mathfrak{F}_i$ to $\mathfrak{F}_{i+1}$. There is a natural way to define a limit frame, called a *direct limit*, for a direct system, the exact definition can be found in the above exercise. It can be shown that there exists a direct limit for any direct system and that this direct limit is unique up to isomorphism. Now, according to the exercise, a closed first-order formula ϕ is equivalent to the conjunction of the formulas in a geometric theory if and only if ϕ is preserved under direct limits, that is, whenever each of the frames $\mathfrak{F}_1, \mathfrak{F}_2, \ldots$ in a direct system validates ϕ, the direct limit also validates ϕ. It follows that a class of frames definable by a closed first-order formula is definable by a geometric theory if and only if the class in question is closed under direct limits.

1.4.4 Some Admissible Rules

Below we shall prove a small proposition regarding some admissible rules. A rule is *admissible* in a natural deduction system if, for every derivation of a formula ϕ from a set of formulas Γ involving the rule in question, there exists a derivation of ϕ from Γ not involving the rule. We first need a convention: The *degree* of a formula is the number of occurrences of non-nullary connectives in it.

Proposition 1.5. *The rules*

$$\dfrac{\begin{array}{c}[@_a\neg\phi]\\ \vdots \\ @_a\bot\end{array}}{@_a\phi}\,(\bot) \qquad\qquad \dfrac{@_a c \qquad @_a\phi}{@_c\phi}\,(Nom)$$

are admissible in $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$.

Proof. The proof that $(\bot)$ is admissible is along the lines of a similar proof for ordinary classical first-order logic given in Prawitz (1965). The proof that (Nom) is admissible is analogous. ■

Note in the proposition above that ϕ can be any formula; not just a propositional symbol. Thus, the rule $(\bot)$ generalises the rule $(\bot 1)$ whereas (Nom) generalises $(Nom1)$ (and the rule $(Nom2)$ as well). The side-conditions on the rules $(\bot 1)$ and $(Nom1)$ enable us to prove a normalisation theorem such that normal derivations

[16] This was pointed out to the author by Balder ten Cate (personal communication).

satisfy a version of the subformula property called the quasi-subformula property (Theorem 1.15). In the case with $(\bot 1)$, it is well-known from the literature that the subformula property does not hold without the side-condition (cf. Prawitz 1965, 1971). We shall return to the subformula property later.

1.4.5 Soundness and Completeness

The aim of this subsection is to prove soundness and completeness of the natural deduction system for hybrid logic. Recall that we are working with a fixed basic geometric theory $\mathbf{T}$. A model $\mathfrak{M}$ for hybrid logic is called a $\mathbf{T}$-*model* if and only if $\mathfrak{M}^* \models \theta$ for every formula $\theta \in \mathbf{T}$ (recall that $\mathfrak{M}^*$ is the first-order model corresponding to the hybrid-logical model $\mathfrak{M}$). Remark: Being a $\mathbf{T}$-model is really a property of the frame on which the model is based, the reason being that the formulas in $\mathbf{T}$ do not contain predicate symbols besides R and $=$.

The completeness proof we give is similar to the completeness proof in Blackburn (2000a). However, we use maximal consistent sets instead of Hintikka sets. Also, our proof is in some ways similar to the completeness proof in Basin et al. (1997).

Definition 1.6. A set of satisfaction statements Γ in $\mathcal{H}(\mathcal{O})$ is $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-*inconsistent* if and only if $@_a\bot$ is derivable from Γ in $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$ for some nominal a and Γ is $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-*consistent* if and only if Γ is not $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-inconsistent. Moreover, Γ is *maximal* $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-*consistent* if and only if Γ is $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-consistent and any set of satisfaction statements in $\mathcal{H}(\mathcal{O})$ that properly extends Γ is $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-inconsistent.

We shall frequently omit the reference to $\mathcal{H}(\mathcal{O})$ and $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$ where no confusion can occur. The definition above leads us to the lemma below.

Lemma 1.7. *If a set of satisfaction statements Γ is consistent, then for every satisfaction statement $@_a\phi$, either $\Gamma \cup \{@_a\phi\}$ is consistent or $\Gamma \cup \{@_a\neg\phi\}$ is consistent.*

Proof. Straightforward. ■

The Lindenbaum lemma below is similar to the Lindenbaum lemma in Basin et al. (1997).

Lemma 1.8 (Lindenbaum lemma). *Let $\overline{\mathcal{H}(\mathcal{O})}$ be the hybrid logic obtained by extending the set of nominals in $\mathcal{H}(\mathcal{O})$ with a countably infinite set of new nominals. Let $\phi_1, \phi_2, \phi_3, \ldots$ be an enumeration of all satisfaction statements in $\overline{\mathcal{H}(\mathcal{O})}$. For every $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-consistent set of satisfaction statements Γ, a maximal $\mathbf{N}_{\overline{\mathcal{H}(\mathcal{O})}} + \mathbf{T}$-consistent set of satisfaction statements $\Gamma^* \supseteq \Gamma$ is defined as follows. Firstly, Γ^0 is defined to be Γ. Secondly, Γ^{n+1} is defined by induction. If $\Gamma^n \cup \{\phi_{n+1}\}$ is $\mathbf{N}_{\overline{\mathcal{H}(\mathcal{O})}} + \mathbf{T}$-inconsistent, then Γ^{n+1} is defined to be Γ^n. Otherwise Γ^{n+1} is defined to be*

1. $\Gamma^n \cup \{\phi_{n+1}, @_b\psi, @_a\Diamond b\}$ *if* ϕ_{n+1} *is of the form* $@_a\Diamond\psi$;
2. $\Gamma^n \cup \{\phi_{n+1}, @_b\psi[b/c], @_a b\}$ *if* ϕ_{n+1} *is of the form* $@_a\downarrow c\psi$;
3. $\Gamma^n \cup \{\phi_{n+1}, @_a\psi[b/c]\}$ *if* ϕ_{n+1} *is of the form* $@_a\exists c\psi$;
4. $\Gamma^n \cup \{\phi_{n+1}, @_e \vee_{j=1}^m (s_{j1} \wedge \ldots \wedge s_{jn_j})[\overline{d},\overline{b}/\overline{a},\overline{c}]\}$ *if there exists a formula in* $\mathbf{T}$ *of the form* $\forall\overline{a}((S_1 \wedge \ldots \wedge S_n) \rightarrow \exists\overline{c} \vee_{j=1}^m (S_{j1} \wedge \ldots \wedge S_{jn_j}))$ *such that* $m \geq 1$ *and* $\phi_{n+1} = @_e(s_1 \wedge \ldots \wedge s_n)[\overline{d}/\overline{a}]$ *for some nominals* $\overline{d}$ *and* e; *and*
5. $\Gamma^n \cup \{\phi_{n+1}\}$ *if none of the clauses above apply.*

In clause 1, 2, and 3, b *is a new nominal that does not occur in* Γ^n *or* ϕ_{n+1}, *and similarly, in clause 4,* $\overline{b}$ *is a list of new nominals such that none of the nominals in* $\overline{b}$ *occur in* Γ^n *or* ϕ_{n+1}. *Finally,* Γ^* *is defined to be* $\cup_{n\geq 0}\Gamma^n$.

Proof. Firstly, Γ^0 is $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-consistent by definition and hence also $\mathbf{N}_{\overline{\mathcal{H}(\mathcal{O})}} + \mathbf{T}$-consistent. Secondly, to check that the consistency of Γ^n implies the consistency of Γ^{n+1}, we need to check the first four clauses in the definition of Γ^{n+1}. We only consider the first clause.

Assume conversely $@_f\bot$ is derivable from $\Gamma^n \cup \{\phi_{n+1}, @_b\psi, @_a\Diamond b\}$. Then $@_b\neg\psi$ is derivable from $\Gamma^n \cup \{\phi_{n+1}, @_a\Diamond b\}$ wherefore $@_a\Box\neg\psi$ is derivable from $\Gamma^n \cup \{\phi_{n+1}\}$ by the rule $(\Box I)$. But then $@_a\bot$ is derivable from the set of formulas $\Gamma^n \cup \{\phi_{n+1}\}$ as $\phi_{n+1} = @_a\neg\Box\neg\psi$.

It follows from each Γ^n being consistent that Γ^* is consistent. We now just need to prove that Γ^* is maximal consistent. Assume conversely that there exists a satisfaction statement $@_a\phi$ such that $@_a\phi \notin \Gamma^*$ as well as $@_a\neg\phi \notin \Gamma^*$, cf. Lemma 1.7. Then $\phi_p \notin \Gamma^p$ and $\phi_q \notin \Gamma^q$ where $\phi_p = @_a\phi$ and $\phi_q = @_a\neg\phi$. So $\Gamma^{p-1} \cup \{\phi_p\}$ is inconsistent and so is $\Gamma^{q-1} \cup \{\phi_q\}$. If $p < q$, then $\Gamma^{p-1} \subseteq \Gamma^{q-1}$ wherefore $\Gamma^{q-1} \cup \{\phi_p\}$ is inconsistent. Thus, Γ^{q-1} is inconsistent by Lemma 1.7. The argument is analogous if $q < p$. ■

Below we shall define a canonical model. First a small lemma.

Lemma 1.9. *Let* Δ *be a maximal* $\mathbf{N}_{\overline{\mathcal{H}(\mathcal{O})}} + \mathbf{T}$*-consistent set of satisfaction statements. Let* $\sim_\Delta$ *be the binary relation on the set of nominals of* $\overline{\mathcal{H}(\mathcal{O})}$ *defined by the convention that* $a \sim_\Delta a'$ *if and only if* $@_a a' \in \Delta$. *Then the relation* $\sim_\Delta$ *is an equivalence relation with the following properties.*

1. *If* $a \sim_\Delta a'$, $c \sim_\Delta c'$, *and* $@_a\Diamond c \in \Delta$, *then* $@_{a'}\Diamond c' \in \Delta$.
2. *If* $a \sim_\Delta a'$ *and* $@_a p \in \Delta$, *then* $@_{a'} p \in \Delta$.

Proof. It follows straightforwardly from Lemma 1.7 and the rules (*Ref*) and (*Nom1*) that $\sim_\Delta$ is reflexive, symmetric, and transitive. The first mentioned property follows from Lemma 1.7, the rule (*Nom2*), and the observation that $@_{a'}\Diamond c'$ is derivable from $\{@_{a'}\Diamond c, @_c c'\}$. The second property follows from Lemma 1.7 and the rule (*Nom1*). ■

Given a nominal a, we let $[a]$ denote the equivalence class of a with respect to $\sim_\Delta$. We now define a canonical model.

Definition 1.10 (Canonical model). Let Δ be a maximal $\mathbf{N}_{\overline{\mathcal{H}(\mathcal{O})}} + \mathbf{T}$-consistent set of satisfaction statements. Below we define a model

$$\mathfrak{M}^\Delta = (W^\Delta, R_1^\Delta, \ldots, R_m^\Delta, \{V_w^\Delta\}_{w \in W^\Delta})$$

and an assignment g^Δ for $\mathfrak{M}^\Delta$.

- $W^\Delta = \{[a] \mid a \text{ is a nominal of } \overline{\mathcal{H}(\mathcal{O})}\}$.
- $R^\Delta = \{([a],[c]) \mid @_a \Diamond c \in \Delta\}$.
- $V^\Delta_{[a]}(p) = \begin{cases} 1 & \text{if } @_a p \in \Delta. \\ 0 & \text{otherwise.} \end{cases}$
- $g^\Delta(a) = [a]$.

Note that the first property of $\sim_\Delta$ mentioned in Lemma 1.9 implies that R^Δ is well-defined, and similarly, the second property implies that V^Δ_w is well-defined. Now a truth lemma.

Lemma 1.11 (Truth lemma). *Let Γ be a $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$-consistent set of satisfaction statements. Then for any satisfaction statement $@_a\phi$, $@_a\phi \in \Gamma^*$ if and only if $\mathfrak{M}^{\Gamma^*}, g^{\Gamma^*}, [a] \models \phi$.*

Proof. Induction on the degree of ϕ. We only consider the case where ϕ is of the form $\Box\theta$.

Assume that $@_a\Box\theta \in \Gamma^*$. We then have to prove that $\mathfrak{M}^{\Gamma^*}, g^{\Gamma^*}, [c] \models \theta$ for any nominal c such that $[a]R^{\Gamma^*}[c]$, that is, such that $@_a\Diamond c \in \Gamma^*$. But $@_a\Diamond c \in \Gamma^*$ implies $@_c\theta \in \Gamma^*$ by the rule $(\Box E)$ and this implies $\mathfrak{M}^{\Gamma^*}, g^{\Gamma^*}, [c] \models \theta$ by induction.

Assume that $\mathfrak{M}^{\Gamma^*}, g^{\Gamma^*}, [a] \models \Box\theta$, that is, $\mathfrak{M}^{\Gamma^*}, g^{\Gamma^*}, [c] \models \theta$ for any nominal c such that $@_a\Diamond c \in \Gamma^*$. Now, if $@_a\neg\Box\theta \in \Gamma^*$, then also $@_a\Diamond\neg\theta \in \Gamma^*$ as $@_a(\neg\Box\theta \rightarrow \Diamond\neg\theta)$ is derivable. Therefore by definition of Γ^*, there exists a nominal b such that $@_b\neg\theta \in \Gamma^*$ and $@_a\Diamond b \in \Gamma^*$. But then $\mathfrak{M}^{\Gamma^*}, g^{\Gamma^*}, [b] \models \theta$ by assumption and hence $@_b\theta \in \Gamma^*$ by induction. Thus, we conclude that $@_a\neg\Box\theta \notin \Gamma^*$ and hence $@_a\Box\theta \in \Gamma^*$ by Lemma 1.7. ■

Now soundness and completeness.

Theorem 1.12. *Let ψ be a satisfaction statement and let Γ be a set of satisfaction statements. The first statement below implies the second statement (soundness) and vice versa (completeness).*

1. *ψ is derivable from Γ in $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$.*
2. *For any $\mathbf{T}$-model $\mathfrak{M}$ and any assignment g, if, for any formula $\theta \in \Gamma$, $\mathfrak{M}, g \models \theta$, then $\mathfrak{M}, g \models \psi$.*

Proof. Soundness is proved by induction on the structure of the derivation of ψ.

Completeness is proved as follows. We are done if Γ is inconsistent, cf. Proposition 1.5. So assume that Γ is consistent. Now, assume that ψ is not derivable from Γ and let $\psi = @_a\phi$. Then $\Gamma \cup \{@_a\neg\phi\}$ is consistent. Let $\Delta = (\Gamma \cup \{@_a\neg\phi\})^*$, cf. Lemma 1.8, and consider the model $\mathfrak{M}^\Delta$ and the assignment g^Δ. By Lemma 1.11, $\mathfrak{M}^\Delta, g^\Delta \models \theta$ for any formula $\theta \in \Gamma$, and also $\mathfrak{M}^\Delta, g^\Delta \models @_a\neg\phi$. But it can be proved that $\mathfrak{M}^\Delta$ is a $\mathbf{T}$-model, hence, the second statement in the theorem is contradicted. ■

1.4.6 Normalisation

In this subsection we give reduction rules for the natural deduction system $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$ and we prove a normalisation theorem. First some conventions. If a premise of a rule has the form $@_a c$ or $@_a \Diamond c$, then it is called a *relational premise*, and similarly, if the conclusion of a rule has the form $@_a c$ or $@_a \Diamond c$, then it is called a *relational conclusion*. Moreover, if an assumption discharged by a rule has the form $@_a c$ or $@_a \Diamond c$, then it is called a *relationally discharged assumption*. The premise of the form $@_a \phi$ in the rule $(\rightarrow E)$ is called *minor*. A premise of an elimination rule that is neither minor nor relational is called *major*. Note that the notion of a relational premise is defined in terms of rules; not rule-instances. A similar remark applies to the other notions above. Thus, a formula occurrence in a derivation might be of the form $@_a \Diamond c$ and also be the major premise of an instance of $(\rightarrow E)$. Note that the premises $s_1, \ldots, s_n$ in a (R_θ) rule are relational and that all the assumptions discharged by such a rule are relationally discharged.

A *maximum formula* in a derivation is a formula occurrence that is both the conclusion of an introduction rule and the major premise of an elimination rule. Maximum formulas can be removed by applying *proper reductions*. The rules for proper reductions are given below. We consider each case in turn. In what follows, we let $\pi[\overline{c}/\overline{a}]$ be the derivation π where each formula occurrence ψ has been replaced by $\psi[\overline{c}/\overline{a}]$.

$(\wedge I)$ followed by $(\wedge E1)$ (analogously in the case involving $(\wedge E2)$)

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_1\\ @_a\phi\end{array}\qquad\begin{array}{c}\vdots\,\pi_2\\ @_a\psi\end{array}}{@_a(\phi\wedge\psi)}}{@_a\phi} \quad\rightsquigarrow\quad \begin{array}{c}\vdots\,\pi_1\\ @_a\phi\end{array}$$

$(\rightarrow I)$ followed by $(\rightarrow E)$

$$\dfrac{\dfrac{\begin{array}{c}[@_a\phi]\\ \vdots\,\pi_1\\ @_a\psi\end{array}}{@_a(\phi\rightarrow\psi)}\qquad\begin{array}{c}\vdots\,\pi_2\\ @_a\phi\end{array}}{@_a\psi} \quad\rightsquigarrow\quad \begin{array}{c}\vdots\,\pi_2\\ @_a\phi\\ \vdots\,\pi_1\\ @_a\psi\end{array}$$

$(@I)$ followed by $(@E)$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi\\ @_a\phi\end{array}}{@_c@_a\phi}}{@_a\phi} \quad\rightsquigarrow\quad \begin{array}{c}\vdots\,\pi\\ @_a\phi\end{array}$$

($\Box I$) followed by ($\Box E$)

$$\dfrac{\dfrac{\dfrac{[@_a\Diamond c]}{\vdots\,\pi_1}}{\dfrac{@_c\phi}{@_a\Box\phi}} \qquad \dfrac{\vdots\,\pi_2}{@_a\Diamond e}}{@_e\phi} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdots\,\pi_2}{@_a\Diamond e}}{\dfrac{\vdots\,\pi_1[e/c]}{@_e\phi}}$$

($\downarrow I$) followed by ($\downarrow E$)

$$\dfrac{\dfrac{\dfrac{[@_a c]}{\vdots\,\pi_1}}{\dfrac{@_c\phi[c/b]}{@_a\downarrow b\phi}} \qquad \dfrac{\vdots\,\pi_2}{@_a e}}{@_e\phi[e/b]} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdots\,\pi_2}{@_a e}}{\dfrac{\vdots\,\pi_1[e/c]}{@_e\phi[e/b]}}$$

($\forall I$) followed by ($\forall E$)

$$\dfrac{\dfrac{\dfrac{\vdots\,\pi}{@_a\phi[c/b]}}{@_a\forall b\phi}}{@_a\phi[e/b]} \quad \rightsquigarrow \quad \dfrac{\vdots\,\pi[e/c]}{@_a\phi[e/b]}$$

We also need reduction rules in connection with the (R_θ) derivation rules corresponding to geometric theories. A *permutable formula* in a derivation is a formula occurrence that is both the conclusion of a (R_θ) rule and the major premise of an elimination rule. Permutable formulas in a derivation can be removed by applying *permutative reductions*. The rule for permutative reductions is as follows in the case where the elimination rule has two premises.

$$\dfrac{\dfrac{\vdots\,\tau_1 \quad \vdots\,\tau_n}{s_1 \quad \dots \quad s_n} \qquad \dfrac{\dfrac{[s_{11}]\dots[s_{1n_1}]}{\vdots\,\pi_1}}{\phi} \qquad \dots \qquad \dfrac{\dfrac{[s_{m1}]\dots[s_{mn_m}]}{\vdots\,\pi_m}}{\phi} \qquad \dfrac{\vdots\,\pi}{\theta}}{\dfrac{\phi}{\psi}}$$

$$\rightsquigarrow$$

$$\dfrac{\vdots\,\tau_1 \quad \vdots\,\tau_n}{s_1 \quad \dots \quad s_n} \qquad \dfrac{\dfrac{[s_{11}[\overline{b}/\overline{c}]]\dots[s_{1n_1}[\overline{b}/\overline{c}]]}{\vdots\,\pi_1[\overline{b}/\overline{c}]} \qquad \dfrac{\vdots\,\pi}{\theta}}{\dfrac{\phi}{\psi}} \qquad \dots \qquad \dfrac{\dfrac{[s_{m1}[\overline{b}/\overline{c}]]\dots[s_{mn_m}[\overline{b}/\overline{c}]]}{\vdots\,\pi_m[\overline{b}/\overline{c}]} \qquad \dfrac{\vdots\,\pi}{\theta}}{\dfrac{\phi}{\psi}}}{\psi}$$

The nominals in the list $\overline{b}$ are pairwise distinct and new. Note that according to the side-condition on the rules (R_θ), cf. Fig. 1.4, none of the nominals in $\overline{c}$ occur free in ϕ, hence, it is ensured that the formula $\phi[\overline{b}/\overline{c}]$ is identical to ϕ. This remark also applies to the undischarged assumptions in the derivations of ϕ. The case where the elimination rule has only one premise is obtained by deleting all instances of the derivation π from the reduction rule above.

A derivation is *normal* if it contains no maximum or permutable formula. The natural deduction system satisfies a normalisation theorem.

Theorem 1.13 (Normalisation). *Any derivation in* $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$ *can be rewritten to a normal derivation by repeated applications of proper and permutative reductions.*

Proof. The proof is somewhat technically involved and will not be given here. We only make a few brief remarks, see the book Braüner (2011) for a detailed proof.

In the case of ordinary classical first-order logic, it is always possible to select reductions such that applying a reduction to a maximum formula only generates new maximum formulas having a lower degree than the original one. The technique used in the standard normalisation proof for first-order logic (originally given in Prawitz (1965)) is based on this property. The natural deduction system considered here does not have this property since the reduction rule for $\Box$ might generate new maximum formulas of the form $@_a\Diamond e$, that is, maximum formulas that do not necessarily have a lower degree than the original one (here we ignore permutable formulas). Essentially, this is a consequence of the fact that the hybrid-logical introduction rule for $\Box$ not only exhibits $\Box$ in the conclusion, but also in the discharged assumptions, namely in the formula $@_a\Diamond c$ displayed in the introduction rule (recall that $@_a\Diamond c$ is an abbreviation for the formula $@_a\neg\Box\neg c$).[17] See Fig. 1.1 of Sect. 1.4.2. Thus, the standard technique for proving normalisation does not work directly here. In the paper Braüner (2004a) and the book Braüner (2011) this problem is solved by using what we have called the $\Box$-graph of a derivation to systematically control the application of reductions to new maximum formulas like those of the form $@_a\Diamond e$ mentioned above.[18] ■

[17] According to Prawitz' terminology (Prawitz 1978), a natural deduction introduction rule for a connective is *explicit* if the connective in question is exhibited exactly once, namely in the conclusion of the rule. Thus, according to this terminology, the hybrid-logical introduction rule for the modal operator is not explicit.

[18] This is actually a generally occurring problem (with a generally applicable solution) since the same problem crops up (and is solved in the same way) in connection with normalisation for intuitionistic hybrid logic, cf. Braüner and de Paiva (2006), and normalisation for first-order hybrid logic, cf. Braüner (2005). See also Braüner (2011). In the first case the reduction rule for the modal operator $\Diamond$, which in intuitionistic hybrid logic is primitive, not defined, might generate new maximum formulas on the form $@_a\Diamond e$, and in the second case, the reduction rule for the quantifier $\forall$ might generate new maximum formulas on the form $@_a E(t)$ where $E(t)$, called the existence predicate, is an abbreviation for $\exists y(y = t)$ which in turn is an abbreviation for $\neg\forall y\neg(y = t)$.

Note that our notion of normalisation involves permutative reductions which is unusual for a classical natural deduction system. Intuitionistic systems, on the other hand, generally involve permutative reductions in connection with derivation rules for the connectives $\bot$, $\vee$, and $\exists$.

1.4.7 The Form of Normal Derivations

Below we shall adapt an important definition from Prawitz (1965) to hybrid logic.

Definition 1.14. A *branch* in a derivation π is a non-empty list $\phi_1, \ldots, \phi_n$ of formula occurrences in π with the following properties.

1. For each $i < n$, ϕ_i stands immediately above ϕ_{i+1}.
2. ϕ_1 is an assumption, or a relational conclusion, or the conclusion of a (R_θ) rule with zero non-relational premises.
3. ϕ_n is either the end-formula of π or a minor or relational premise.
4. For each $i < n$, ϕ_i is not a minor or relational premise.

In the theorem below we make use of the convention that a formula $@_a\phi$ is a *quasi-subformula* of a formula $@_c\psi$ if and only if ϕ is a subformula of ψ in the standard sense. Now comes the theorem which says that normal derivations satisfy a version of the subformula property.

Theorem 1.15 (Quasi-subformula property). *Let Γ be a set of satisfaction statements and let π be a normal derivation of ϕ from Γ in $\mathbf{N}_{\mathcal{H}(\mathcal{O})} + \mathbf{T}$. Moreover, let θ be a formula occurrence in π such that*

1. *θ is not an assumption discharged by an instance of the rule $(\bot 1)$ where the discharged assumption is the major premise of an instance of $(\rightarrow E)$;*
2. *θ is not an occurrence of $@_a\bot$ in a branch whose first formula is an assumption discharged by an instance of the rule $(\bot 1)$ where the discharged assumption is the major premise of an instance of $(\rightarrow E)$; and*
3. *θ is not an occurrence of $@_a\bot$ in a branch whose first formula is the conclusion of a (R_θ) rule with zero non-relational premises.*

Then θ is a quasi-subformula of ϕ, or of some formula in Γ, or of some relational premise, or of some relational conclusion, or of some relationally discharged assumption.

Proof. The proof is along the lines of a similar proof for ordinary classical first-order logic given in Prawitz (1965). The proof will not be given here, we only make a couple of brief remarks, see the book Braüner (2011) for a detailed proof.

The proof is by induction on the *order* of a branch in π which is the number of formula occurrences in π which stand below the last formula occurrence of the branch. The proof makes use of a lemma which says that a branch in a normal derivation can be split into three parts: An analytical part in which formulas are

broken down into their components by successive applications of the elimination rules, a minimum part in which an instance of the rule $(\bot 1)$ may occur, and a synthetical part in which formulas are put together by successive applications of the introduction rules. It follows from the lemma that any formula occurrence in a branch $\phi_1, \ldots, \phi_n$ is a quasi-subformula of either ϕ_1 or ϕ_n. The lemma is more technically involved than the corresponding result in Prawitz (1965), the reason being the disturbing effect of $(Nom1)$, $(\bot 2)$, and the (R_θ) rules. ■

The first two exceptions in the theorem above are inherited from the standard natural deduction system for classical logic, see Prawitz (1965), whereas the third is related to the possibility of having a (R_θ) rule with zero non-relational premises. Remark: If the formula occurrence θ is not covered by one of the three exceptions, then it is a quasi-subformula of ϕ, or of some formula in Γ, or of a formula of the form $@_a c$ or $@_a \Diamond c$ (since relational premises, relational conclusions, and relationally discharged assumptions are of the form $@_a c$ or $@_a \Diamond c$). Note that the formulation of the theorem involves the notion of a branch in what appears to be an indispensable way.

1.4.8 Discussion

The natural deduction systems given in the present section share several features with the tableau systems given in the next section, for example the feature that all formulas in derivations are satisfaction statements. This feature is also shared by the hybrid-logical tableau and Gentzen systems given by Patrick Blackburn in (2000a). A difference between the natural deduction systems of the present section and the systems of Blackburn (2000a) is that we consider additional derivation rules corresponding to first-order conditions expressed by geometric theories whereas Blackburn (2000a) considers tableau systems extended with axioms being pure hybrid-logical formulas, that is, formulas that contain no ordinary propositional symbols (thus, the only propositional symbols in such formulas are nominals).

The use of geometric theories in the context of proof-theory traces back to Alex Simpson's PhD thesis (1994) where it was pointed out that formulas in basic geometric theories correspond to simple natural deduction rules for intuitionistic modal logic. First-order conditions expressed by geometric theories cover a very wide class of logics. This is for example witnessed by the fact that any *Geach axiom schema*, that is, modal-logical axiom schema of the form

$$\Diamond^k \Box^m \phi \to \Box^l \Diamond^n \phi$$

where $\Box^j$ (respectively $\Diamond^j$) is an abbreviation for a sequence of j occurrences of $\Box$ (respectively $\Diamond$), corresponds to a formula of the form required in a basic geometric theory. To be precise, such a Geach axiom schema corresponds to the first-order formula

$$\forall a \forall b \forall c((R^k(a,b) \wedge R^l(a,c)) \to \exists d(R^m(b,d) \wedge R^n(c,d)))$$

where $R^0(a,b)$ means $a = b$ and $R^{j+1}(a,b)$ means $\exists e(R(a,e) \wedge R^j(e,b))$. The displayed formula is then equivalent to a formula of the form required in a basic geometric theory, cf. Simpson (1994); Basin et al. (1997). An example of a Geach axiom schema is the axiom schema obtained by taking each of the numbers k, m, l, and n to be one. The corresponding first-order condition is called *directedness* (or *Church-Rosser*), see Sect. 1.4.3 for the natural deduction rule corresponding to this condition. It is notable that this property is not definable in terms of pure formulas involving just nominals and satisfaction operators (cf. Areces and ten Cate 2007, p. 843).

In the natural deduction system considered in Simpson (1994), a distinction is made between the language of ordinary modal logic and a meta-language involving atomic first-order formulas of the form $R(a,c)$ together with formulas of the form $@_a\phi$, where ϕ is a formula of ordinary modal logic. A natural deduction system for classical modal logic which is similar to the system of Simpson (1994) has been given in Basin et al. (1997), see Sect. 1.6.

The feature of our natural deduction and Gentzen systems that all formulas in derivations are satisfaction statements is at a general level in line with the fundamental idea of Melvin Fitting's prefixed tableau systems (Fitting 1983) and Dov Gabbay's labelled deductive systems (Gabbay 1996) which is to prefix formulas in derivations by meta-linguistic indexes, or labels, with the aim of regulating the proof process. The fundamental idea of Gabbay's labelled deductive systems is to prefix formulas in derivations by labels with the aim of regulating the proof process. In fact, labelled deductive systems are proposed as a systematic way of giving proof systems to many different logics. Note that the work of Simpson (1994) fits naturally into this framework. It should also be mentioned that labelled deductive systems are the basis for the natural deduction systems for substructural logics given in Broda et al. (1999). The crucial difference between the work of Fitting (1983), Gabbay (1996), Simpson (1994), Broda et al. (1999) and our work is that the indexes, or labels, used in the mentioned work belong to a meta-language whereas in our systems they are part of the object language, namely the language of hybrid logic.[19] Thus, in the terminology of Blackburn (2000a), the meta-language has been internalized in the object language. We shall return to this issue in more detail in Sect. 1.6.

[19]Labelled systems have the labelling machinery at the metalevel, whereas hybrid-logical systems have machinery with similar effect at the object level. A third option is chosen in Fitting's paper (1972b) where a curious modal-logical axiom system is given in which labelling machinery is incorporated directly into the object language itself. In that system sequences of formulas of ordinary modal logic, delimited by a distinguished symbol $*$, are used as names for possible worlds. To be more specific, a sequence $*\Diamond\phi_1,\ldots,\Diamond\phi_n,\Diamond\phi_{n+1}*$ is used as the name of a world accessible from the world named by $*\Diamond\phi_1,\ldots,\Diamond\phi_n*$ and in which the formula ϕ_{n+1} is true, if there is one. It is allowed to form object language formulas by prefixing ordinary modal-logical formulas with such sequences. Intuitively, a prefixed formula $*\Diamond\phi_1,\ldots,\Diamond\phi_n*\psi$ says that the formula ψ is true at the world named by the prefix. Prefixed formulas can be combined using the usual connectives of classical logic.

Jerry Seligman's paper (1997) should also be mentioned here: This paper gives a natural deduction system for a logic of situations similar to hybrid logic; the system in question is, however, quite different from ours, for a comparison see the paper Braüner (2004b) or the book Braüner (2011).

1.5 Tableaus and Decision Procedures for Hybrid Logic

Based on tableau systems, we in this section give a decision procedure for a strong hybrid logic including the universal modality. This decision procedure was originally published in the paper Bolander and Braüner (2006). Moreover, we show how the decision procedure can be modified such that simpler tableau-based decision procedures (that is, without loop-checks) are obtained for a weaker hybrid logic where the universal modality is not included. This is inspired by a tableau-based decision procedure given in the paper by Thomas Bolander and Patrick Blackburn (2007). More details can be found in the book Braüner (2011).

1.5.1 The Basics of Tableau Systems

Before giving our hybrid-logical tableau systems, we shall sketch the basics of tableau systems.

A number of persons have played a role in the invention of tableau systems, a leading figure being Jaako Hintikka (see Hintikka 1955). A milestone in the later development of tableau systems is Fitting's book (1983). See the handbook D'Agostino et al. (1999) for further details. Hintikka made the following remarks on the idea behind tableau systems, namely to mimic the recursive truth-conditions in the semantics, whereby a formula is broken down into its components.

> ... the typical situation is one in which we are confronted by a complex formula (or sentence) the truth or falsity of which we are trying to establish by inquiring into its components. Here the rules of truth operate from the complex to the simple: they serve to tell us what, under the supposition that a given complex formula or sentence is true, can be said about the truth-values of its components (Hintikka 1955, p. 20).

The method of reasoning in tableau systems is called "backwards" reasoning: Starting with a particular formula whose validity you want to prove, a tableau is built step by step using the rules, whereby more and more information about counter-models for the formula is obtained, and if at some stage it can be concluded that there cannot be such models, it has been proved that the formula in question is valid. This is contrary to natural deduction systems which are forward reasoning systems since you start with natural deduction rules and try to build a derivation of the formula you have in mind, cf. Sect. 1.4.1.

A *tableau* is a well-founded tree in which each node is labelled with a formula, and the edges represent applications of tableau rules. Where it is appropriate, we shall blur the distinction between a formula and an occurrence of the formula in a tableau. By applying rules to a tableau, the tableau is expanded, that is, new edges and formulas are added to the leaves. A tableau is displayed such that it grows downwards. Technically, premises and conclusions of tableau rules are finite sets of formulas, and a tableau rule has one premise, and one or more conclusions. Most often the premise contains zero, one, or two formulas whereas a conclusion most often contains one or two formulas. A requirement for applying a rule to a branch in a tableau is that all the formulas in the premise are present at the branch, and the result of applying the rule is that for each conclusion of the rule, the end of the branch is extended with a path containing a node for each of the formulas in the conclusion in question. Thus, if for example the rule has two conclusions, then the result of applying the rule is that the end of the branch is extended with two paths, one path for each conclusion. If the rule only has one conclusion, no splitting takes place. A branch in a tableau is called *open* if for no formula χ occurring on the branch, it is the case that $\neg\chi$ also occurs on the branch. A branch is called *closed* if it is not open. A tableau is called *closed* if all branches are closed.

Tableau rules are read from top to bottom, and given an appropriate notion of a model, the intuition behind tableau rules is that the rules step by step attempt to define a model for the root formula of a tableau. This intuition presupposes that tableau rules are sound in the sense that the rules preserve the existence of models, to be more precise, if the premise of a rule has a model (all formulas in the premise are true), then this model is a model for at least one conclusion of the rule (all formulas in the conclusion in question are true). It follows that if the root formula of a tableau has a model, then there is at least one branch in the tableau such that the model for the root formula is a model for all the formulas on the branch, and hence, information about the model for the root formula can be read off from the branch. On the other hand, such a branch obviously has to be open since no formulas χ and $\neg\chi$ can both be true in the same model, so if the tableau does not have any open branches, that is, all its branches are closed, then it can be concluded that the root formula does not have a model. Thus, if a tableau with only closed branches can be constructed having a formula $\neg\phi$ as the root formula, then it has been proved that the formula ϕ is valid.

1.5.2 A Tableau System Including the Universal Modality

A central issue throughout this section is the very expressive hybrid logic obtained by extending the hybrid logic $\mathcal{H}$ in Sect. 1.1.2 with the universal modality E (which is dual to the modality A considered in Sect. 1.2). Formally, the notion of a model is kept as it is in Sect. 1.1.2, but the definition of the relation $\mathfrak{M}, g, w \models \phi$ is extended with the clause

$$\mathfrak{M}, g, w \models E\phi \text{ iff for some } v \in W, \mathfrak{M}, g, v \models \phi$$

$$\frac{@_a\neg\phi}{\neg@_a\phi}\,(\neg) \qquad \frac{\neg@_a\neg\phi}{@_a\phi}\,(\neg\neg)$$

$$\frac{@_a(\phi\wedge\psi)}{@_a\phi,@_a\psi}\,(\wedge) \qquad \frac{\neg@_a(\phi\wedge\psi)}{\neg@_a\phi \mid \neg@_a\psi}\,(\neg\wedge)$$

$$\frac{@_c@_a\phi}{@_a\phi}\,(@) \qquad \frac{\neg@_c@_a\phi}{\neg@_a\phi}\,(\neg@)$$

$$\frac{@_a\Diamond\phi}{@_c\phi,@_a\Diamond c}\,(\Diamond)^{a,b} \qquad \frac{\neg@_a\Diamond\phi,@_a\Diamond e}{\neg@_e\phi}\,(\neg\Diamond)$$

$$\frac{@_aE\phi}{@_c\phi}\,(E)^{a} \qquad \frac{\neg@_aE\phi}{\neg@_e\phi}\,(\neg E)^{c}$$

[a] The nominal c is new
[b] The formula ϕ is not a nominal
[c] The nominal e is on the branch

Fig. 1.6 Tableau rules for connectives

$$\frac{}{@_aa}\,(Ref)^{a} \qquad \frac{@_ac,@_a\phi}{@_c\phi}\,(Nom1)^{b} \qquad \frac{@_ac,@_a\Diamond b}{@_c\Diamond b}\,(Nom2)$$

[a] The nominal a is on the branch
[b] The formula ϕ is a propositional symbol (ordinary or a nominal)

Fig. 1.7 Tableau rules for nominals

where $\mathfrak{M}=(W,R,\{V_w\}_{w\in W})$ is a model, g is an assignment, and w is an element of W. The hybrid logic $\mathcal{H}$ extended with the universal modality will be denoted $\mathcal{H}(E)$. In the present section we define the dual operator A of E by the convention that $A\phi$ is an abbreviation for $\neg E\neg\phi$, thus, E is primitive and A is defined (note that it is opposite in Sect. 1.2). Moreover, in this section we take the connectives $\neg$ and $\Diamond$ to be primitive and $\rightarrow$, $\bot$, and $\Box$ to be defined. It is well-known that the hybrid logic $\mathcal{H}(E)$ is decidable, see Areces et al. (2001a), but decision procedures for this logic are usually not based on tableau or Gentzen systems. In what follows, we shall give a decision procedure for $\mathcal{H}(E)$ based on a tableau system. An essential feature of our decision procedure is that it makes use of a technique called loop-checks.

The rules for the hybrid-logical tableau system are given in Figs. 1.6 and 1.7. The tableau system will be denoted $\mathbf{T}_{\mathcal{H}(E)}$. All formulas in the rules are satisfaction statements or negated satisfaction statements, hence, each node in a tableau is labelled with a satisfaction statement or the negation of a satisfaction statement. Note that since we have taken the connectives $\rightarrow$, $\bot$, $\Box$, and A to be defined, not primitive, they do not need separate rules. It is straightforward to check that the rules

of $\mathbf{T}_{\mathcal{H}(E)}$ are sound in the sense that if the premise of a rule has a model (strictly speaking together with an assignment), then this model, possibly with modified references to new nominals, is a model for at least one conclusion of the rule.

We shall make use of the following conventions about the tableau rules. The rules $(\neg)$, $(\neg\neg)$, $(\wedge)$, $(\neg\wedge)$, $(@)$, $(\neg@)$, $(\Diamond)$, and (E) will be called *destructive* rules and the remaining rules will be called *non-destructive*. The reason why we call the mentioned rules destructive is that in the systematic tableau construction algorithm we define later in this section, application of destructive rules is restricted such that a destructive rule is applied at most once to a formula (a destructive rule has exactly one formula in the premise).[20] The destructive rules $(\Diamond)$ and (E) will also be called *existential* since they introduce new nominals. Note that non-destructive rules are only applicable to formulas in the forms $@_a p$, $@_a c$, $@_a \Diamond c$, $\neg@_a \Diamond \phi$, and $\neg@_a E\phi$, and conversely, destructive rules are only applicable to formulas not in these forms (in fact, exactly one destructive rule is applicable to any formula which is not in one of these forms). So, the classification of rules as destructive and non-destructive corresponds to a classification of formulas according to their form.

In what follows we shall give a decision procedure $\mathcal{H}(E)$ which works as follows: Given a formula $@_a\phi$ whose validity we have to decide, a systematic tableau construction algorithm constructs a finite tableau having the formula $\neg@_a\phi$ as the root formula. If the tableau has an open branch, then a model for $\neg@_a\phi$ can be defined.[21] Thus, in this case the formula $@_a\phi$ is not valid. On the other hand, in the case where there are no open branches in the tableau, it follows from soundness of the tableau rules that $\neg@_a\phi$ does not have a model, hence $@_a\phi$ is valid.

[20] This terminology is used in a somewhat different sense than is common: Our destructive rules preserve information in the sense that if a conclusion of a destructive rule has a model, then this model is a model for the premise of the rule as well, that is, no models are included (note that this is opposite of soundness which says that no models are excluded). In the usual sense destructive rules are rules that do not preserve information (see Fitting 1972a).

[21] An occurrence of a satisfaction statement $@_a\phi$ or the negation of a satisfaction statement $\neg@_a\phi$ in a tableau can be seen as a formula ϕ together with a pair consisting of the representation of a possible world (the nominal a) and the representation of a truth-value (depending on whether the satisfaction statement is negated or not). Note in this connection that in the possible worlds semantics, the semantic value assigned to a formula is a function from possible worlds to truth-values, and set-theoretically, such a function is a set of pairs of possible worlds and truth-values (called the graph of the function). Hence, the pairs of nominals and representations of truth-values associated with formulas in the tableau system can be considered representations of elements of functions constituting semantic values. Thus, the tableau rules step by step build up semantic values of the formulas involved, similar to the way in which the accessibility relation step by step is built up (there is a difference however; the accessibility relation can be *any* relation, but the semantic value of a formula has to be a *function*, that is, a relation where no element of the domain is related to more than one element of the codomain, and this is exactly what is required of an open branch in a tableau, namely that no satisfaction statement is related to more than one truth-value).

1.5.3 Some Properties of the Tableau System

In this subsection we shall prove some properties of the tableau system $\mathbf{T}_{\mathcal{H}(E)}$. Only Theorem 1.16 and Proposition 1.24 are used later in connection with $\mathbf{T}_{\mathcal{H}(E)}$, but we find that the other results of the subsection are of independent interest. Later we give a decision procedure based on a tableau system for the weaker hybrid logic $\mathcal{H}$ that does not include the universal modality, and in this connection we shall make crucial use of Corollary 1.20 and a strengthened version of Theorem 1.22 (as well as Theorem 1.16 and Proposition 1.24 again).

The tableau system $\mathbf{T}_{\mathcal{H}(E)}$ satisfies the following variant of the quasi-subformula property.

Theorem 1.16 (Quasi-subformula property). *If a formula $@_a\phi$ occurs in a tableau where ϕ is not a nominal and ϕ is not of the form $\Diamond b$, then ϕ is a subformula of the root formula. If a formula $\neg@_a\phi$ occurs in a tableau, then ϕ is a subformula of the root formula.*

Proof. A simultaneous induction where each rule is checked. ■

Below we shall give some further results which shows some interesting features of the tableau system. First two definitions.

Definition 1.17. Let Θ be a branch of a tableau and let N^Θ be the set of nominals occurring in the formulas of Θ. Define a binary relation $\sim_\Theta$ on N^Θ by $a \sim_\Theta b$ if and only if the formula $@_a b$ occurs on Θ. Let $\sim_\Theta^*$ be the reflexive, symmetric, and transitive closure of $\sim_\Theta$.

Definition 1.18. An occurrence of a nominal in a formula is *equational* if the occurrence is a formula (that is, if the occurrence is not part of a satisfaction operator).

For example, the occurrence of the nominal c in the formula $\phi \wedge c$ is equational but the occurrence of c in $\psi \wedge @_c\chi$ is not. The justification for this terminology is that a nominal in the first-order correspondence language (and thereby also in the semantics) gives rise to an equality statement if and only if the nominal occurrence in question occurs equationally. Note that a nominal occurs equationally in a formula if and only if the nominal is a subformula of the formula.

Theorem 1.19. *Let $@_a b$ be a formula occurrence on a branch Θ of a tableau. If the nominals a and b are different, then each of the nominals is identical to, or related by $\sim_\Theta$ to, a nominal with an equational occurrence in the root formula.*

Proof. Check each rule. Theorem 1.16 is needed in a number of the cases. In the case with the rule $(\Diamond)$, we make use of the restriction that the rule cannot be applied to formulas of the form $@_a\Diamond\phi$ where ϕ is a nominal. ■

Corollary 1.20. *Let Θ be a branch of a tableau. Any non-singleton equivalence class with respect to the equivalence relation $\sim_\Theta^*$ contains a nominal with a equational occurrence in the root formula.*

Proof. Follows directly from Theorem 1.19. ■

The corollary above says that non-trivial equational reasoning, that is, reasoning involving non-singleton equivalence classes, only takes place in connection with certain nominals in the root formula, namely those that occur equationally. Note that this implies that pure modal input to the tableau only gives rise to reasoning involving singleton equivalence classes.

Definition 1.21. A formula occurrence in a tableau is an *accessibility* formula occurrence if it is an occurrence of the formula $@_a\Diamond c$ generated by the rule $(\Diamond)$.

Note that if the rule $(\Diamond)$ is applied to a formula occurrence $@_a\Diamond\Diamond b$, resulting in the branch being extended with $@_a\Diamond c$ and $@_c\Diamond b$, then the occurrence of $@_a\Diamond c$ is an accessibility formula occurrence, but the occurrence of $@_c\Diamond b$ is not.

Theorem 1.22. *Let $@_a\Diamond b$ be a formula occurrence on a branch Θ of a tableau. Either there is an accessibility formula occurrence $@_{a'}\Diamond b$ on Θ such that $a \sim^*_\Theta a'$ or the formula $\Diamond b$ is a subformula of the root formula.*

Proof. Check each rule. Theorem 1.16 is needed in some of the cases. ■

The only way new nominals can be introduced to a tableau is by using one of the rules $(\Diamond)$ or (E) which we called existential rules. This motivates the following definition.

Definition 1.23. Let Θ be a branch of a tableau. If a new nominal c is introduced by applying an existential rule to a satisfaction statement $@_a\phi$, then we write $a <_\Theta c$.

The definition above gives us a binary relation $<_\Theta$ on the set N^Θ.

Proposition 1.24. *Let Θ be a branch of a tableau. Assume that if an existential rule is applied to a formula occurrence on Θ, then the existential rule is not applied to any other formula occurrence at Θ having the same form. The graph $(N^\Theta, <_\Theta)$ is the disjoint union of a finite set of well-founded and finitely branching trees.*

Proof. That the graph is well-founded follows from the observation that if $a <_\Theta c$, then the first occurrence of a on the branch is before the first occurrence of c. That the graph is the disjoint union of a set of trees follows from well-foundedness together with the observation that if $a <_\Theta c$ and $b <_\Theta c$, then the nominals a and b are identical. That the set of trees is finite follows from the observation that for any nominal c that occurs in the branch, but does not occur in the root formula, there is a nominal a such that $a <_\Theta c$, thus, the nominal c cannot be the root of a tree.

The following argument shows that the trees are finitely branching. Assume conversely that there exists an infinite sequence $a <_\Theta c_1$, $a <_\Theta c_2, \ldots$ of pairwise distinct edges. For each i, the edge $a <_\Theta c_i$ is generated by applying an existential rule to some formula occurrence χ_i. Consider the sequence χ_1, $\chi_2, \ldots$ of formula occurrences. These rule applications are distinct since the nominals c_1, $c_2, \ldots$ are distinct, and by assumption, if an existential rule is applied to a formula occurrence, then the existential rule is not applied to any other formula occurrence having the

same form, so the formula occurrences in the sequence $\chi_1, \chi_2, \ldots$ are occurrences of infinitely many different formulas. Now, if the edge $a <_\Theta c_i$ is generated by applying the existential rule $(\Diamond)$ to χ_i, then χ_i is of the form $@_a \Diamond \phi_i$ where ϕ_i is not a nominal, and hence, $\Diamond \phi_i$ is a subformula of the root formula by Theorem 1.16, and if $a <_\Theta c_i$ is generated by applying the other existential rule (E) to χ_i, then χ_i is of the form $@_a E \phi_i$, and hence, $E\phi_i$ is a subformula of the root formula, again by Theorem 1.16. But there are only finitely many subformulas of the root formula, which contradicts that infinitely many different formulas occur in the sequence $\chi_1, \chi_2, \ldots$. ■

Note that in the above results we have not made any assumptions on which rules are applied on the branch Θ, but if we assume that Θ is closed under the rules (Ref) and $(Nom1)$, then $\sim^*_\Theta$ coincides with $\sim_\Theta$.

1.5.4 Systematic Tableau Construction

In this subsection we give a systematic tableau construction algorithm for $\mathbf{T}_{\mathcal{H}(E)}$. Before giving the algorithm, we need an important definition.

Definition 1.25. Let b and a be nominals occurring on a branch Θ of a tableau. The nominal a is *included* in the nominal b with respect to Θ if the following is the case: For any subformula ϕ of the root formula, if the formula $@_a \phi$ occurs on Θ, then $@_b \phi$ also occurs on Θ, and similarly, if $\neg @_a \phi$ occurs on Θ, then $\neg @_b \phi$ also occurs on Θ. If a is included in b with respect to Θ, and the first occurrence of b on Θ is before the first occurrence of a, then we write $a \subseteq_\Theta b$.

We are now ready to give the systematic tableau construction algorithm.

Definition 1.26 (Tableau construction algorithm). Given a formula $@_a \phi$ of $\mathcal{H}(E)$ whose validity has to be decided, we define by induction a sequence $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$ of finite tableaus in $\mathbf{T}_{\mathcal{H}(E)}$, each of which is embedded in its successor. Let $\mathcal{T}_0$ be the finite tableau constituted by the single formula $\neg @_a \phi$. If possible, apply an arbitrary rule to $\mathcal{T}_n$ with the following three restrictions:

1. If a formula to be added to a branch by applying a rule already occurs on the branch, then the addition of the formula is simply omitted.
2. After the application of a destructive rule to a formula occurrence ϕ on a branch, it is recorded that the rule was applied to ϕ with respect to the branch and the rule will not again be applied to ϕ with respect to the branch or any extension of it.
3. The existential rule $(\Diamond)$ is not applied to a formula occurrence $@_a \Diamond \phi$ on a branch Θ if there exists a nominal b such that $a \subseteq_\Theta b$ (and analogously for the existential rule (E)).

Let $\mathcal{T}_{n+1}$ be the resulting tableau.

Note that due to the first restriction, a formula cannot occur more than once on a branch. Also note that no information is recorded about applications of non-destructive rules. The conditions on applications of the existential rules $(\Diamond)$ and (E) in the third restriction are the loop-check conditions. The intuition behind loop-checks is that an existential rule is not applied in a world if the information in that world can be found already in an ancestor world. Hence, the introduction of a new world by the existential rule is blocked.

Theorem 1.27. *The systematic tableau construction algorithm for* $\mathbf{T}_{\mathcal{H}(E)}$ *terminates in the sense that there exists an n such that* $\mathcal{T}_n = \mathcal{T}_{n+1}$.

Proof. Assume conversely that the algorithm does not terminate. Then the resulting tableau is infinite, and hence, has an infinite branch Θ. The graph $(N^\Theta, <_\Theta)$ is the disjoint union of a finite set of finitely branching trees cf. Proposition 1.24, so it has an infinite branch $a_1 <_\Theta a_2 <_\Theta a_3, \ldots$ (otherwise $\mathcal{N}^\Theta$ would be finite, and hence, by Theorem 1.16 there would only be finitely many formulas occurring on the branch Θ, contradicting that it is infinite). Now, for each i, let Θ_i be the initial segment of Θ up to, but not including, the first formula containing an occurrence of the nominal a_{i+1}. Thus, an existential rule was applied to a formula occurrence on the branch Θ_i resulting in the generation of a_{i+1}. Let Γ_i be the set of formulas which contains any subformula ϕ of the root formula such that $@_{a_i}\phi$ occurs on the branch Θ_i, and similarly, let Δ_i be the set of formulas which contains any subformula ϕ of the root formula such that $\neg@_{a_i}\phi$ occurs on the branch Θ_i. Since there are only finitely many sets of subformulas of the root formula, there exists j and k such that $j < k$ and $\Gamma_j = \Gamma_k$ as well as $\Delta_j = \Delta_k$. Clearly, the first occurrence of a_j on Θ_k is before the first occurrence of a_k. Moreover, for any subformula ϕ of the root formula, if $@_{a_k}\phi$ occurs on Θ_k, then $\phi \in \Gamma_k$, and hence, $\phi \in \Gamma_j$, but then $@_{a_j}\phi$ occurs on Θ_j which is an initial segment of Θ_k. A similar argument shows that if $\neg@_{a_k}\phi$ occurs on Θ_k, then $\neg@_{a_j}\phi$ also occurs on Θ_k. Hence, a_k is included in a_j with respect to Θ_k. We conclude that $a_k \subseteq_{\Theta_k} a_j$. But this contradicts that an existential rule was applied to a formula occurrence on the branch Θ_k resulting in the addition of the first formula containing an occurrence of the nominal a_{k+1}. Thus, the algorithm terminates. ■

We have thus given a systematic tableau construction algorithm which step by step builds up a tableau and which terminates with a tableau having the property that no rules are applicable to it except for applications of rules blocked by the three restrictions in Definition 1.26. It is important to note that except for these three restrictions, the tableau construction algorithm does not make any restrictions on the order in which rules are applied. In this sense the algorithm is non-deterministic.

1.5.5 The Model Existence Theorem and Decidability

In this subsection we give a model existence theorem and we give the decision procedure. The model existence theorem implies that the tableau system $\mathbf{T}_{\mathcal{H}(E)}$ is complete. Throughout the subsection, we shall assume that Θ is a given branch of a tableau generated by the systematic tableau construction algorithm, Definition 1.26. Where no confusion can occur, we shall often omit reference to the branch Θ. First some machinery.

Definition 1.28. Let W be the subset of N^Θ containing any nominal a having the property that there is no nominal b such that $a \subsetneq_\Theta b$. Let $\approx$ be the restriction of $\sim_\Theta$ to W.

Note that W contains all nominals of the root formula since the root formula is the first formula of the branch Θ. Observe that Θ is closed under the rules (Ref) and $(Nom1)$, so the relation $\sim_\Theta$ and hence also the relation $\approx$ are equivalence relations. Given a nominal a in W, we let $[a]_\approx$ denote the equivalence class of a with respect to $\approx$ and we let $W/\approx$ denote the set of equivalence classes.

Definition 1.29. Let R be the binary relation on W defined by aRc if and only if there exists a nominal $c' \approx c$ such that one of the following two conditions is satisfied.

1. The formula $@_a \Diamond c'$ occurs on Θ.
2. There exists a nominal d in N^Θ such that the formula $@_a \Diamond d$ occurs on Θ and $d \subsetneq_\Theta c'$.

Note that the nominal d referred to in the second item in the definition is not an element of W. It follows from Θ being closed under the rule $(Nom2)$ that R is compatible with $\approx$ in the first argument and it is trivial that R is compatible with $\approx$ in the second argument. We let $\overline{R}$ be the binary relation on $W/\approx$ defined by $[a]_\approx \overline{R} [c]_\approx$ if and only if aRc.

Definition 1.30. For any element a of W, let V_a be the function that to each ordinary propositional symbol assigns an element of $\{0,1\}$ such that $V_a(p) = 1$ if $@_a p$ occurs on Θ and $V_a(p) = 0$ otherwise.

It follows from Θ being closed under the rule $(Nom1)$ that V_a is compatible with $\approx$ in the index a, so we let $\overline{V}_{[a]_\approx}$ be defined by $\overline{V}_{[a]_\approx}(p) = V_a(p)$. We are now ready to define a model.

Definition 1.31. Let $\mathfrak{M}$ be the model $(W/\approx, \overline{R}, \{\overline{V}_{[a]_\approx}\}_{[a]_\approx \in W/\approx})$ and let the assignment g for $\mathfrak{M}$ be defined by $g(a) = [a]_\approx$.

The model above is in some respects similar to the model defined in Blackburn (2000a). One crucial difference, however, is that the model above is necessarily finite since the tableau branch Θ is finite.

Theorem 1.32 (Model existence). *Assume that the branch Θ is open. For any satisfaction statement $@_a\phi$ which only contains nominals from W, the following two statements hold.*

- *If $@_a\phi$ occurs on Θ, then it is the case that $\mathfrak{M}, g, [a]_\approx \models \phi$.*
- *If $\neg @_a\phi$ occurs on Θ, then it is not the case that $\mathfrak{M}, g, [a]_\approx \models \phi$.*

Proof. Induction on the structure of ϕ. We only consider the most interesting case, namely where ϕ is of the form $\Diamond\psi$, the remaining cases can be found in the book Braüner (2011).

Assume that $@_a\Diamond\psi$ occurs on the branch Θ. We then have to prove that $\mathfrak{M}, g, [a]_\approx \models \Diamond\psi$, that is, for some equivalence class $[c]_\approx$ such that $[a]_\approx \overline{R}[c]_\approx$, it is the case that $\mathfrak{M}, g, [c]_\approx \models \psi$. We have two cases, according to whether the formula ψ is a nominal or not. We first consider the case where ψ is a nominal, say b. So we just have to prove that $[a]_\approx \overline{R}[b]_\approx$ which trivially follows from the definition of the relation $\overline{R}$. We now consider the case where ψ is not a nominal. By the rule $(\Diamond)$ some formulas $@_a\Diamond c$ and $@_c\psi$ also occur on Θ where the nominal c is new (note that $a \in W$, so the application of the rule is not blocked by a loop-check condition). If $c \in W$, then clearly $[a]_\approx \overline{R}[c]_\approx$ and $\mathfrak{M}, g, [c]_\approx \models \psi$ by induction. If $c \notin W$, then by definition of W there exists a nominal d such that $c \subseteq_\Theta d$. Without loss of generality we assume that there does not exist a nominal e such that $d \subseteq_\Theta e$. But this implies that $d \in W$. Moreover, by Theorem 1.16, the formula ψ is a subformula of the root formula, so $@_d\psi$ occurs on Θ. By induction, $\mathfrak{M}, g, [d]_\approx \models \psi$, and clearly, $[a]_\approx \overline{R}[d]_\approx$.

Assume that $\neg @_a\Diamond\psi$ occurs on the branch Θ. We then have to prove that $\mathfrak{M}, g, [a]_\approx \models \Diamond\psi$ does not hold, that is, for any equivalence class $[c]_\approx$ such that $[a]_\approx \overline{R}[c]_\approx$, it is not the case that $\mathfrak{M}, g, [c]_\approx \models \psi$. From $[a]_\approx \overline{R}[c]_\approx$ it follows that there exists a nominal $c' \approx c$ satisfying one of the two conditions in the definition of the relation R. In the first condition in this definition, the formula $@_a\Diamond c'$ occurs on Θ. Thus, by the rule $(\neg\Diamond)$ the formula $\neg @_{c'}\psi$ occurs on Θ. By induction we conclude that $\mathfrak{M}, g, [c']_\approx \models \psi$ does not hold and trivially, $[c']_\approx = [c]_\approx$. In the second condition in the definition there exists a nominal d in N^Θ such that the formula $@_a\Diamond d$ occurs on Θ and $d \subseteq_\Theta c'$. By the rule $(\neg\Diamond)$ the formula $\neg @_d\psi$ occurs on Θ. But by Theorem 1.16, the formula ψ is a subformula of the root formula, and $d \subseteq_\Theta c'$, so $\neg @_{c'}\psi$ occurs on Θ. By induction we conclude that $\mathfrak{M}, g, [c']_\approx \models \psi$ does not hold and trivially, $[c']_\approx = [c]_\approx$. ∎

We are now finally able to give the decision procedure.

Definition 1.33 (Decision procedure). Given a formula $@_a\phi$ of $\mathcal{H}(E)$ whose validity we have to decide, let $\mathcal{T}_n$ be a terminal tableau generated by the tableau construction algorithm, Definition 1.26. If there are no open branches in the tableau $\mathcal{T}_n$, then the root formula $\neg @_a\phi$ of $\mathcal{T}_n$ does not have a model since the tableau rules are sound, hence, the formula $@_a\phi$ is valid. If the tableau $\mathcal{T}_n$ has an open branch, then it follows from the model existence theorem, Theorem 1.32, that the formula $@_a\phi$ is not valid.

As a spin-off from the decision procedure we get the finite model property.

Theorem 1.34 (Finite model property). *If a formula of $\mathcal{H}(E)$ is satisfiable, then it is satisfiable by a finite model.*

Proof. A straightforward application of the decision procedure, Definition 1.33, together with observation that the model defined in Theorem 1.31 is finite. ■

We shall finish this subsection by making some remarks on complexity issues. Consider a branch Θ of a tableau with root formula ψ. If there are n distinct subformulas of ψ, then there are 2^n distinct sets of subformulas of ψ. It follows from inspection of the termination proof, Theorem 1.27, that the height of a tree in the graph $(N^\Theta, <_\Theta)$, cf. Proposition 1.24, is $O(2^n)$. By inspection of Proposition 1.24, the outdegrees of nodes in the graph $(N^\Theta, <_\Theta)$ are bounded by n (to be more precise, the outdegrees are bounded by the number of distinct subformulas of ψ having the form $\Diamond\phi$ or $E\phi$). It follows that the size of N^Θ is $O(2^{2^n})$. Combining this with the quasi-subformula property, Theorem 1.16, we can calculate that the length of the branch Θ is $O(2^{2^n})$.

It follows that our algorithm solves the satisfiability problem for $\mathcal{H}(E)$ formulas in nondeterministic double exponential time (2-NEXPTIME) in the size of formulas. However, the satisfiability problem for $\mathcal{H}(E)$ formulas is in fact solvable in exponential time (EXPTIME), cf. Areces et al. (2001a), so the algorithm is not optimal from a complexity theoretic point of view. Our aim has been to give a simple and straightforward algorithm, but we believe that by sacrificing some of the simplicity, the algorithm can be optimized by applying the techniques which in the paper Donini and Massacci (2000) are applied to give an optimal EXPTIME tableau-based algorithm for a description logic variant of the modal logic K extended with background theories. The techniques of the paper involve caching of unsatisfiability results for already explored tableau branches. However, according to the handbook chapter (Horrocks et al. 2007, p. 220), the optimal EXPTIME algorithm for K with background theories given in the paper Donini and Massacci (2000) has never been implemented, whereas the handbook chapter describes a simple 2-NEXPTIME tableau-based algorithm for the same logic which, again according to the handbook chapter, has proven to work surprisingly well in practice.

1.5.6 Tableau Examples

As a first example, consider the formula $@_a\neg\Diamond(a \wedge \neg\Diamond a)$ which is valid (this is straightforward to see by considering the equivalent formula $@_a\Box(a \rightarrow \Diamond a)$). Given this formula as input, a possible tableau generated by the tableau construction algorithm is the tableau below (recall that the algorithm is non-deterministic).

$\neg @_a \neg \Diamond (a \wedge \neg \Diamond a)$	1.
$@_a \Diamond (a \wedge \neg \Diamond a)$	2. by $(\neg\neg)$ rule on 1
$@_c (a \wedge \neg \Diamond a)$	3. by $(\Diamond)$ rule on 2
$@_a \Diamond c$	4. by $(\Diamond)$ rule on 2
$@_c a$	5. by $(\wedge)$ rule on 3
$@_c \neg \Diamond a$	6. by $(\wedge)$ rule on 3
$\neg @_c \Diamond a$	7. by $(\neg)$ rule on 6
$@_c c$	8. by (Ref) rule
$@_a c$	9. by $(Nom1)$ rule on 5 and 8
$@_a a$	10. by (Ref) rule
$@_c \Diamond c$	11. by $(Nom2)$ rule on 9 and 4
$\neg @_c a$	12. by $(\neg\Diamond)$ rule on 7 and 11

The enumeration of the formulas and the notation in the right-hand-side column is not a formal part of the tableau, but has been added to describe how the tableau was constructed. The tableau above only has one branch and that branch is closed since it contains the formula $@_c a$ as well as $\neg @_c a$ (in lines 5 and 12). It follows from the tableau rules being sound that the formula $@_a \neg \Diamond (a \wedge \neg \Diamond a)$ is valid.

As a second example, consider the formula $@_a \neg \Diamond (a \wedge r)$ which is not valid. Given this formula as input, a possible tableau generated by the tableau construction algorithm is the tableau below.

$\neg @_a \neg \Diamond (a \wedge r)$	1.
$@_a \Diamond (a \wedge r)$	2. by $(\neg\neg)$ rule on 1
$@_c (a \wedge r)$	3. by $(\Diamond)$ rule on 2
$@_a \Diamond c$	4. by $(\Diamond)$ rule on 2
$@_c a$	5. by $(\wedge)$ rule on 3
$@_c r$	6. by $(\wedge)$ rule on 3
$@_a r$	7. by $(Nom1)$ rule on 5 and 6
$@_c c$	8. by (Ref) rule
$@_a c$	9. by $(Nom1)$ rule on 5 and 8
$@_c \Diamond c$	10. by $(Nom2)$ rule on 9 and 4
$@_a a$	11. by (Ref) rule

Note that the tableau above only has one branch and that branch is open. It follows from the model existence theorem, Theorem 1.32, that the formula $@_a \neg \Diamond (a \wedge r)$ is not valid, and Definition 1.31 gives a counter-model, namely a model having one world, the set $\{a, c\}$, which is an equivalence class as the formulas $@_c a, @_c c, @_a c, @_a a$ are on the branch. The propositional symbol r is true at the world as the formulas $@_c r, @_a r$ are on the branch and the world is related to itself by the accessibility relation as $@_a \Diamond c, @_c \Diamond c$ are on the branch.

Now, loop-checks were not needed to ensure termination in the two tableau examples above. Below we shall consider a third and a fourth tableau example

where loop-checks are actually needed, namely an example involving the universal modal operator E and an example involving the standard modal operator $\Diamond$. In the example involving E there is no non-trivial equational reasoning, that is, there is no reasoning with formulas like $@_a c$ where the nominals a and c are distinct, however, the example involving $\Diamond$ does make use of such reasoning, which makes it the most complicated of the two examples.

In the third example (the example involving E) we consider the formula $@_b \neg(r \wedge \neg E \neg E r)$ which is not valid. A possible tableau generated by the tableau construction algorithm is the tableau below.

$\neg @_b \neg(r \wedge \neg E \neg E r)$	1.
$@_b(r \wedge \neg E \neg E r)$	2. by $(\neg\neg)$ rule on 1
$@_b r$	3. by $(\wedge)$ rule on 2
$@_b \neg E \neg E r$	4. by $(\wedge)$ rule on 2
$\neg @_b E \neg E r$	5. by $(\neg)$ rule on 4
$@_b b$	6. by (Ref) rule
$\neg @_b \neg E r$	7. by $(\neg E)$ rule on 5
$@_b E r$	8. by $(\neg\neg)$ rule on 7
$@_a r$	9. by (E) rule on 8
$@_a a$	10. by (Ref) rule
$\neg @_a \neg E r$	11. by $(\neg E)$ rule on 5
$@_a E r$	12. by $(\neg\neg)$ rule on 11

The counter-model to $@_b \neg(r \wedge \neg E \neg E r)$ given by Definition 1.31 has two worlds, the equivalence classes $\{b\}$ and $\{a\}$, where the propositional symbol r is true at both. In the tableau above, note that the nominal a is included in the nominal b with respect the branch, cf. Definition 1.25, the reason being that the two sets of subformulas of the root formula which are respectively prefixed by $@_a$ and $\neg @_a$, are the sets $\{r, E r\}$ and $\{\neg E r\}$, whereas the two sets of subformulas of the root formula which are respectively prefixed by $@_b$ and $\neg @_b$, are the sets $\{r \wedge \neg E \neg E r, r, \neg E \neg E r, E r\}$ and $\{\neg(r \wedge \neg E \neg E r), E \neg E r, \neg E r\}$. Of course, the important observations are that

$$\{r, E r\} \subseteq \{r \wedge \neg E \neg E r, r, \neg E \neg E r, E r\}$$

and

$$\{\neg E r\} \subseteq \{\neg(r \wedge \neg E \neg E r), E \neg E r, \neg E r\}.$$

Thus, application of the rule (E) to the occurrence of $@_a E r$ in line 12 is blocked by the loop-check condition in the tableau construction algorithm, that is, the third restriction in Definition 1.26. If the loop-check condition is removed, the tableau construction algorithm can continue in "cycles" as follows.

$@_c r$	11. (E) rule on 10
$\neg @_c \neg E r$	12. $(\neg E)$ rule on 5
$@_c E r$	13. $(\neg\neg)$ rule on 12
$@_d r$	14. (E) rule on 12
$\neg @_d \neg E r$	15. $(\neg E)$ rule on 5
$@_d E r$	16. $(\neg\neg)$ rule on 15

$\vdots$

Of course, the dashed lines that separate the cycles are not a formal part of the tableau. Note that the second cycle, lines 14–16, is identical to the first cycle, lines 11–13, except that the nominal d occurs in the second cycle where the nominal c occurs in the first cycle. What happens is that when a new nominal has been generated, say the nominal c above, the formula $\neg @_b E \neg E r$ in line 5 produces a formula $\neg @_c \neg E r$, which in turn produces $@_c E r$, and this formula generates yet another new nominal, and so on.

In the fourth example (the example involving $\Diamond$) we consider the formula $@_b \neg((b \wedge r) \wedge (\Diamond b \wedge \neg\Diamond\neg\Diamond(b \wedge r)))$ which is not valid. A possible tableau generated by the tableau construction algorithm is the tableau below.

$\neg @_b \neg((b \wedge r) \wedge (\Diamond b \wedge \neg\Diamond\neg\Diamond(b \wedge r)))$	1.
$@_b((b \wedge r) \wedge (\Diamond b \wedge \neg\Diamond\neg\Diamond(b \wedge r)))$	2. by $(\neg\neg)$ rule on 1
$@_b(b \wedge r)$	3. by $(\wedge)$ rule on 2
$@_b(\Diamond b \wedge \neg\Diamond\neg\Diamond(b \wedge r))$	4. by $(\wedge)$ rule on 2
$@_b b$	5. by $(\wedge)$ rule on 3
$@_b r$	6. by $(\wedge)$ rule on 3
$@_b \Diamond b$	7. by $(\wedge)$ rule on 4
$@_b \neg\Diamond\neg\Diamond(b \wedge r)$	8. by $(\wedge)$ rule on 4
$\neg @_b \Diamond \neg\Diamond(b \wedge r)$	9. by $(\neg)$ rule on 8
$\neg @_b \neg\Diamond(b \wedge r)$	10. by $(\neg\Diamond)$ rule on 9 and 7
$@_b \Diamond(b \wedge r)$	11. by $(\neg\neg)$ rule on 10
$@_a(b \wedge r)$	12. by $(\Diamond)$ rule on 11
$@_b \Diamond a$	13. by $(\Diamond)$ rule on 11
$@_a b$	14. by $(\wedge)$ rule on 12
$@_a r$	15. by $(\wedge)$ rule on 12
$@_a a$	16. by (Ref) rule
$@_b a$	17. by $(Nom1)$ rule on 14 and 16
$\neg @_a \neg\Diamond(b \wedge r)$	18. by $(\neg\Diamond)$ rule on 9 and 13
$@_a \Diamond(b \wedge r)$	19. by $(\neg\neg)$ rule on 18

The counter-model given by Definition 1.31 has one world, the equivalence class $\{b,a\}$, such that the propositional symbol r is true at the world and such that the world is related to itself by the accessibility relation. In the tableau above, note that the nominal a is included in the nominal b with respect the branch, hence, application of the rule $(\Diamond)$ to the occurrence of $@_a\Diamond(b \wedge r)$ in line 19 is blocked by the loop-check condition. Like in the previous example, if the loop-check condition is removed, the tableau construction algorithm can continue in cycles.

$@_c(b \wedge r)$	20. by $(\Diamond)$ rule on 19
$@_a\Diamond c$	21. by $(\Diamond)$ rule on 19
$@_c b$	22. by $(\wedge)$ rule on 20
$@_c r$	23. by $(\wedge)$ rule on 20
$@_b\Diamond c$	24. by $(Nom2)$ rule on 14 and 21
$\neg@_c\neg\Diamond(b \wedge r)$	25. by $(\neg\Diamond)$ rule on 9 and 24
$@_c\Diamond(b \wedge r)$	26. by $(\neg\neg)$ rule on 25
$@_d(b \wedge r)$	27. by $(\Diamond)$ rule on 26
$@_c\Diamond d$	28. by $(\Diamond)$ rule on 26
$@_d b$	29. by $(\wedge)$ rule on 27
$@_d r$	30. by $(\wedge)$ rule on 27
$@_b\Diamond d$	31. by $(Nom2)$ rule on 22 and 26
$\neg@_d\neg\Diamond(b \wedge r)$	32. by $(\neg\Diamond)$ rule on 9 and 31
$@_d\Diamond(b \wedge r)$	33. by $(\neg\neg)$ rule on 32

$\vdots$

Here the second cycle, lines 27–33, is identical to the first cycle, lines 20–26, except that the nominals d and c occur in the second cycle where the nominals c and a occur in the first cycle.

Note the use of the rule $(Nom2)$ in the tableau example above. Without this rule loop-checks are not needed to ensure termination of the tableau example. This is actually the case for any input to the tableau construction algorithm which only involves the standard modal operator $\Diamond$. This is a consequence of a result in the following section, namely Theorem 1.37, which concerns a tableau system without the rule $(Nom2)$. It is in this connection an interesting observation that the rule $(Nom2)$ can only be used in the presence of non-trivial equational reasoning, the reason being that an application of $(Nom2)$ can only generate a new formula if the nominals a and c in the premise $@_a c$ are distinct (see Fig. 1.7). This implies that any tableau example where loop-checks are actually needed to ensure termination, and where only the modal operator $\Diamond$ is involved, must make use of non-trivial equational reasoning. This in turn implies that in any tableau example

where loop-checks are actually needed to ensure termination, and where only the modal operator $\Diamond$ is involved, the root formula must contain equationally occurring nominals, cf. Corollary 1.20, like the nominal b in the example above.

If we only consider formulas involving the standard modal operator $\Diamond$, not the universal modal operator E, then it is tempting to ask whether we cannot simply omit the rule $(Nom2)$ and thereby obtain a tableau system that does not require loop-checks. The answer is that the tableau system will not be complete without this rule, that is, without this rule there are valid formulas which do not result in closed tableaus when given as input to the algorithm. This can be seen by considering the first tableau example of this subsection. In that example the valid formula $@_a\neg\Diamond(a \wedge \neg\Diamond a)$ is given as input, but without the rule $(Nom2)$, the algorithm will terminate already at line 10, resulting in a tableau that is not closed, and by inspecting the example, is straightforward to see that any other tableau having the same root formula will not be closed either, unless $(Nom2)$ is used. Thus, if $(Nom2)$ is omitted, something else must be added to regain completeness, and this will be the topic of the following section.

1.5.7 A Tableau System Not Including the Universal Modality

If the universal modality is omitted in the decision procedure for $\mathcal{H}(E)$ given above then of course a decision procedure for the weaker hybrid logic $\mathcal{H}$ is obtained. However, it turns out that if the universal modality is omitted, then it is possible to give a tableau system such that loop-checks are not needed to ensure termination of the decision procedure. The first tableau-based decision procedure for $\mathcal{H}$, that does not involve loop-checks, was a tableau system given in Bolander and Blackburn's paper (2007). In the present subsection we shall consider a similar tableau system not involving loop-checks. The system in the present subsection is obtained by directly modifying the tableau system, and the associated definitions and results, already introduced in the present section. One crucial modification is the replacement of the rule $(Nom2)$ by two new rules which are variants of a rule in Bolander and Blackburn (2007), however, most of the definitions and results already introduced can be reused. A difference between the system in Bolander and Blackburn (2007) and the systems under consideration here is that the present systems make use of unrestricted equational reasoning, which is not the case with the system in Bolander and Blackburn (2007).

Now, recall that the rules for the tableau system $\mathbf{T}_{\mathcal{H}(E)}$ were given in Figs. 1.6 and 1.7. The rules for the tableau system for $\mathcal{H}$ not involving loop-checks are obtained from the rules of Figs. 1.6 and 1.7 by omitting the rules (E) and $(\neg E)$ for the universal modality and by replacing the rule $(Nom2)$ by the two rules given in Fig. 1.8. The system thus obtained will be denoted $\mathbf{T}_{\mathcal{H}}$. Earlier in the present section we introduced some conventions for the rules of Figs. 1.6 and 1.7: Some rules were called destructive, some were called non-destructive, and some were called existential. These conventions are unchanged, but we add the convention that

$$\frac{@_a c, @_a \phi}{@_c \phi}\,(Id)^a \qquad \frac{@_a c, \neg @_a \phi}{\neg @_c \phi}\,(\neg Id)^a$$

[a] The nominal c and the formula ϕ are subformulas of the root formula

Fig. 1.8 New rules for the tableau system without the universal modality

the rule (Id) and $(\neg Id)$ are non-destructive (as destructive rules we only want rules with exactly one formula in the premise).

It is straightforward to check that all results for $\mathbf{T}_{\mathcal{H}(E)}$ in Sect. 1.5.3 also hold for the tableau system $\mathbf{T}_{\mathcal{H}}$, however, we shall need the following strengthened version of Theorem 1.22.

Theorem 1.35. *Let $@_a \Diamond b$ be a formula occurrence on a branch Θ of a tableau. Either $@_a \Diamond b$ is an accessibility formula occurrence on Θ or the formula $\Diamond b$ is a subformula of the root formula.*

Proof. Check each rule. Theorem 1.16 is needed in some of the cases. ■

The systematic tableau construction algorithm for $\mathbf{T}_{\mathcal{H}}$ is defined as follows. Note that the definition below is identical to Definition 1.26, the tableau construction algorithm for $\mathbf{T}_{\mathcal{H}(E)}$, except that the third restriction (the loop-check) of Definition 1.26 is omitted.

Definition 1.36 (Tableau construction algorithm). Given a formula $@_a \phi$ of $\mathcal{H}$ whose validity has to be decided, we define by induction a sequence $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$ of finite tableaus in $\mathbf{T}_{\mathcal{H}}$, each of which is embedded in its successor. Let $\mathcal{T}_0$ be the finite tableau constituted by the single formula $\neg @_a \phi$. If possible, apply an arbitrary rule to $\mathcal{T}_n$ with the following two restrictions:

1. If a formula to be added to a branch by applying a rule already occurs on the branch, then the addition of the formula is simply omitted.
2. After the application of a destructive rule to a formula occurrence ϕ on a branch, it is recorded that the rule was applied to ϕ with respect to the branch and the rule will not again be applied to ϕ with respect to the branch or any extension of it.

Let $\mathcal{T}_{n+1}$ be the resulting tableau.

In the proof of the theorem below we make use of the convention from Sect. 1.4.4 that the degree of a formula is the number of occurrences of non-nullary connectives in it.

Theorem 1.37. *The tableau construction algorithm for $\mathbf{T}_{\mathcal{H}}$ terminates in the sense that there exists an n such that $\mathcal{T}_n = \mathcal{T}_{n+1}$.*

Proof. Assume conversely that the algorithm does not terminate. Then the resulting tableau is infinite, and hence, has an infinite branch Θ. Analogous to the proof of Theorem 1.27, it follows from Proposition 1.24 and Theorem 1.16 that the graph $(N^\Theta, <_\Theta)$ has an infinite branch $a_1 <_\Theta a_2 <_\Theta a_3, \ldots$. Now, for any i, consider the

set of formula occurrences on Θ either having the form $@_{a_i}\phi$ where ϕ is not of the form $\Diamond b$ or having the form $\neg@_{a_i}\phi$. Let d_i be the maximal degree of such formula occurrences and let d_i be 0 if there are no such formula occurrences (by Theorem 1.16 the degrees of such formula occurrences are bounded by the degree of the root formula plus two). By inspection of the rules, it is straightforward to see that $d_i > d_{i+1}$ for any i such that $d_{i+1} > 0$, where in the case with the rule $(\neg\Diamond)$ we use Theorem 1.35 and in the cases with the rules $(@)$ and $(\neg@)$ we use Theorem 1.16. Hence, there exists a j such that any formula $@_{a_j}\phi$ occurring on Θ has the property that ϕ is of the form $\Diamond b$ or ϕ has degree 0, and any formula $\neg@_{a_j}\phi$ occurring on Θ has the property that ϕ has degree 0. This contradicts $a_j <_\Theta a_{j+1}$ being the case. Thus, the algorithm terminates. ■

Informally, the proof above is based on the observation that formulas $@_a\phi$ and $\neg@_a\phi$ in the branch Θ get smaller when the path from the nominal a to a root in the graph $(N^\Theta, <_\Theta)$ gets longer (except for formulas of the form $@_a\Diamond b$). A similar observation is also the basis of the standard termination proof for prefixed tableau systems for the modal logic K, cf. the book Fitting (1983).

In what follows, we shall assume that Θ is a given branch of a tableau generated by the systematic tableau construction algorithm Definition 1.36. Before we come to the model existence theorem, we introduce some important machinery. Given a nominal a in N^Θ, we let $[a]_\sim$ denote the equivalence class of a with respect to the binary relation $\sim$ and we let $N^\Theta/\sim$ denote the set of equivalence classes. Note that it follows from Corollary 1.20 that any non-singleton equivalence class $[a]_\sim$ contains a nominal with an equational occurrence in the root formula.

Definition 1.38. Given some fixed total order on N^Θ, we define a function u from N^Θ to N^Θ as follows: If $[a]_\sim$ is non-singleton, then we let $u(a)$ be the smallest nominal in $[a]_\sim$ with an equational occurrence in the root formula, and if $[a]_\sim$ is singleton, then we let $u(a)$ be a. The nominal $u(a)$ is called the *urfather* of a.

The idea of letting a function pick out an equivalent nominal occurring equationally in the root formula, if such a nominal exists, stems from the paper Bolander and Blackburn (2007) where a similar function is defined. The definition above leads to the following proposition.

Proposition 1.39 (Urfather closure property). *Assume that the branch Θ is open. Let ϕ be a subformula of the root formula. If $@_a\phi$ occurs on Θ, then also $@_{u(a)}\phi$ occurs at Θ, and similarly, if $\neg@_a\phi$ occurs on Θ, then also $\neg@_{u(a)}\phi$ occurs on Θ.*

Proof. Follows straightforwardly from applications of the rules (Id) and $(\neg Id)$. ■

The urfather closure property is a basic idea behind the tableau system given in Bolander and Blackburn (2007). Intuitively, the urfather closure property allows information to be moved freely from any world to identical worlds referred to in the root formula.

Definition 1.40. Let R be the binary relation on N^Θ defined by aRc if and only if there exists a nominal $c' \sim c$ such that $@_{u(a)}\Diamond c'$ occurs on Θ.

We let $\overline{R}$ be the binary relation on $N^\Theta/\sim$ defined by $[a]_\sim \overline{R} [c]_\sim$ if and only if aRc.

Definition 1.41. For any element a of N^Θ, let V_a be the function that to each ordinary propositional symbol assigns an element of $\{0,1\}$ such that $V_a(p) = 1$ if $@_a p$ occurs on Θ and $V_a(p) = 0$ otherwise.

We let $\overline{V}_{[a]_\sim}$ be defined by $\overline{V}_{[a]_\sim}(p) = V_a(p)$. We are now ready to define a model.

Definition 1.42. Let $\mathfrak{M}$ be the model $(N^\Theta/\sim, \overline{R}, \{\overline{V}_{[a]_\sim}\}_{[a]_\sim \in N^\Theta/\sim})$ and let the assignment g for $\mathfrak{M}$ be defined by $g(a) = [a]_\sim$.

Theorem 1.43 (Model existence). *Assume that the branch Θ is open. For any satisfaction statement $@_a\phi$ where ϕ is a subformula of the root formula, the following two statements hold.*

- *If $@_a\phi$ occurs on Θ, then it is the case that $\mathfrak{M}, g, [a]_\sim \models \phi$.*
- *If $\neg @_a\phi$ occurs on Θ, then it is not the case that $\mathfrak{M}, g, [a]_\sim \models \phi$.*

Proof. induction on the structure of ϕ. We only cover the case where ϕ is of the form $\Diamond\psi$, all the other cases are exactly as in the full proof of Theorem 1.32, which can be found in the book Braüner (2011).

Assume that $@_a\Diamond\psi$ occurs on the branch Θ. We then have to prove that $\mathfrak{M}, g, [a]_\sim \models \Diamond\psi$, that is, for some equivalence class $[c]_\sim$ such that $[a]_\sim \overline{R} [c]_\sim$, it is the case that $\mathfrak{M}, g, [c]_\sim \models \psi$. If ψ is a nominal, say b, we just have to prove that $[a]_\sim \overline{R} [b]_\sim$, which trivially follows from the definition of the relation $\overline{R}$ together with the observation that if $@_a\Diamond b$ occurs on Θ, then by Proposition 1.39 also $@_{u(a)}\Diamond b$ occurs on Θ. If ψ is not a nominal, by Proposition 1.39 and the rule $(\Diamond)$ also some formulas $@_{u(a)}\Diamond c$ and $@_c\psi$ occur on Θ. Clearly, $[a]_\sim \overline{R} [c]_\sim$ and $\mathfrak{M}, g, [c]_\sim \models \psi$ by induction.

Assume that $\neg @_a\Diamond\psi$ occurs on the branch Θ. We then have to prove that $\mathfrak{M}, g, [a]_\sim \models \Diamond\psi$ does not hold, that is, for any equivalence class $[c]_\sim$ such that $[a]_\sim \overline{R} [c]_\sim$, it is not the case that $\mathfrak{M}, g, [c]_\sim \models \psi$. From $[a]_\sim \overline{R} [c]_\sim$ it follows that there exists a nominal $c' \sim c$ such that $@_{u(a)}\Diamond c'$ occurs on Θ. By Proposition 1.39, $\neg @_{u(a)}\Diamond\psi$ occurs on Θ. Thus, by the rule $(\neg\Diamond)$ the formula $\neg @_{c'}\psi$ occurs on Θ. By induction we conclude that $\mathfrak{M}, g, [c']_\sim \models \psi$ does not hold and trivially, $[c']_\sim = [c]_\sim$. ■

Given the machinery introduced above, the decision procedure is defined exactly as in Definition 1.33.

It should be mentioned that the rules (Id) and $(\neg Id)$ can be restricted to the case where ϕ is of the form $\Diamond\psi$ without affecting the results for $\mathbf{T}_\mathcal{H}$ we consider in this subsection, except Proposition 1.39, the urfather closure property, which is restricted in the same way.

As an example, consider the formula $@_a\neg\Diamond(a \wedge \neg\Diamond a)$ which also was considered in the first tableau example for $\mathbf{T}_{\mathcal{H}(E)}$ given in Sect. 1.5.6. Given this formula as input, a possible tableau generated by the tableau construction algorithm for $\mathbf{T}_\mathcal{H}$, Definition 1.36, is the tableau below where we have imposed the restriction on the rules (Id) and $(\neg Id)$ mentioned in the previous paragraph.

$\neg@_a\neg\Diamond(a\wedge\neg\Diamond a)$	1.
$@_a\Diamond(a\wedge\neg\Diamond a)$	2. by $(\neg\neg)$ rule on 1
$@_c(a\wedge\neg\Diamond a)$	3. by $(\Diamond)$ rule on 2
$@_a\Diamond c$	4. by $(\Diamond)$ rule on 2
$@_c a$	5. by $(\wedge)$ rule on 3
$@_c\neg\Diamond a$	6. by $(\wedge)$ rule on 3
$\neg@_c\Diamond a$	7. by $(\neg)$ rule on 6
$@_c c$	8. by (Ref) rule
$@_a c$	9. by $(Nom1)$ rule on 5 and 8
$@_a a$	10. by (Ref) rule
$\neg@_a\Diamond a$	11. by $(\neg Id)$ rule on 5 and 7
$\neg@_c a$	12. by $(\neg\Diamond)$ rule on 11 and 4

Note that lines 1–10 in the tableau above are identical to lines 1–10 in the tableau example for $@_a\neg\Diamond(a\wedge\neg\Diamond a)$ given in Sect. 1.5.6, thus, it is only in the last two lines that the difference between $\mathbf{T}_{\mathcal{H}}$ and $\mathbf{T}_{\mathcal{H}(E)}$ crops up. Note that the branch is closed since it contains the formula $@_c a$ as well as $\neg@_c a$.

1.5.8 A Hybrid-Logical Version of Analytic Cuts

In this subsection we shall consider an alternative version of the tableau system without loop-checks, $\mathbf{T}_{\mathcal{H}}$, which was considered above. Now, at the end of Sect. 1.5.6 we concluded that if the rule $(Nom2)$ is omitted from $\mathbf{T}_{\mathcal{H}(E)}$, something else must be added to regain completeness, and in the version of $\mathbf{T}_{\mathcal{H}}$ considered above, the rules (Id) and $(\neg Id)$ in Fig. 1.8 were added. In the alternative version of $\mathbf{T}_{\mathcal{H}}$ we consider in this subsection, we shall investigate how much standard proof-theoretic machinery in the form of cuts is needed to replace the $(Nom2)$ rule (with the implicit requirement that loop-checks are avoided).

To explicate the goal of this investigation, we shall make some general remarks about cuts. The following rule is the standard *cut* rule.

$$\frac{}{\phi \mid \neg\phi}\ (Cut)$$

The formula ϕ is called the *cut-formula*. As mentioned in the beginning of Sect. 1.5.1, the idea behind tableau systems is to mimic the recursive truth-conditions in the semantics, whereby a formula is broken down into its components. The cut rule does not follow this idea since no formula is broken down into its components. If a tableau system includes the cut rule, then the cut rule can most often be proved to be redundant, that is, the cut rule is admissible in the system obtained by leaving out the cut rule. However, in some cases the cut rule is not completely redundant, but a restricted version is needed where the cut-formula is a subformula of the root formula. A cut rule with this restriction is called an *analytic*

$$\frac{}{@_a\phi \mid \neg @_a\phi}\ (\textit{Quasi-analytic cut})^{a,b}$$

[a] The nominal a and the formula ϕ are subformulas of the root formula
[b] None of the formulas $@_a\phi$ and $\neg @_a\phi$ are on the branch

Fig. 1.9 A hybrid-logical version of the analytic cut rule

cut rule. In most tableau systems not including the cut rule, tableaus satisfy the subformula property which says that any formula in a tableau is a subformula of root formula. Clearly, this property does not hold if the cut rule is allowed (but note that the property is not violated by analytic cuts).

The alternative version of $\mathbf{T}_{\mathcal{H}}$ is obtained by replacing the rules in Fig. 1.8 by the rule in Fig. 1.9, and moreover, by changing the definition of an open branch in a tableau such that a branch is called *open* if for no satisfaction statements $@_a\chi$ and $@_ab$ occurring on the branch, it is the case that $\neg @_b\chi$ also occurs on the branch. To avoid excessive proliferation of terminology, we use the notation $\mathbf{T}_{\mathcal{H}}$ also for the alternative system. As in the case of the rules (Id) and $(\neg Id)$, the rule $(\textit{Quasi-analytic cut})$ is classified as non-destructive. Note that in the alternative version of $\mathbf{T}_{\mathcal{H}}$, the rules (Id) and $(\neg Id)$ are derivable.[22] Of course, the rule $(\textit{Quasi-analytic cut})$ is a hybrid-logical version of the standard analytic cut rule, cf. the previous paragraph.

Note that in one branch of a tableau, there can only be finitely many applications of the rule $(\textit{Quasi-analytic cut})$ if the cut-formulas are different, the reason being that there are only finitely many subformulas of the root formula. It is straightforward to check that all the results for the first version of $\mathbf{T}_{\mathcal{H}}$ also hold for the alternative version considered in this subsection. A difference between the two versions of $\mathbf{T}_{\mathcal{H}}$ is that in the first version, the urfather closure property is essentially built-in as two derivation rules, namely (Id) and $(\neg Id)$, whereas in the alternative version, the urfather closure property follows from the presence of other rules, in particular $(\textit{Quasi-analytic cut})$, together with a more general closure condition on branches. At a more intuitive level, a difference between the two versions of $\mathbf{T}_{\mathcal{H}}$ is that the rules (Id) and $(\neg Id)$ have a semantical motivation (the intuition being that they allow information to be moved freely from any world to identical worlds referred to in the root formula) whereas the rule $(\textit{Quasi-analytic cut})$ has a proof-theoretical motivation (it is a hybrid-logical version of standard proof-theoretic machinery, namely the analytic cut rule).

A couple of more general remarks should be made in connection with analytic cut rules. A defence of analytic cuts can be found in the paper D'Agostino and Mondadori (1994) where it is pointed out that ordinary cut-free tableau and Gentzen systems have a number of anomalies that can be avoided in proof systems allowing analytic cuts. According to that paper, cut-free systems are anomalous from three different points of view.

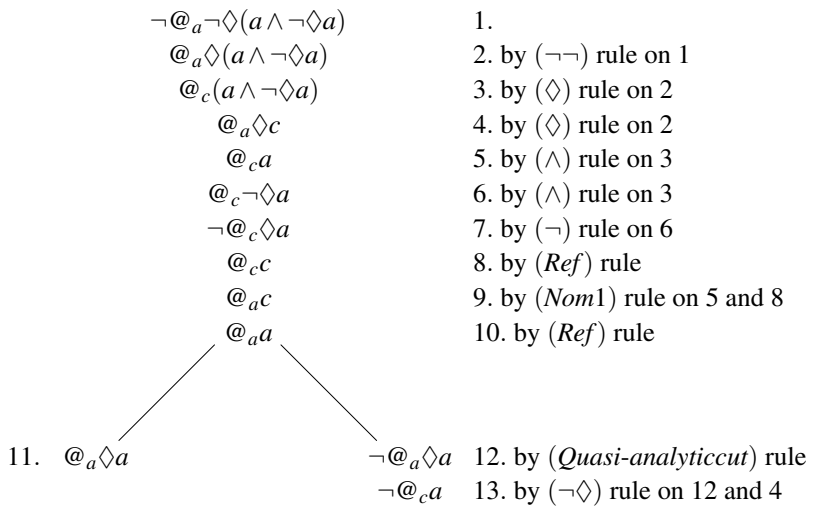
[22] This was pointed out to the author by Jens Ulrik Hansen.

1. From a proof-theoretical point of view, it is an anomaly that cut-free systems cannot represent lemmas in proofs.
2. From a semantical point of view, it is an anomaly that cut-free systems cannot express the bivalence of classical logic.
3. From a computational point of view, it is an anomaly that for some classes of propositional formulas, decision procedures based on cut-free systems are incomparably slower than the truth-table method (in the more precise technical sense that there is no polynomial time computable function that maps truth-table proofs of such formulas to proofs of the same formulas in cut-free tableau or Gentzen systems).

In relation to the computational anomaly, see also the paper Boolos (1984) where examples of first-order formulas are given whose derivations in cut-free systems are much larger than their derivations in natural deduction systems, which implicitly allow unrestricted cuts (in one case more than 10^{38} characters compared to less than 3280 characters). However, at present it is not clear to which extent the discussion outlined above is directly relevant to the proof-theory of hybrid logic.

Above it was mentioned that the rules (Id) and $(\neg Id)$ of Fig. 1.8 can be restricted to the case where ϕ is of the form $\Diamond\psi$. This also applies to the rule $(Quasi\text{-}analytic\ cut)$ in the alternative version of $\mathbf{T}_{\mathcal{H}}$ considered in this subsection, however, in the light of the remarks in the previous paragraph, it is not clear whether this restriction on $(Quasi\text{-}analytic\ cut)$ is desirable.

As an example we consider the formula $@_a\neg\Diamond(a \wedge \neg\Diamond a)$ which was also considered in connection with the first version of $\mathbf{T}_{\mathcal{H}}$ (and in connection with $\mathbf{T}_{\mathcal{H}(E)}$ in Sect. 1.5.6). A possible tableau generated by the tableau construction algorithm for the alternative version of $\mathbf{T}_{\mathcal{H}}$ is the tableau below where we have imposed the restriction on $(Quasi\text{-}analytic\ cut)$ mentioned in the previous paragraph.



Note that lines 1–10 in the tableau above are identical to lines 1–10 in the tableau in connection with the first version of $\mathbf{T}_{\mathcal{H}}$ (and the tableau in connection with $\mathbf{T}_{\mathcal{H}(E)}$ in Sect. 1.5.6). Also, note that both branches are closed, and in the case of the left-hand-side branch, we make use of the more general closure condition for the alternative version of $\mathbf{T}_{\mathcal{H}}$ (the branch contains $@_a\Diamond a$ and $@_a c$ as well as $\neg@_c\Diamond a$).

Remark: Let $@_a\phi$ be a formula whose validity has to be decided. Thus, according to the decision procedure, a tableau is constructed with $\neg@_a\phi$ as the root formula. Clearly, $\neg@_a\phi$ is equivalent to the formula

$$\neg@_a\phi \wedge \bigwedge_{i=1}^{n}\bigwedge_{j=1}^{m}(@_{c_i}\psi_j \vee \neg@_{c_i}\psi_j)$$

where $c_1,\ldots,c_n$ and $\psi_1,\ldots,\psi_m$ are respectively the nominals and the formulas that are subformulas of $\neg@_a\phi$. If the tableau having $\neg@_a\phi$ as the root is closed, then a closed tableau having the displayed formula (strictly speaking prefixed by a "dummy" satisfaction operator to fit the format of the tableau system) as root can be constructed without applying the rule (*Quasi-analytic cut*). Conversely, if the tableau having $\neg@_a\phi$ as the root has an open branch Θ, then a tableau having the displayed formula (again, strictly speaking prefixed by a dummy satisfaction operator) as root can be constructed without applying (*Quasi-analytic cut*) such that the tableau has an open branch containing all the formulas of Θ. Thus, applications of (*Quasi-analytic cut*) are dispensable in the sense that they can be simulated by preprocessing the input formula to the tableau construction algorithm, where in the preprocessing a conjunct is added for each possible application of (*Quasi-analytic cut*).[23]

1.5.9 Discussion

The tableau system $\mathbf{T}_{\mathcal{H}(E)}$ given in Sect. 1.5.2 is a slightly simplified version of a system given in the paper Bolander and Braüner (2006), that is, the system given in Bolander and Braüner (2006) includes the rule

$$\frac{@_c\Diamond a, @_a b}{@_c\Diamond b}\ (\mathit{Bridge})$$

which has turned out to be superfluous. A tableau system similar to that of Bolander and Braüner (2006) was considered in Blackburn (2000a). The tableau-based decision procedure given in Bolander and Braüner (2006) was published already in Bolander and Braüner (2005). We are only aware of one tableau-based decision

[23] This was pointed out to the author by Thomas Bolander.

procedure for hybrid logic published before the publication of Bolander and Braüner (2005), namely the prefixed tableau system given in Miroslava Tzakova's paper Tzakova (1999). However, it turns out that Tzakova's termination proof is flawed.

Loop-checks are applicable in connection with a spectrum of different proof systems. This is corroborated by the fact that in the paper Bolander and Braüner (2006) loop-checks are applied in connection with yet another three kinds of proof systems for the hybrid logic $\mathcal{H}(E)$, namely a prefixed tableau system along the lines of the system given in Tzakova (1999), a tableau system involving a rule for nominal substitution, and a Gentzen system. The Gentzen system is described in details in the book Braüner (2011).

In ordinary modal logic, loop-checks are used in connection with standard Fitting-style prefixed tableau systems for transitive logics such as $\mathsf{K4}$ (see Goré 1999; Massacci 2000). The loop-check technique can be tracked to Fitting (1983), although a similar idea was involved in a graphical formalism for deciding validity of modal-logical formulas in the earlier book Hughes and Cresswell (1968). Now, a simple prefixed tableau system can be formulated for the modal logic K such that a systematic tableau construction always terminates (cf. Fitting 1983). The systematic tableau construction algorithm for K does not involve loop-checks. However, if the tableau system for K is extended with the standard prefixed tableau rule for transitivity

$$\frac{(a,\Box\phi),R(a,b)}{(b,\Box\phi)}$$

(the notation should be self-explanatory) whereby a tableau system for $\mathsf{K4}$ is obtained, then a systematic tableau construction may not terminate. Intuitively, the problem is that the rule allows information to be moved forward from a world to any accessible world. The standard way to fix this problem is to incorporate loop-check conditions on the applications of existential rules. The intuitive reason why this technique works in the context of hybrid logic, is that the problem here is also that information can be moved between worlds, namely in connection with applications of the rules $(Nom1)$ and $(Nom2)$ in the tableau system described in Sect. 1.5.2. Intuitively, these rules allow atomic information to be moved freely between worlds that are identical.

There is a close connection between the hybrid logic $\mathcal{H}(E)$ and description logics, which are a family of logics used for knowledge representation in Artificial Intelligence, see the paper Blackburn and Tzakova (1998) and Carlos Areces' PhD thesis Areces (2000). The hybrid logic $\mathcal{H}(E)$ is the mono-modal hybrid logic $\mathcal{H}$ extended with the universal modality, but all the results in the present section also hold if a multi-modal version of the hybrid logic $\mathcal{H}$ is extended with the universal modality, that is, if the single modal operator $\Diamond$ in the hybrid logic is replaced by an arbitrary, finite number of modal operators $\Diamond_1, \ldots, \Diamond_m$. Such a multi-modal hybrid logic with the universal modality can be seen as a natural generalisation of a description logic. Now, the description logic called $\mathcal{ALC}$ is a notational variant of ordinary multi-modal logic, that is, propositional logic extended with a finite number of modal operators $\Diamond_1, \ldots, \Diamond_m$. The *concept expressions* of $\mathcal{ALC}$ simply

correspond to formulas of multi-modal logic and vice versa. Given a description logic, for example $\mathcal{ALC}$, a knowledge base is a set of metalinguistic statements expressing relationships between concepts and individuals. There are two kinds of metalinguistic statements; they are called *TBox-statements* and *ABox-statements* respectively. A TBox-statement $\phi \sqsubseteq \psi$ expresses that the concept ϕ is subsumed by the concept ψ, that is, that any individual that belongs to the extension of ϕ also belongs to the extension of ψ. An ABox-statement $\phi(a)$ expresses that the individual a belongs to the extension of the concept ϕ and an ABox-statement $R_i(a,c)$ expresses that the individual a is R_i-related to the individual c. This can all be expressed in terms of the multi-modal hybrid logic with the universal modality: The TBox-statement above is expressed by the formula $A(\phi \rightarrow \psi)$, where A is the universal modality, and the ABox-statements are expressed by the formulas $@_a\phi$ and $@_a\Diamond_i c$. Note that no binders are needed. Of course, a nominal is here considered a name of an individual. Thus, the hybrid logic here can be seen as a generalised version of this description logic where no distinction between an object language and a metalanguage is made.

Nominals are often used in description logics, and certain tableau-based decision procedures for such logics also make use of loop-checks. An example is the decision procedure given in Horrocks and Sattler (2005) which is based on a prefixed tableau system that uses metalinguistic prefixes and accessibility formulas.

1.6 Why Does the Proof-Theory of Behave So Well?

The material in this section is primarily of a conceptual nature, the goal of the section being to put into perspective hybrid logic and the proof-theory of hybrid logic. In the section we shall try to give an answer to the following question: Why does the proof-theory of hybrid logic behave so well compared to the proof-theory of ordinary modal logic?

We start by explaining what we mean when we say that the proof-theory of hybrid logic behaves well. That is, we shall describe our success criteria. We state our success criteria in terms of natural deduction systems (but similar criteria can be given for Gentzen systems).

1. The introduction and elimination rules associated with each connective satisfy Prawitz' inversion principle.
2. The system satisfies normalisation such that normal derivations satisfy a version of the subformula property.
3. Conditions on the accessibility relation can be incorporated into the system in a uniform way, that is, by just adding appropriate rules.

We find that these three criteria are absolutely central.[24] As we explained in Sect. 1.4.1, it follows from Prawitz' inversion principle that by rewriting a derivation, it is possible to remove a formula occurrence that is both introduced by an introduction rule and eliminated by an elimination rule, that is, a maximum formula. This rewrite process is formalized in proper reduction rules, hence, the inversion principle can be seen as a prerequisite for formulating a normalisation theorem since such a theorem is relative to a set of reduction rules.

Now, roughly, there are two different kinds of natural deduction, Gentzen, and tableau systems for ordinary modal logic. The first kind of systems are called *labelled* systems where formulas involved in the rules are metalinguistic formulas obtained by attaching labels to ordinary modal-logical formulas. The labels of labelled systems represent possible worlds of the Kripke semantics. Labelled systems often also involve an explicit representation of the accessibility relation of the Kripke semantics. Thus, rules in labelled systems are rules for reasoning directly about the Kripke models. The second kind of systems are systems where formulas involved in the rules are formulas of the object language, that is, ordinary modal-logical formulas. Systems of the second kind will be called *standard* systems.[25]

In what follows we discuss standard and labelled systems and we then try to give an answer to the initial question of why the proof-theory of hybrid logic behave so well compared to the proof-theory of ordinary modal logic. At the end of this section we make some concluding philosophical remarks. The material in the section stems from the paper Braüner (2007) and has been further developed in the book Braüner (2011).

[24] Also other criteria could be considered, one important example being interpolation, that is, the criterion that a proof system should lend itself to the calculation of interpolants, perhaps after being enhanced with further machinery, like the tableau system for first-order hybrid logic which in the paper Blackburn and Marx (2003) is used as the basis of an algorithm that calculates interpolants. See the remarks on interpolation in Sect. 1.3. Note that there are two steps: The first step is the requirement of a logic (which here is a formal language together with a semantics) that it satisfies interpolation. This might be proved semantically, independent of any proof systems. If the logic does satisfy interpolation, then the second step is the requirement of a proof system for the logic that the proof system in question can be used as the basis for calculating interpolants.

[25] It should be mentioned that there are a number of natural deduction and Gentzen style formulations for modal logic that do not fit this categorisation well. Notable here are formulations in terms of Nuel Belnap's display logic and Kosta Dŏsen's higher-level sequents. However, these formulations differ considerably from Gentzen's original natural deduction and sequent systems and they are more complicated from a technical point of view. (Although it has to be acknowledged that display sequents as well as higher-level sequents were introduced as natural generalisations of Gentzen's notion of a sequent, intended to allow a uniform sequent-style formulation of many different logics.) An overview can be found in Wansing (1994). Also notable are modal hypersequent systems, see Avron (1996) as well as the handbook chapter Fitting (2007).

1.6.1 Standard Systems for Modal Logic

No known standard natural deduction systems for modal logic satisfy all three success criteria given above. The first two criteria are satisfied by a number of systems, but when a standard modal-logical natural deduction, Gentzen, or tableau system is given, it is usually for one particular modal logic. This is for example the case with the natural deduction systems for the modal logics S4 and S5 given in Prawitz' book Prawitz (1965).[26] With reference to Prawitz's systems for S4 and S5, Robert Bull and Krister Segerberg note the following in their survey paper on modal logic in *Handbook of Philosophical Logic*.

> However, it has proved difficult to extend this sort of analysis to the great multitude of other systems of modal logic. It seems fair to say that a deductive treatment congenial to modal logic is yet to be found, for Hilbert systems are not suited for the purpose of actual deduction, ... (Bull and Segerberg 2001, p. 25).

Bull and Segerberg continue.

> ... only exceptional systems ... seem to be characterizable in terms of reasonably simple rules (Bull and Segerberg 2001, p. 27).

In the paper (Wansing 1994) Heinrich Wansing gives a summary of the status of modal-logical proof-theory in which he comments on the above passage from Bull and Segerberg (2001) as follows.

> Compared with the multitude of not only existing but also interesting axiomatically presentable normal modal propositional logics, the number of systems for which sequent calculus presentations (of some sort) are known is disappointingly small. In contrast to the axiomatic approach, the standard sequent-style proof theory for normal modal logics fails to be 'modular', and the very mechanism behind the small range of known possible variations is not very clear. One might be inclined to agree with Segerberg's ... remark (in connection with natural deduction systems for modal logics) that 'only exceptional systems ... seem to be characterizable in terms of reasonably simple rules' (Wansing 1994, p. 128).

We find that the lack of uniformity described above is a major deficiency of standard modal-logical proof-theory.

1.6.2 A Labelled system for Modal Logic

Contrary to standard natural deduction systems for modal logic, labelled systems usually satisfy all three criteria given initially in this section. Thus, rules for reasoning directly about the Kripke models are proof-theoretically well-behaved.

---

[26] In fact, Prawitz' systems for S4 and S5 deviate from most standard systems since his introduction rules for $\Box$ make use of "non-local" side-conditions, that is, side-conditions that do not just refer to the premises of the rules and to undischarged assumptions, but to the whole derivations of the premises.

This is for example the case with a natural deduction system for classical modal logic given by Luca Viganò et al. in the paper Basin et al. (1997). See also Viganò's book (Viganò 2000). Viganò's system can be seen as a classical version of a natural deduction system for intuitionistic modal logic given by Alex Simpson in 1994.

Viganò and Simpson's systems both involve metalinguistic formulas of two sorts, namely atomic first-order formulas of the form $R(a,c)$ and labelled formulas of the form (a,ϕ) where ϕ is a modal-logical formula. The first sort is used for relational reasoning and the second sort is used for propositional reasoning, relative to worlds. This metalinguistic machinery enables the formulation of the following introduction and elimination rules for the modal operator.

$$\frac{\begin{array}{c}[R(a,c)]\\ \vdots\\ (c,\phi)\end{array}}{(a,\Box\phi)}\,(\Box I) \qquad\qquad \frac{(a,\Box\phi)\quad R(a,c)}{(c,\phi)}\,(\Box E)$$

The introduction rule, that is, the rule $(\Box I)$, is equipped with the side-condition that the label c does not occur in $(a,\Box\phi)$ or in any undischarged assumptions other than the specified occurrences of $R(a,c)$. Compare these introduction and elimination rules to the truth-condition for the modal operator in the Kripke semantics.

$\Box\phi$ is true at a iff for any c such that aRc, ϕ is true at c

Clearly, the introduction rule for the modal operator can be read off from the right-to-left direction of the truth-condition and the elimination rule can be read off from the left-to-right direction. Thus, the rules can be read off from the modal operator's truth-condition in the Kripke semantics. We shall come back to this in Sect. 1.6.6. Beside the above introduction and elimination rules for the modal operator, the metalinguistic machinery enables the formulation of rules for first-order conditions on the accessibility relation, in Viganò's case conditions expressed by Horn clause theories, and in Simpson's case, conditions expressed by geometric theories.

The derivation rules of a slightly modified version of Viganò's labelled system for modal logic is given in Fig. 1.10. All formulas in the rules are metalinguistic formulas of the forms $R(a,c)$ and (a,ϕ). It is assumed that a countably infinite set $a, b, c, \ldots$ of labels is given. The system is sound and complete in the appropriate sense (see Basin et al. 1997). It is instructive to compare the rules for the labelled system given in Fig. 1.10 with the rules for the system $\mathbf{N}_{\mathcal{H}}$ which are given in Figs. 1.1 and 1.2 of Sect. 1.4.2 (the rules for binders are disregarded). First note that contrary to the labelled system, all formulas in the rules for $\mathbf{N}_{\mathcal{H}}$ are formulas of the object language, thus, in this sense the system $\mathbf{N}_{\mathcal{H}}$ is internalized.

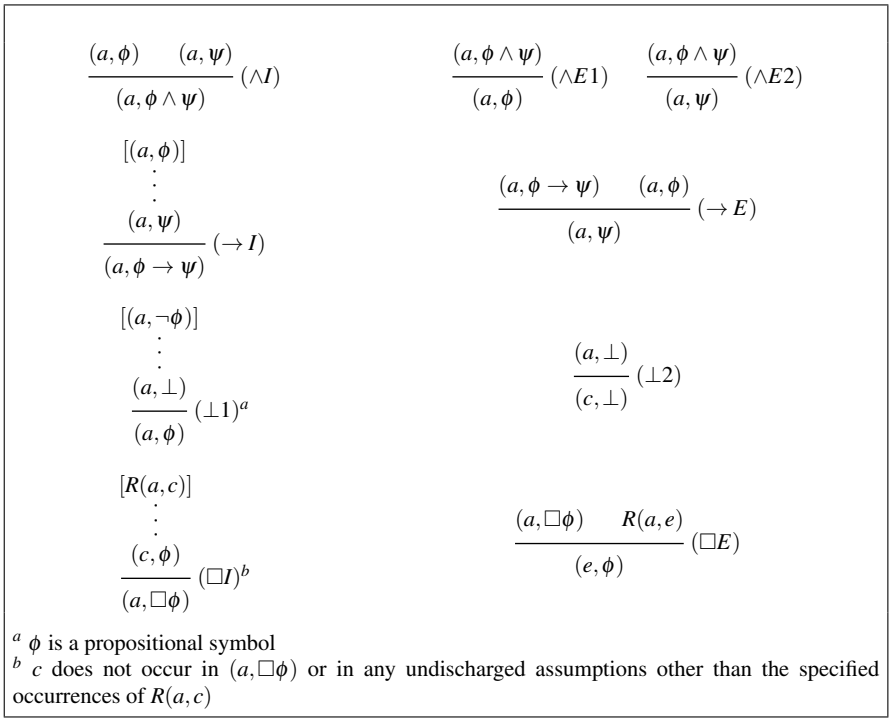


Fig. 1.10 Labelled natural deduction rules for modal logic

1.6.3 The Internalisation Translation

It is straightforward to translate formulas and derivations of the labelled system for modal logic given in Fig. 1.10 into the internalized system $\mathbf{N}_{\mathcal{H}}$ for hybrid logic given in Sect. 1.4.2. We shall call this translation the *internalisation* translation and denote it I. A metalinguistic formula ϕ of the first system is translated to a hybrid-logical satisfaction statement $I(\phi)$ by letting $I((a,\phi)) = @_a\phi$ and $I(R(a,c)) = @_a\Diamond c$. Obviously, the internalisation translation preserves the semantics, where the modal-logical semantics is defined in an appropriate way, taking the metalinguistic machinery into account (details are left to the reader).

Having translated formulas, we translate derivations. In the next subsection we show that the translation preserves reductions and this involves a small lemma saying that the translation commutes with substitution of derivations for parcels of undischarged assumptions, and since a derivation is substituted for each undischarged assumption in a specified parcel, we need to be able to keep track of the identity of parcels when translating a derivation. To this end we introduce a few further conventions: A set of annotated formulas will be called a *context* and the metavariables Φ, Ψ, Ω, ... will range over contexts. Moreover, a

derivation π is a *derivation from* a context Φ if each undischarged assumption in π is an occurrence of an annotated formula in Φ. Note that in Sect. 1.4 we considered derivations as being derivations from sets of formulas, that is, we ignored numbers annotating undischarged assumptions. Keeping the numbers as we do in the present section enables us to keep track of the identity of parcels of undischarged assumptions when translating a derivation. The above translation of metalinguistic formulas is extended to contexts in the obvious way, namely by letting $I(\Phi) = \{(I(\theta))^r \mid \theta^r \in \Phi\}$.

Definition 1.44. A derivation π of ϕ from Φ in the modal-logical natural deduction system is translated to a derivation $I(\pi)$ of $I(\phi)$ from $I(\Phi)$ in the hybrid-logical natural deduction system $\mathbf{N}_{\mathcal{H}}$ by replacing each formula occurrence ψ in π by $I(\psi)$.

Note that the hybrid-logical introduction and elimination rules for the connectives $\wedge$, $\rightarrow$, and $\Box$ can be seen as obtained by taking the image under the translation I of the labelled modal-logical rules for these connectives.

1.6.4 Reductions

In this subsection we show that the internalisation translation preserves reductions. Before doing so, we need to fix reduction rules for the natural deduction systems under consideration. We have already given reduction rules for $\mathbf{N}_{\mathcal{H}}$ in Sect. 1.4.6, so we just need to give reduction rules for the modal-logical natural deduction system. First some conventions in connection with the modal-logical derivation rules. The premise of the form $R(a,e)$ in the rule $(\Box E)$ is called the *relational premise* and the premise of the form (a,ϕ) in the rule $(\rightarrow E)$ is called the *minor premise*. A premise of an elimination rule that is neither minor nor relational is called *major*. Of course, these conventions are analogous to the conventions for $\mathbf{N}_{\mathcal{H}}$ given in Sect. 1.4.6.

As usual for natural deduction systems, a *maximum formula* in a derivation is a formula occurrence that is both the conclusion of an introduction rule and the major premise of an elimination rule. Maximum formulas can be removed by applying *reduction rules*. The rules for reductions of the modal-logical system are as follows.

$(\wedge I)$ followed by $(\wedge E1)$ (analogously in the case of $(\wedge E2)$)

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_1\\(a,\phi)\end{array}\qquad\begin{array}{c}\vdots\,\pi_2\\(a,\psi)\end{array}}{(a,\phi\wedge\psi)}}{(a,\phi)} \quad\rightsquigarrow\quad \begin{array}{c}\vdots\,\pi_1\\(a,\phi)\end{array}$$

$(\rightarrow I)$ followed by $(\rightarrow E)$

$$\dfrac{\dfrac{\begin{array}{c}[(a,\phi)]\\ \vdots\,\pi_1\\ (a,\psi)\end{array}}{(a,\phi\rightarrow\psi)} \qquad \begin{array}{c}\vdots\,\pi_2\\ (a,\phi)\end{array}}{(a,\psi)} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\,\pi_2\\ (a,\phi)\\ \vdots\,\pi_1\\ (a,\psi)\end{array}$$

$(\Box I)$ followed by $(\Box E)$

$$\dfrac{\dfrac{\begin{array}{c}[R(a,c)]\\ \vdots\,\pi_1\\ (c,\phi)\end{array}}{(a,\Box\phi)} \qquad \begin{array}{c}\vdots\,\pi_2\\ R(a,e)\end{array}}{(e,\phi)} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\,\pi_2\\ R(a,e)\\ \vdots\,\pi_1[e/c]\\ (e,\phi)\end{array}$$

Note that the reduction rules for $\mathbf{N}_{\mathcal{H}}$ given in Sect. 1.4.6 can be seen as obtained by applying the internalisation translation to all formulas displayed in the modal-logical reduction rules above, and adding a reduction rule for satisfaction operators. The reduction rules above can also be found in Basin et al. (1997) and Simpson (1994).

As usual for natural deduction systems, a derivation is *normal* if it contains no maximum formula. Using a variation of a standard technique for ordinary classical first-order logic (originally given in Prawitz (1965)), in Basin et al. (1997) it is proved that the modal-logical system satisfies a *normalisation* theorem, that is, any derivation can be rewritten to a normal derivation by repeated applications of reductions. The standard technique used in this normalisation proof is the technique mentioned in Theorem 1.13 of Sect. 1.4.6 which does *not* work directly for the hybrid-logical natural deduction system. Besides normalisation, in Basin et al. (1997) it is proved that every normal derivation satisfies a version of the subformula property.

Before proving that the internalisation translation preserves reductions, we give a small lemma which says that the internalisation translation commutes with substitution of derivations for undischarged assumptions in derivations.

Lemma 1.45. *Let τ and π be modal-logical derivations such that τ is a derivation of ϕ from Φ and π is a derivation from $\{\phi^r\}\cup\Phi\cup\Psi$ where $\phi^r\notin\Phi\cup\Psi$ and $\Phi\cap\Psi=\emptyset$. Moreover, let κ be the derivation obtained by substituting $I(\tau)$ for $(I(\phi))^r$ in $I(\pi)$ and let λ be the derivation obtained by substituting τ for ϕ^r in π. Then $\kappa=I(\lambda)$.*

Proof. Induction on the structure of the derivation of π. ■

We can now give the theorem which says that the internalisation translation preserves reductions.

Theorem 1.46. *Let π be a modal-logical derivation. If $\pi \rightsquigarrow \tau$, then also $I(\pi) \rightsquigarrow I(\tau)$.*

Proof. By induction on the structure of the derivation of π. If the end-formula of π is the conclusion of an elimination rule that is involved in a reduction, then in the cases $\rightarrow$ and $\Box$ we use Lemma 1.45, and in the latter case we furthermore use the observation that the translation commutes with substitution of nominals for nominals. ■

Preservation of reductions is a desirable property since the application of a reduction rule to a derivation is supposed to leave the identity of the proof represented by the derivation unchanged. Rather, the application of a reduction rule just removes a "detour" in the derivation. See the discussion in Prawitz (1971, p. 257).

From the above we conclude that formulas and derivations of the labelled modal-logical natural deduction system correspond in a natural way to formulas and derivations of the hybrid-logical natural deduction system via the internalisation translation, and moreover, reductions of the labelled modal-logical system correspond naturally to reductions of the hybrid-logical system, as was shown in Theorem 1.46.

1.6.5 Why the Proof-Theory Behaves So Well

To sum up, standard natural deduction systems for modal logic do not satisfy all three criteria given initially in this section, whereas labelled systems usually do satisfy the criteria, but at the expense of making use of metalinguistic machinery. As has been demonstrated in Sect. 1.4, with the hybrid-logical natural deduction system $\mathbf{N}_{\mathcal{H}}$, this deficiency can be remedied by hybridisation, that is, hybridisation of modal logic enables the formulation of a natural deduction system such that the criteria all are satisfied without involving metalinguistic machinery, in particular, rules can be added to the system corresponding to first-order conditions on the accessibility relation expressed by geometric theories.[27]

Which features of hybrid logic have enabled us to formulate natural deduction systems satisfying the three criteria without involving metalinguistic machinery? In technical terms, the answer is that hybrid-logic has the following two features.[28]

---

[27] See Braüner and de Paiva (2006) and Braüner (2005) for the cases of intuitionistic and first-order hybrid logic. In the latter case the accessibility relation as well as the quantifier domains are subject to first-order conditions expressed by geometric theories. See also Braüner (2011).

[28] This can actually be generalized: These two features also enable the formulation of natural deduction systems for intuitionistic hybrid logics satisfying the criterias, cf. Braüner and de Paiva (2006) and Braüner (2011), but in that case the features are interpreted intuitionistically, that is, they are interpreted as statements in intuitionistic first-order logic and intuitionistic hybrid logic. See also Braüner (2006). In the case of first-order hybrid logic, cf. Braüner (2005) and Braüner (2011), we can furthermore express that an individual t exists at a world a, that is, the formula

- We can express that a formula ϕ is true at a world a, that is, the formula $@_a\phi$ is true.
- We can express that a world a is R-related to a world c, that is, the formula $@_a\Diamond c$ is true.

In Sect. 1.6.3 these features enabled us to define the internalisation translation I which translates metalinguistic formulas (a,ϕ) and $R(a,c)$ into hybrid-logical formulas by letting $I((a,\phi)) = @_a\phi$ and $I(R(a,c)) = @_a\Diamond c$. Furthermore, by applying the translation I to all formulas in a derivation, the translation was extended such that it translates a derivation of the labelled modal-logical natural deduction system to a derivation of the hybrid-logical natural deduction system $\mathbf{N}_{\mathcal{H}}$. By considering this translation, we can see why we can formulate a hybrid-logical natural deduction system, namely $\mathbf{N}_{\mathcal{H}}$, that satisfies the three criteria. We consider each of the criteria in turn.

As pointed out in Sect. 1.6.3, the hybrid-logical introduction and elimination rules for the connectives $\wedge$, $\rightarrow$, and $\Box$ can be seen as obtained by taking the image under the translation I of the labelled modal-logical rules for these connectives, that is, rules for reasoning directly about the Kripke models. In the case of the modal operator, this results in the following introduction and elimination rules (cf. Fig. 1.1 of Sect. 1.4.2).

$$\frac{\begin{array}{c}[@_a\Diamond c]\\ \vdots \\ @_c\phi\end{array}}{@_a\Box\phi}\,(\Box I) \qquad\qquad \frac{@_a\Box\phi \qquad @_a\Diamond c}{@_c\phi}\,(\Box E)$$

The introduction rule is equipped with the side-condition that the nominal c does not occur in $@_a\Box\phi$ or in any undischarged assumptions other than the specified occurrences of $@_a\Diamond c$. For each of the connectives $\wedge$, $\rightarrow$, and $\Box$, the hybrid-logical rules satisfy the inversion principle as the labelled rules satisfy it. Besides these connectives, it is straightforward to give introduction and elimination rules for satisfaction operators which satisfy the inversion principle.

The inversion principle gives rise to hybrid-logical reduction rules such that normalisation is satisfied and such that normal derivations satisfy a version of the subformula property, see Sects. 1.4.6 and 1.4.7. Reductions in the labelled modal-logical system correspond to reductions in the hybrid-logical system, as the translation I preserves reductions, cf. Sect. 1.6.4, but there are reduction sequences in the hybrid-logical system involving $\Box$ that do not correspond to any reduction sequences in the labelled modal-logical system, which is a consequence of the fact that the introduction rule for $\Box$ not only exhibits $\Box$ in the conclusion, but

$@_aE(t)$ is true, which enables the formulation of natural deduction systems for first-order hybrid logics satisfying the criteria (here $E(t)$ is the existence predicate which is defined as an abbreviation for $\exists y(y = t)$).

also in the discharged assumptions. The proof of the hybrid-logical normalisation theorem, Theorem 1.13 of Sect. 1.4.6, involves controlling such reduction sequences in a systematic way. See the brief remarks in Theorem 1.13 and see the detailed treatment in the book Braüner (2011).

Conditions on the accessibility relation can be incorporated into the system by adding hybrid-logical rules obtained by taking the image under the translation I of labelled modal-logical natural deduction rules given by Simpson in 1994 (strictly speaking, extended with atomic first-order formulas of the form $a = c$ which are translated to $@_a c$). The conditions on the accessibility relation are first-order conditions expressed by geometric theories. This actually requires the addition of further reduction rules with the aim of removing permutable formulas in a derivation, see Sect. 1.4.6 for more on such permutative reductions.

In conclusion, what has happened is that the metalinguistic formulas and rules of the labelled modal-logical natural deduction system have been internalized as hybrid-logical formulas and rules via the translation I, which has enabled the formulation of an internalized hybrid-logical natural deduction system involving only object language formulas such that the three criteria are satisfied.[29] In other words, we have provided a proof-theoretic analysis demonstrating that the good proof-theoretic behaviour of the labelled rules for reasoning directly about models is preserved by internalisation. We are now in position to give an answer to the initial question of this section: Why does the proof-theory of hybrid logic behave so well? The answer is that internalisation of metalinguistic model-theoretic machinery in the object language enables us to give well-behaved proof-theory for hybrid logic.

The issue of internalizing a metalanguage in a hybrid-logical object language is also discussed in a range of papers by Patrick Blackburn, notably Blackburn (2000a,b). See also the discussion in Sect. 1.4.8.

Moreover, internalizing a metalanguage in an object language is the subject of the paper Seligman (2001). The approach in that paper is, however, different from the approach taken here: In the paper Seligman (2001) a Gentzen system for hybrid logic is developed from a Gentzen system for first-order predicate logic by a series of transformations which step by step internalizes the (first-order) semantic theory of hybrid logic.

[29] In the paper (Brünnler 2006) Kai Brünnler compares labelled and unlabelled Gentzen systems for modal logic. A system of the latter kind is a system that does not use labels, which he makes more precise by calling a Gentzen system *pure* if each sequent has an equivalent object language formula. Clearly, what we here call standard proof systems for modal logic are pure: In natural deduction terminology, a derivation of a modal-logical formula ϕ from a set of modal-logical formulas Γ is equivalent to the modal-logical formula $\bigwedge \Gamma \to \phi$. On the other hand, labelled natural deduction systems for modal logic are clearly not pure, but it is remarkable that hybrid-logical natural deduction systems actually are pure in Brünnler's sense.

1.6.6 Some Concluding Philosophical Remarks

Recall that in Sect. 1.6.2 we pointed out that the labelled introduction and elimination rules for the modal operator can be read off from the modal operator's truth-condition in the Kripke semantics. This also applies to the hybrid-logical, internalized introduction and elimination rules for the modal operator discussed in Sect. 1.6.5. Thus, we have justified the derivation rules for the modal operator by an antecedent understanding of the modal operator's meaning, namely by its truth-condition in the Kripke semantics. Some remarks should be made in this connection.

First note that our justification of the derivation rules for the modal operator is in terms of model-theoretic semantics, namely in terms of the Kripke semantics. This is related to a distinction Jaako Hintikka draws between two different traditions in viewing the relation of logic to reality, namely "language as the universal medium" (or "the universalist tradition") and "language as calculus" (or "the model-theoretical tradition"), see the paper Hintikka (1988). According to the first tradition, one cannot step outside the language and one cannot theorize about changes in the interpretation of the language. Contrary to this, the second tradition includes meta-logical considerations and the tradition takes as a cornerstone the relation between formulas and models defined by Tarski-style truth-conditions. Clearly, our justification of the introduction and elimination rules for the modal operator presupposes the second view since the justification is in terms of a model-theoretic semantics.

An alternative to justifying derivation rules for logical connectives in terms of model-theoretic semantics, is to explain the meaning of the connectives in terms of the roles the connectives play in derivations, independently of model-theoretic notions. It is arguable that both kinds of explanation are legitimate, cf. the following long quotation by Nuel D. Belnap (1962).

> It seems plain that throughout the whole texture of philosophy one can distinguish between two modes of explanation: the analytic mode, which tends to explain wholes in terms of parts, and the synthetic mode, which explains parts in terms of the wholes or contexts in which they occur. In logic, the analytic mode would be represented by Aristotle, who commences with terms as the ultimate atoms, explains propositions or judgements by means of these, syllogisms by means of the propositions which go to make them up, and finally ends with the notion of science as a tissue of syllogisms. The analytic mode is also represented by the contemporary logician who first explains the meaning of complex sentences, by means of truth-tables, as a function of their parts, and then proceeds to give an account of correct inference in terms of the sentences occurring therein. . . . Among formal logicians, use of the synthetic mode in logic is illustrated by Kneale and Popper . . . , as well as by Jaskowski, Gentzen, Fitch and Curry, all of these treating the meaning of connectives as arising from the role they play in the context of formal inference. It is equally well illustrated, I think, by aspects of Wittgenstein and those who learned from him to treat the meanings of words as arising from the role they play in the context of discourse. It seems to me nearly self-evident that employment of both modes of explanation is important and useful (Belnap 1962, pp. 130–131)[30].

[30]Belnap's paper (1962) is a response to Prior's paper (1960) in which Prior raises doubt as to whether the meaning of logical connectives can be explained in terms of derivation rules along the

The programme of explaining the meaning of logical connectives in terms of the roles they play in derivations has developed into a separate branch of logic called *proof-theoretic semantics*. To be more precise, proof-theoretic semantics is based on the idea of explaining the meaning of a logical connective in terms of a set of derivation rules.[31] In certain respects, proof-theoretic semantics is in line with the first tradition identified by Hintikka. We shall not go into further details of proof-theoretic semantics. See the paper Wansing (2000) for a presentation of a number of different semantic paradigms.

References

Alur, R., and T.A. Henzinger. 1989. A really temporal logic. In *Proceedings of the 30th annual symposium on foundations of computer science (FOCS)*, 164–169. Washington, DC: IEEE Computer Society Press.

Alur, R., and T.A. Henzinger. 1992. Logics and models of real time: A survey. In *Real time: Theory in practice*, *volume 600 of lecture notes in computer science*, ed. J.W. de Bakker, C. Huizing, W.P. de Roever, G. Rozenberg, 74–106. Berlin: Springer.

Areces, C. 2000. Logic engineering. The case of description and hybrid logics. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam.

Areces, C., and J. Heguiabehere. 2002. HyLoRes 1.0: Direct resolution for hybrid logics. In *Proceedings of the 18th international conference on automated deduction*, Lecture notes in computer science, vol. 2392, ed. A. Voronkov, 156–160. Berlin: Springer.

Areces, C., and B. ten Cate. 2007. Hybrid logics. In *Handbook of modal logic*, ed. P. Blackburn, J. van Benthem, and F. Wolter, 821–868. Amsterdam: Elsevier.

Areces, C., P. Blackburn, and M. Marx. 1999. The computational complexity of hybrid temporal logics. *Logic Journal of IGPL* 8: 653–679.

Areces, C., P. Blackburn, and M. Marx. 2001a. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic* 66: 977–1010.

Areces, C., M. de Rijke, and H. de Nivelle. 2001b. Resolution in modal, description and hybrid logic. *Journal of Logic and Computation* 11: 717–736.

lines of natural deduction introduction and elimination rules. In his paper, Prior introduces a logical connective *tonk* with introduction rules similar to the standard natural deduction introduction rules for disjunction and elimination rules similar to the standard natural deduction elimination rules for conjunction. An effect of extending a formal system with *tonk* together with the mentioned rules is that any formula becomes derivable, which obviously is absurd. In his response to Prior's paper, Belnap suggests imposing certain restrictions on derivation rules, thereby excluding Prior's rules for *tonk* from the permissible rules for a connective. According to Belnap, the crucial restriction is *conservativity*: When a formal system is extended with a new logical connective together with a set of derivation rules, then for any formula built using only the original connectives, it is required that if the formula is derivable in the extended system, then it is also derivable in the original system, that is, it is derivable without using the derivation rules for the new connective.

[31] A frequently discussed issue in proof-theoretic semantics is which restrictions to impose on the derivation rules for a connective, that is, which sets of derivation rules to take as permissible. This discussion can be traced back to Prior (1960) and Belnap's (1962) papers, cf. the previous footnote. A number of restrictions on derivation rules have been proposed, one proposal being conservativity, cf. the previous footnote, another proposal being the inversion principle (cf. the paper Prawitz (1971)).

Areces, C., P. Blackburn, and M. Marx. 2003. Repairing the interpolation theorem in quantified modal logic. *Annals of Pure and Applied Logic* 124: 287–299.

Avron, A. 1996. The method of hypersequents in the proof theory of propositional non-classical logics. In *Logic: Foundations to applications*, ed. W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, 1–32. Oxford: Oxford Science Publications.

Basin, D., S. Matthews, and L. Viganò. 1997. Labelled propositional modal logics: Theory and practice. *Journal of Logic and Computation* 7: 685–717.

Belnap, N.D. 1962. Tonk, plonk and plink. *Analysis* 22: 130–134.

Blackburn, P., and M. Marx. 2003. Constructive interpolation in hybrid logic. *Journal of Symbolic Logic* 68: 463–480.

Blackburn, P., and J. Seligman. 1995. Hybrid languages. *Journal of Logic, Language and Information* 4: 251–272.

Blackburn, P., and B. ten Cate. 2006. Pure extensions, proof rules, and hybrid axiomatics. *Studia Logica* 84: 277–322.

Blackburn, P., and M. Tzakova. 1998. Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence* 24: 23–49.

Blackburn, P., and M. Tzakova. 1999. Hybrid languages and temporal logic. *Logic Journal of IGPL* 7: 27–54.

Blackburn, P., M. de Rijke, and Y. Venema. 2001. *Modal logic*, Cambridge tracts in theoretical computer science, vol. 53. Cambridge: Cambridge University Press.

Blackburn, P. 1993. Nominal tense logic. *Notre Dame Journal of Formal Logic* 14: 56–83.

Blackburn, P. 2000a. Internalizing labelled deduction. *Journal of Logic and Computation* 10: 137–168.

Blackburn, P. 2000b. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of IGPL* 8: 339–365.

Blackburn, P. 2006. Arthur Prior and hybrid logic. *Synthese* 150: 329–372 (Special issue edited by T. Braüner, P. Hasle, and P. Øhrstrøm).

Bolander, T., and P. Blackburn. 2007. Termination for hybrid tableaus. *Journal of Logic and Computation* 17: 517–554.

Bolander, T., and P. Blackburn. 2009. Terminating tableau calculi for hybrid logics extending K. In *Proceedings of methods for modalities 5*, ed. C. Areces and S. Demri, Electronic notes in theoretical computer science, vol. 231, 21–39. Amsterdam: Elsevier.

Bolander, T., and T. Braüner. 2005. Two tableau-based decision procedures for hybrid logic. In *4th Workshop "Methods for Modalities" (M4M), Informatik-Bericht Nr. 194*, ed. H. Schlingloff, 79–96. Berlin: Humboldt-Universität zu Berlin.

Bolander, T., and T. Braüner. 2006. Tableau-based decision procedures for hybrid logic. *Journal of Logic and Computation* 16: 737–763. Revised and extended version of (Bolander and Braüner, 2005).

Boolos, G. 1984. Don't eliminate cut. *Journal of Philosophical Logic* 13: 373–378.

Braüner, T., and V. de Paiva. 2006. Intuitionistic hybrid logic. *Journal of Applied Logic* 4: 231–255.

Braüner, T. 2002b. Modal logic, truth, and the master modality. *Journal of Philosophical Logic* 31: 359–386.

Braüner, T. 2004a. Natural deduction for hybrid logic. *Journal of Logic and Computation* 14: 329–353.

Braüner, T. 2004b. Two natural deduction systems for hybrid logic: A comparison. *Journal of Logic, Language and Information* 13: 1–23.

Braüner, T. 2005. Natural deduction for first-order hybrid logic. *Journal of Logic, Language and Information* 14: 173–198.

Braüner, T. 2006. Axioms for classical, intuitionistic, and paraconsistent hybrid logic. *Journal of Logic, Language and Information* 15: 179–194.

Braüner, T. 2007. Why does the proof-theory of hybrid logic work so well? *Journal of Applied Non-Classical Logics* 17: 521–543.

Braüner, T. 2011. *Hybrid logic and its proof-theory*, Applied logic series, vol. 37. Berlin: Springer.

Broda, K., M. Finger, and A. Russo. 1999. Labelled natural deduction for substructural logics. *Logic Journal of IGPL* 7: 283–318.

Brünnler, K. 2006. Deep sequent systems for modal logic. In *Advances in modal logic*, vol. 6, ed. G. Governatori, I. Hodkinson, and Y. Venema, 117–119. London: College Publications.

Bull, R.A. 1970. An approach to tense logic. *Theoria* 36: 282–300.

Bull, R.A., and K. Segerberg. 2001. Basic modal logic. In *Handbook of philosophical logic*, vol. 3, 2nd ed, ed. D.M. Gabbay and F. Guenthner, 1–81. Dordrecht: Kluwer Academic Publishers.

Chang, C.C., and H.J. Keisler. 1990. *Model theory*, 3rd ed. Amsterdam: Elsevier.

Copeland, J. (ed.). 1996. *Logic and reality: Essays in the legacy of Arthur Prior*. Oxford: Oxford University Press/Clarendon Press.

Copeland, J. 2007. Arthur Prior. In *The Stanford encyclopedia of philosophy*, ed. E.N. Zalta. Stanford: Stanford University. On-line encyclopedia article available at http://plato.stanford.edu/entries/prior.

D'Agostino, M., and M. Mondadori. 1994. The taming of the cut. Classical refutations with analytical cut. *Journal of Logic and Computation* 4: 285–319.

D'Agostino, M., D.M. Gabbay, R. Hähnle, and J. Posegga (eds.). 1999. *Handbook of tableau methods*. Berlin: Springer.

Donini, F.M., and F. Massacci. 2000. Exptime tableaux for ALC. *Artificial Intelligence* 124: 87–138.

Fine, K., and A.N. Prior. 1977. *Worlds, times and selves*. London: Duckworth. Based on manuscripts by Prior with a preface and a postscript by K. Fine.

Fitting, M. 1972a. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic* 13: 237–247.

Fitting, M. 1972b. ε-calculus based axiom systems for some propositional modal logics. *Notre Dame Journal of Formal Logic* 13: 381–384.

Fitting, M. 1983. *Proof methods for modal and intuitionistic logic*. Dordrecht: Reidel.

Fitting, M. 1984. Linear reasoning in modal logic. *Journal of Symbolic Logic* 49: 1363–1378.

Fitting, M. 1992a. Many-valued modal logics. *Fundamenta Informaticae* 15: 235–254.

Fitting, M. 1992b. Many-valued modal logics II. *Fundamenta Informaticae* 17: 55–73.

Fitting, M. 1995. Tableaus for many-valued modal logic. *Studia Logica* 55: 63–87.

Fitting, M. 2007. Modal proof theory. In *Handbook of modal logic*, ed. P. Blackburn, J. van Benthem, and F. Wolter, 85–138. Amsterdam: Elsevier.

Franceschet, M., and M. de Rijke. 2006. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic* 4: 279–304.

Gabbay, D.M. 1996. *Labelled deductive systems*. Oxford: Oxford University Press.

Gabbay, D.M., and G. Malod. 2002. Naming worlds in modal and temporal logic. *Journal of Logic, Language and Information* 11: 29–65.

Galton, A. 2006. Operators vs. arguments: The ins and outs of reification. *Synthese* 150: 415–441 (Special issue edited by T. Braüner, P. Hasle, and P. Øhrstrøm).

Gargov, G., and V. Goranko. 1993. Modal logic with names. *Journal of Philosophical Logic* 22: 607–636.

Gentzen, G. 1969. Investigations into logical deduction. In *The collected papers of Gerhard Gentzen*, ed. M.E. Szabo, 68–131. Amsterdam: North-Holland Publishing Company.

Goranko, V. 1994. Temporal logic with reference pointers. In *Proceedings of the 1st international conference on temporal logic* Lecture notes in artificial intelligence, vol. 827, 133–148. Berlin: Springer.

Goranko, V. 1996. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information* 5: 1–24.

Goranko, V. 2000. *Sorting and hybrid modal logics*. Manuscript.

Goré, R. 1999. Chapter 6: Tableau methods for modal and temporal logics. In *Handbook of tableau methods*, ed. M. D'Agostino, D. Gabbay, R. Haehnle, and J. Posegga, 297–396. Dordrecht: Kluwer Academic Publishers.

Hansen, J.U. 2007. A tableau system for a first-order hybrid logic. In *Proceedings of the international workshop on hybrid logic 2007*, ed. J. Villadsen, T. Bolander, and T. Braüner, 32–40. *19th European Summerschool in Logic, Language and Information*. Trinity College: Dublin.

Hansen, J.U. 2010. Terminating tableaux for dynamic epistemic logics. In *Proceedings of the 6th workshop on methods for modalities (M4M-6 2009)*, Electronic notes in theoretical computer science, vol. 262, ed. T. Bolander and T. Braüner, 141–156. Amsterdam: Elsevier.

Hansen, J.U., T. Bolander, and T. Braüner. 2008. Many-valued hybrid logic. In *Advances in modal logic*, vol. 7, ed. C. Areces and R. Goldblatt, 111–132. London: College Publications.

Hardt, M., and G. Smolka. 2007. Higher-order syntax and saturation algorithms for hybrid logic. In *Proceedings of the international workshop on hybrid logic 2006*, Electronic notes in theoretical computer science, vol. 174, ed. P. Blackburn, T. Bolander, T. Braüner, V. de Paiva, and J. Villadsen, 15–27. Amsterdam: Elsevier.

Hasle, P., P. Øhrstrøm, T. Braüner, and J. Copeland. 2003. *Revised and expanded edition of Arthur N. Prior: Papers on time and tense*. Oxford: Oxford University Press.

Hintikka, J. 1955. Form and content in quantification theory. *Acta Philosophica Fennica* 8: 8–55.

Hintikka, J. 1988. On the development of the model-theoretic viewpoint in logical theory. *Synthese* 77: 1–36.

Horrocks, I., and U. Sattler. 2005. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proceedings of the 19th international joint conference on artificial intelligence (IJCAI 2005)*, ed. L.P. Kaelbling and A. Saffiotti, 448–453.

Horrocks, I., U. Hustadt, U. Sattler, and R. Schmidt. 2007. Computational modal logic. In *Handbook of modal logic*, ed. P. Blackburn, J. van Benthem, and F. Wolter, 181–245. Amsterdam: Elsevier.

Hughes, G.E., and M.J. Cresswell. 1968. *An introduction to modal logic*. York: Methuen.

Kaminski, M., and G. Smolka. 2007. Hybrid tableaux for the difference modality. In *Workshop Proceedings of methods for modalities 5*, ed. Carlos Areces and Stéphane Demri, 269–284. Paris: École Normale Supérieure de Cachan.

Kaminski, M., and G. Smolka. 2009. Terminating tableau systems for hybrid logic with difference and converse. *Journal of Logic, Language and Information* 18: 437–464 (Special issue edited by T. Braüner and T. Bolander).

Kenny, A.J.P. 1970. Arthur Norman Prior (1914–1969). *Proceedings of the British Academy* LVI: 321–349.

Lange, M. 2009. Model checking for hybrid logic. *Journal of Logic, Language and Information* 18: 465–491 (Special issue edited by T. Braüner and T. Bolander).

Massacci, F. 2000. Single step tableaux for modal logics. *Journal of Automated Reasoning* 24: 319–364.

Mayer, M.C., and S. Cerrito. 2010. Nominal substitution at work with the global and converse modalities. Nominal substitution at work with the global and converse modalities. In *Advances in modal logic*, vol. 8, ed. L. Beklemishev, V. Goranko, and V. Shehtman, 57–74. London: College Publications.

McTaggart, J.M.E. 1908. The unreality of time. *Mind* 187: 457–474.

Müller, T. 2007. Prior's tense-logical universalism. *Logique et Analyse* 50: 223–252.

Øhrstrøm, P., and P. Hasle. 1993. A.N. Prior's rediscovery of tense logic. *Erkenntnis* 39: 23–50.

Øhrstrøm, P., and P. Hasle. 1995. *Temporal logic: From ancient ideas to artificial intelligence*. Dordrecht: Kluwer Academic Publishers.

Øhrstrøm, P., and P. Hasle. 2005a. A.N. Prior's logic. In *Logic and the modalities in the twentieth century*, The handbook of the history of logic, vol. 6, ed. D.M. Gabbay and J. Woods. Amsterdam: Elsevier.

Øhrstrøm, P., and P. Hasle. 2005b. Modern temporal logic: The philosophical background. In *Logic and the modalities in the twentieth century*, The handbook of the history of logic, vol. 6, ed. D.M. Gabbay and J. Woods. Amsterdam: Elsevier.

Passy, S., and T. Tinchev. 1985. Quantifiers in combinatory PDL: Completeness, definability, incompleteness. In *Fundamentals of computation theory FCT 85*, Lecture notes in computer science, vol. 199, 512–519. Berlin: Springer.

Passy, S., and T. Tinchev. 1991. An essay in combinatory dynamic logic. *Information and Computation* 93: 263–332.

Prawitz, D. 1965. *Natural deduction: A proof-theoretical study*. Stockholm: Almqvist and Wiksell.

Prawitz, D. 1971. Ideas and results in proof theory. In *Proceedings of the second Scandinavian logic symposium*, Studies in logic and the foundations of mathematics, vol. 63, ed. J.E. Fenstad, 235–307. Amsterdam: North-Holland Publishing Company.

Prawitz, D. 1978. Proofs and the meaning and completeness of the logical constants. In *Essays on mathematical and philosophical logic*, Studies in logic and the foundations of mathematics, ed. J. Hintikka et al., 25–40. Dordrecht: Reidel.

Prior, A.N. 1960. The runabout inference-ticket. *Analysis* 21: 38–39.

Prior, A.N. 1967. *Past, present and future*. Oxford: Clarendon Press/Oxford University Press.

Prior, A.N. 1968. *Papers on time and tense*. Oxford: Clarendon Press/Oxford University Press.

Prior, A.N. 1996. Some free thinking about time. In *Logic and reality: Essays in the legacy of Arthur Prior*, ed. J. Copeland. Oxford: Oxford University Press/Clarendon Press (With introduction by P. Øhrstrøm in which the original dating is discussed, 43–44 and 47–51).

Seligman, J. 1997. The logic of correct description. In *Advances in intensional logic*, Applied logic series, ed. M. de Rijke, vol. 7, 107–135. Dordrecht: Kluwer Academic Publishers.

Seligman, J. 2001. Internalisation: The case of hybrid logics. *Journal of Logic and Computation* 11: 671–689 (Special Issue on Hybrid Logics. C. Areces and P. Blackburn (eds.)).

Simons, P. 2006. The logic of location. *Synthese* 150: 443–458 (Special issue edited by T. Braüner, P. Hasle, and P. Øhrstrøm).

Simpson, A. 1994. The proof theory and semantics of intuitionistic modal logic. PhD thesis, University of Edinburgh.

Sustretov, D. 2009. Hybrid logics of separation axioms. *Journal of Logic, Language and Information* 18: 541–558 (Special issue edited by T. Braüner and T. Bolander).

Sylvan, R. 1996. Other withered stumps of time. In *Logic and reality: Essays in the legacy of Arthur Prior*, ed. J. Copeland, 111–130. Oxford: Oxford University Press/Clarendon Press.

ten Cate, B. 2004. Model theory for extended modal languages. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam.

ten Cate, B., and T. Litak. 2007. Topological perspective on the hybrid proof rules. In *Proceedings of the international workshop on hybrid logic 2006*, Electronic notes in theoretical computer science, vol. 174, ed. P. Blackburn, T. Bolander, T. Braüner, V. de Paiva, and J. Villadsen, 79–94. Amsterdam: Elsevier.

Troelstra, A.S., and H. Schwichtenberg. 1996. *Basic proof theory*, Cambridge tracts in theoretical computer science, vol. 43. Cambridge: Cambridge University Press.

Tzakova, M. 1999. Tableaux calculi for hybrid logics. In *Automated reasoning with analytic tableaux and related methods (TABLEAUX 1999)*, Lecture notes in artificial intelligence, vol. 1617, ed. N.V. Murray, 278–292. New York: Springer.

van Benthem, J. 1983. *Modal logic and classical logic*. Napoli: Bibliopolis.

Vickers, S. 1988. *Topology via logic, volume 5 of Cambridge tracts in theoretical computer science*. Cambridge: Cambridge University Press.

Viganò, L. 2000. *Labelled non-classical logics*. Dordrecht: Kluwer Academic Publishers.

Wansing, H. 1994. Sequent calculi for normal modal propositional logics. *Journal of Logic and Computation* 4: 125–142.

Wansing, H. 2000. The idea of a proof-theoretic semantics and the meaning of the logical operations. *Studia Logica* 64: 3–20.

Chapter 2
Nominal Terms and Nominal Logics: From Foundations to Meta-mathematics

Murdoch J. Gabbay

2.1 Introduction

Nominal sets for meta-mathematics Suppose we want to axiomatise the λ-calculus or first-order logic. Then we need to express properties like this:

- If $y \not\in fv(t)$ then $\forall x.t =_\alpha \forall y.(t[y/x])$.
- If $x \not\in fv(u)$ then $(\lambda x.t)[u/y] = \lambda x.(t[u/y])$.
- If $x \not\in fv(t)$ then $\lambda x.(tx) =_\eta t$.

x, y, t, and u here are what we would call *names*. A linguist might call them *referents*, a mathematician might call them *variables*. But the words 'referent' and 'variable' carry connotations (a referent should refer to something, a variable should vary), so we prefer the more neutral term 'name'. So for us, a name is just an atomic symbol, to which we may then associate further properties, at our discretion, using additional axioms.

The axioms above are typical of a certain kind of specification. Mathematical specification is nothing new. First-order logic can specify, to choose a classic trio of examples, groups, rings, and fields. But the λ-calculus, first-order logic itself, the π-calculus, and a very great many other examples, are different. They have *names*.

By adding names to first-order logic in the correct way, we can axiomatise the specifications above, cleanly and in a manner very close to the informal specification. How should we do this? Using a recent application of mathematical foundations originating in computer science: *nominal sets* Gabbay and Pitts (2001), Gabbay (2011b), to which we will use *nominal terms* Urban et al. (2003, 2004) as a corresponding formal syntax. To survey and update the state of the art of logics based on nominal terms and taking semantics in nominal sets, is our goal here.

M.J. Gabbay (✉)
School of Mathematical and Computer Sciences, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, United Kingdom
http://www.gabbay.org.uk

D.M. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic: Volume 17*, Handbook of Philosophical Logic 17, DOI 10.1007/978-94-007-6600-6_2,

In nominal terms, term-formers can bind names and freshening renamings like the $[y/x]$ or $[u/y]$ above are taken as primitive.

Here are the informal statements above, rewritten in permissive-nominal algebra—an algebraic logic based on nominal terms with a sound and complete semantics in nominal sets:

- If $b \notin supp(X)$ then $\forall([a]X) = \forall([b](b\ a)\cdot X)$.
- If $a \notin supp(Y)$ then $\lambda([a]X)[b{\mapsto}Y] = \lambda([a](X[b{\mapsto}Y]))$.
- If $a \notin supp(X)$ then $\lambda([a](Xa)) = X$.

In this chapter we will briefly consider nominal sets, then survey nominal terms, unification, rewriting, algebra, and permissive-nominal logic. We cover the nominal unification algorithm, confluence proofs for nominal rewriting, soundness and completeness results for nominal algebra and permissive-nominal logic, an HSP theorem, and a finite axiomatisation of first-order logic.

By doing this we aim to give an overview of the applications of nominal sets to meta-mathematical syntax. We cannot be exhaustive, but we can try to be representative of what can be achieved.

As we shall see, nominal syntax is more expressive than first-order syntax (for instance we can give a finite first-order axiomatisation of arithmetic), because term-formers that can explicitly manipulate names. Yet, it remains first-order in flavour, preserving theoretical and computational properties like completeness and most general unifiers.

A few words on atoms What nominal sets add to 'ordinary' structures is an assumption of a distinguished class of symmetric atomic elements called *atoms*: these are also called *urelemente* or *names*. We will use these terms more-or-less synonymously.

Indeed, nominal sets are a special case Zermelo-Fraenkel sets with atoms, and are instances of the structures considered by Fraenkel and Mostowski in their celebrated independence proof of the Axiom of Choice from the other axioms of set theory with atoms. For detailed references see (Gabbay 2011b, Remark 2.22). So this chapter really does describe a journey from mathematical foundations to meta-mathematics, and that is representative of how the maths we describe here was arrived at.

We can view the underlying philosophy of nominal techniques is as the following informal inequality, where 'smaller' means 'greater generality':

$$\text{atoms} = \text{urelemente} = \text{names} \leq \text{referents} \leq \text{variables}$$

Discovering to what extent these intuitions can be made precise, concrete, and useful, is the topic of much ongoing research, some of which is reported on here.

Names induce automorphisms generated by permuting them. We shall see that if we model variables as a special case of atoms, then α-renaming becomes a special case of a much more general fact that nominal sets are symmetric under permuting atoms. This generalisation turns out to have powerful consequences, including the atoms-abstraction and $\mathcal{N}$-quantifier introduced by the author with Pitts in Gabbay

and Pitts (2001). So the point of view described above has led to and continues to lead to new reasoning principles.

If we identify a thing with the properties of that thing, then the 'nominal' model suggests that names are equal to the following set of three properties:

$$\text{names} = \{\text{atomic}, \text{symmetric}, \text{generative}\}$$

The reader familiar with nominal techniques can identify these three properties with the use of: atomic symbols a (an atom, name, or urelement, with a distinct existence in the denotation), permutations π (symmetries under permutation of names), and the N-quantifier ('choose a fresh name'). These three properties will appear directly in this chapter as atoms, permutations, and permission sets.[1] Full definitions appear below.

This material in the literature This paper surveys existing literature on logics based on nominal terms, and adds a few new results. Very broadly, Sect. 2.2.1 is based on Gabbay and Pitts (1999, 2001) (nominal sets; they were called *equivariant FM sets* there); Sects. 2.2.2 and 2.3.1 are based on Urban et al. (2003, 2004), Dowek et al. (2009, 2010), Gabbay (2012a) (nominal terms and unification); Sects. 2.3.2 and 2.3.3 are based on Fernández et al. (2004), Fernández and Gabbay (2007), Gabbay (2012a) (rewriting and closed terms); Sect. 2.3.4 is based on Gabbay (2005), Gabbay and Mathijssen (2006a, 2007, 2009) (nominal algebra); Sects. 2.4.1, 2.4.2, and 2.4.3 are based on Dowek and Gabbay (2010, 2012a) (permissive-nominal logic).

Definitions and proofs may have changed from the original presentations. In particular:

- The semantics is *permissive*-nominal, meaning that it is based on possibly infinitely supported nominal sets with co-infinite support. In Gabbay and Pitts (2001) a nominal semantics based on finite and co-infinite support was used.
- Unlike Urban et al. (2004) and Dowek et al. (2010) we use nominal abstract syntax to build our nominal terms. That is, in this paper nominal terms atoms-abstraction is directly equal to Gabbay-Pitts atoms-abstraction. Thus, nominal terms here are an instance of nominal abstract syntax and come quotiented by α-equivalence by construction.
- Permutation may be stronger than usual, and we parameterise over the group of permutations.

  We consider (as usual) finite permutations (generated by *swappings*, also called *transpositions*) as standard, but in particular we also find *shift*-permutations δ useful, which shift infinitely many atoms. The *shift*-permutation δ corresponds to a de Bruijn shift function $\uparrow$ and presheaf reindexing map up, though δ is not equal to them since it is a permutation and so invertible.

[1] In other papers, such as Urban et al. (2004), permission sets are presented instead as syntactic freshness assumptions.

- Syntax includes non-equivariant constant symbols. In Urban et al. (2004) all term-formers/function-symbols (including 0-ary ones, i.e. constants) were equivariant. This does not matter for finite support but it does make a difference with infinite support.
- Nominal unknowns are modelled as arbitrary elements of a strongly-supported nominal set. This means that the X and Y in this paper correspond to *moderated unknowns* from Urban et al. (2004): see Example 2.45.
- Because unknowns have support, there are no freshness contexts and substitutions are characterised as equivariant functions (the freshness conditions normally attached to substitutions follow from equivariance: see Proposition 2.64). The theories of nominal unification, rewriting, and algebra are reformulated to reflect this.
- The simplification rules for unification problems (Fig. 2.2) are new and the treatments of closed terms and closed nominal rewriting (Sect. 2.3.3) are entirely revised with respect to Fernández and Gabbay (2007).

2.2 Nominal Sets and Nominal Terms

2.2.1 Nominal Sets

We open with a brief presentation of nominal sets, which are the semantic basis for this work: this is the universe that the logics we define will describe, and be sound (and complete) for.

Nominal sets were developed with Pitts and introduced in the author's thesis Gabbay (2001), a conference paper Gabbay and Pitts (1999), and journal paper Gabbay and Pitts (2001). The nominal sets here are more general than in Gabbay and Pitts (2001): following Dowek et al. (2010) we are *permissive*, meaning that we split the set of atoms into two infinite halves and consider infinite support. This specific idea was developed jointly with Dowek,[2] but shades of it appear also in Cheney's paper Cheney (2006) and in the author's study of infinite atoms-abstraction Gabbay (2007b).

In addition we parameterise over a group of permutations which need not just be finitely-supported permutations. This is new.

2.2.1.1 Atoms, Permutations, Permission Sets

In Definition 2.2 we need several sets of atoms. This is to model the several sorts of names that will appear in our syntax later on.

[2]The development here is a little different from that in Dowek et al. (2010) because we take permission sets to be sets of the form $\pi \cdot \mathbb{A}^<$ instead of sets of the form $(\mathbb{A}^< \setminus A) \cup B$.

Following Dowek et al. (2010) our development will be *permissive*-nominal. A permission set S splits a set of atoms into two halves $\mathbb{A}^<$ and $\mathbb{A}^>$. One intuition for $\mathbb{A}^<$ is 'the atoms that have been generated so far', and for $\mathbb{A}^>$ is 'the atoms that might be generated later'.

Definition 2.1. Write $\mathbb{N} = \{0,1,2,3,\ldots\}$ for the natural numbers and $\mathbb{Z} = \{0,\text{-}1,1,\text{-}2,2,\ldots\}$ for the integers.

Definition 2.2. For each $i \in \mathbb{N}$ fix a pair of disjoint countably infinite sets of **atoms** $\mathbb{A}^<_i$ and $\mathbb{A}^>_i$.

Write

$$\mathbb{A}_i = \mathbb{A}^< \uplus \mathbb{A}^> \qquad \mathbb{A}^< = \bigcup \mathbb{A}^<_i \qquad \mathbb{A}^> = \bigcup \mathbb{A}^>_i \qquad \mathbb{A} = \bigcup \mathbb{A}_i$$

$a,b,c,\ldots$ will range over *distinct* atoms: we call this the **permutative** convention.

Remark 2.3 (Comments on splitting the set of atoms). The different sets of atoms $\mathbb{A}_i$ are different 'types' of atoms. Thus, later on in Definitions 2.39 and 2.49 we can give each name sort its own distinct population of atoms.

The reasons for splitting the set of atoms into $\mathbb{A}^<$ and $\mathbb{A}^>$ will become clear as the maths develops. It might help to think of $\mathbb{A}^<$ as 'atoms that can be captured' and of $\mathbb{A}^>$ as 'atoms that cannot be captured', or as 'atoms that might have been generated in the past' and 'atoms that may be generated in the future'—but with reservations. In Definition 2.10 we see that this is only true up to permuting atoms.

The real purpose of Definition 2.2 is to ensure that we have plenty—countably infinitely many—of 'capturable' and 'non-capturable' atoms. Permutations (below) can and will move atoms between these worlds, but no permutation can move them *all at once*. So the interest of $\mathbb{A}^<$ is not just for the set itself but for its orbit under permutations; this is a property of the set as a whole, and not of its individual elements.

*Remark 2.4 (*Comments on the permutative convention*).* While visiting Tel-Aviv University in 2006 I gave talks on nominal techniques and Arnon Avron asked: "Do a and b refer to specific atoms (e.g. in the axioms in the Introduction), or to any two atoms?". In other words, are a and b constants or variables?

In response I started using a *permutative convention* that a and b are variables, but they range over distinct atoms so that variables with distinct names refer to distinct objects (the first uses were in Gabbay and Mathijssen (2006c,a); the convention was explicitly named in Gabbay and Mathijssen (2008c)).

For a while this was resisted by some anonymous referees. Yet, we typically apply the permutative convention informally; e.g. we silently assume that $\lambda x.\lambda y.xy$ is never the same term as $\lambda x.\lambda x.xx$. I would claim that the permutative convention expresses something about the foundational origins of the nominal view of names as *urelemente*—constants that are distinguishable yet symmetric—in an underlying set theory.

Perhaps this is why the referees did not like it: the permutative convention may seem unnatural if we are committed to standard (nameless) Zermelo-Fraenkel foundations, since names are then just some set, and like any set should be varied

over non-permutatively by variables. Thus the fact that we accept that $\lambda x.\lambda y.xy$ and $\lambda x.\lambda x.xx$ always signify distinct λ-terms to us, can be taken as a sign that we inhabit a nameful foundation, so that the permutative convention is a signpost on the way to something more extensive.

A formal reflection of the permutative convention appears explicitly in the formal logics of this paper: it lives in the π of the $\pi{\cdot}X$ in Definition 2.125.

Definition 2.5. Given $a,b \in \mathbb{A}_i$ for some $i \in \mathbb{N}$ write $(a\ b)$ for the **swapping** bijection on atoms mapping a to b, b to a, and any other $c \in \mathbb{A} \setminus \{a,b\}$ to c.

Another standard name for a swapping is a **transposition**.

By convention $(a\ a)$ will denote the **identity** function on atoms id.

If π is a bijection on atoms define

$$nontriv(\pi) = \{a \mid \pi(a) \neq a\}.$$

Definition 2.6. A **nominal permutation group** is any set of bijections $\mathbb{P}$ of $\mathbb{A}$ such that:

1. If $a \in \mathbb{A}_i$ and $b \in \mathbb{A}_i$ then $(a\ b) \in \mathbb{P}$.
2. If $\pi \in \mathbb{P}$ then $a \in \mathbb{A}_i$ if and only if $\pi(a) \in \mathbb{A}_i$.
3. There exists some infinite $S \subseteq \mathbb{A}$ such that $nontriv(\pi) \cap S$ is finite for every $\pi \in \mathbb{P}$.

Call a bijection on atoms π a **finite permutation** when it is in the subgroup generated by swappings. (π is finite when $\pi(a) \in \mathbb{A}_i$ if and only if $a \in \mathbb{A}_i$ and $nontriv(\pi)$ is finite.)

Write $\pi \circ \pi'$ for the **composition** of π and π' (so $(\pi \circ \pi')(a) = \pi(\pi'(a))$). Write id for the **identity** permutation (so $id(a) = a$ always).

The purpose of conditions 1–3 of Definition 2.6 are as follows:

1. Swappings make sure we can always rename a to b (and b to a).
2. Condition 2 is a standard typing condition, that we do not try to turn an atom of one sort, into an atom of another sort.
3. This condition guarantees that we can still always choose a fresh atom for any finite set of permutations (see for instance Lemma 2.57).

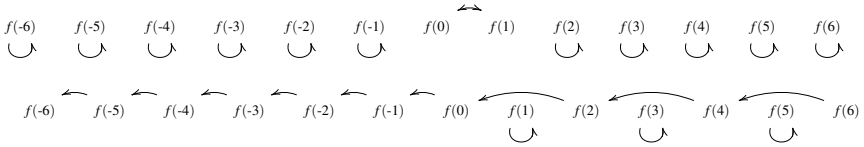
Example 2.7. 1. The set of all finite permutations is a nominal permutation group.

2. For each i fix a bijection f_i between $\mathbb{A}_i$ and the integers $\mathbb{Z}$, such that $\{f(i) \mid i \leq 0\} = \mathbb{A}_i^{\leq}$ and (consequently) $\{f(i) \mid i > 0\} = \mathbb{A}_i^{>}$. We can do this because we assumed atoms are countable.

   Write δ_i for the permutation mapping

   - $f_i(j)$ to $f_i(j-1)$ for $j \leq 0$,
   - $f_i(2j)$ to $f_i(2(j-1))$ and $f_i(2j-1)$ to $f_i(2j-1)$ for $j \geq 1$, and
   - any other $c \in \mathbb{A} \setminus \mathbb{A}_i$ to c.

   This is an example of a *shift*-permutation, considered in more generality in Definition 2.79 and throughout Sect. 2.2.2.6. We illustrate fragments of the actions of a swapping $(f(0)\ f(1))$ and a δ_i:



The atoms corresponding to positive odd integers are taken to be fixed points of δ_i in order to satisfy condition 3 of Definition 2.6, so that these atoms can be taken fresh for δ_i if we need to.

The set of permutations generated by swappings and δ_i, is a nominal permutation group.

Remark 2.8. The nominal permutation group $\mathbb{P}$ determines the symmetries of our nominal syntax and semantics. We consider permutations designed to guarantee (in Definition 2.14) symmetry up to equality/inequality of atoms. We will get sets with atoms that are atomic, symmetric (up to equality and inequality of names), and generative—the main further design choice we care about is whether or not to include a *shift* (Example 2.7), which goes strictly beyond what can be achieved with finite permutations as considered e.g. in Gabbay and Pitts (2001).

Other notions of permutation may lead to other symmetries, so an interesting topic of future research is to weaken the conditions in Definition 2.6.

For instance, if we only allow permutations generated by $f(i) \mapsto f(i+1)$ and $f(i) \mapsto f(\text{-}i)$ then we preserve a notion of 'distance' between atoms.[3] In a similar vein, we can identify atoms with points in a plane and consider Euclidian transformations. It is not known how much of 'nominal techniques' would hold of such examples.

More generally of course, presheaves are a forum within which sets with symmetry structure can be expressed. Indeed, nominal sets can be viewed as a category of presheaves Gabbay and Pitts (1999) and a similar presheaf category was considered at the same time Fiore et al. (1999) (see also the later related *nominal renaming sets* Gabbay and Hofmann (2008), which are in some sense half-way in between those two systems).

There is no shortage of research into this kind of structure Mac Lane and Moerdijk (1992). It remains, however, to understand what are the abstract properties that make a set with a group action, or a presheaf, into something 'nominal'.

Definition 2.9. If $A \subseteq \mathbb{A}$ define the **pointwise** action by

$$\pi{\cdot}A = \{\pi(a) \mid a \in A\}.$$

Definition 2.10. A **permission set** S is a set of the form $\pi{\cdot}\mathbb{A}^{\leq}$.
S, T will range over permission sets.

[3]This example modified from an example by Bartek Klin; private communication from Alexander Kurz.

Remark 2.11. Some preliminary comments on permission sets:

- The notion of permission set used in some previous work, for instance in (Dowek et al. 2010, Definition 2.2), was slightly different: a permission set was taken to be a set of the form $(S \setminus A) \cup B$ for finite $A \subseteq \mathbb{A}^<$ and $B \subseteq \mathbb{A}^>$. In the presence of *shift*-permutations we can do this using a permutation, and any $(S \setminus A) \cup B$ can be written as $\pi \cdot S$ for suitable π (cf. Remark 2.80 and $\delta_{X\text{-}a} \cdot X$ in (**IF**) of Fig. 2.2).

  Given that the designs are equivalent for the cases we will care about, we chose Definition 2.10 because it is somewhat simpler to do mathematics with.
- In the semantics, permission sets are used in the definition of support Definition 2.14; if permutations specify *symmetry*, permission sets specify *capturability* and *generativity* (Remark 2.15).
- In the syntax, permission sets are used to control capture (see Remark 2.70); atoms in S are intuitively 'capturable' and atoms not in S are intuitively 'not capturable'.

  This is reminiscent of some treatments of syntax where a formal distinction is made between 'names that exist to be bound' and 'names that exist to be free'. See for instance the *freie* and *gebundene Gegenstansvariable* of Gentzen (1935, Section 1), and the *individual variables* and *parameters* of Prawitz (1965, Section 1), or Smullyan (1968, [Chapter IV, Section 1]).

  However, note that here, for any $a \in S$ and $b \notin S$, also $a \notin (b\ a) \cdot S$ and $b \in (b\ a) \cdot S$. That is, for any given atom there is no fixed sense in which it is capturable or not capturable. Each individual permission sets defines its own world of capturable/non-capturable atoms, which differs by a permutation π from what is really a fixed but entirely arbitrary representative $\mathbb{A}^<$.

2.2.1.2 Permissive-Nominal Sets

Definition 2.12. A **set with a ($\mathbb{P}$-)permutation action** X is a pair $(|\mathsf{X}|, \cdot)$ of

- a **carrier set** $|\mathsf{X}|$ and
- a group action $(\mathbb{P} \times |\mathsf{X}|) \to |\mathsf{X}|$, written infix as $\pi \cdot x$.
  So, $id \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$ for every π, π', and $x \in |\mathsf{X}|$.

Definition 2.13. Given a set with a $\mathbb{P}$-permutation action X say that $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}|$ when for all permutations $\pi \in \mathbb{P}$, if $\pi(a) = a$ for all $a \in A$ then $\pi \cdot x = x$.

Also, call $A \subseteq \mathbb{A}$ **small** when $A \subseteq S$ for some permission set S.

Definition 2.14. A **permissive-nominal set** is a set with a permutation action such that every element has a unique least small supporting set $supp(x)$. We call this the **support** of x.
X, Y will range over permissive-nominal sets.

Note in Definition 2.14 that $supp(x)$ must be *small*, that is, included in some permission set. For instance, $a \in \mathbb{A}$—with $\mathbb{A}$ having the natural permutation action given by $\pi \cdot x = \pi(x)$ for $x \in \mathbb{A}$—is supported by $\{a\}$ and $\mathbb{A} \setminus \{a\}$, but the former is small while the latter is not.

Remark 2.15. The difference between a set with a permutation action and a 'nominal' set is that nominal sets guarantee for any element, infinitely many atoms fresh for that element.

A mild generalisation of Definition 2.14 is possible, where we insist there is a supporting set but do not insist on the existence of a unique *least* such set. It is possible to do a surprising amount just with that; see for instance Fiore, Plotkin and Turi's paper Fiore et al. (1999) based on presheaves, and the 'nominal' study of infinite permutations and infinite atoms-abstraction in Gabbay (2007b).[4]

Example 2.16.

- First-order syntax with variable symbols (modelled as atoms) is a permissive-nominal set, where the permutation action permutes variable symbols directly in syntax so that e.g. $\pi{\cdot}\lambda a.t = \lambda\pi(a).\pi{\cdot}t$.

  A term t is supported by the variable symbols it contains. In this and the following examples the precise nature of the permutation group is not important.
- First-order syntax up to α-equivalence is a permissive-nominal set. The α-equivalence class of t is supported by the free variable symbols of t. A full proof is in (Gabbay 2011b, Theorem 5.18).
- Traces of π-calculus processes with channel names (atoms) taken from some permission set S, form a permissive-nominal set. A trace is supported by the set of channel names it mentions (which may be infinite in number).
- Given a permissive-nominal nominal set X the set of subsets $U \subseteq |\mathsf{X}|$ with the pointwise action $\pi{\cdot}U = \{\pi{\cdot}u \mid u \in U\}$ is a set with a permutation action (this generalises Definition 2.9).

  The subset of this consisting of those subsets $U \subseteq |\mathsf{X}|$ that have a supporting permission set under this action, forms a permissive-nominal set $pow(\mathsf{X})$.[5]

Lemma 2.17. *Suppose* X *is a permissive-nominal set and* $x \in |\mathsf{X}|$. *Then* $supp(\pi{\cdot}x) = \pi{\cdot}supp(x)$.

Proof. By a routine calculation using the group action. ■

We conclude with a useful condition for checking whether $a \in supp(x)$:

---

[4] If all permutations in $\mathbb{P}$ are finite then we have as a *Technical Lemma* that the existence of *some* supporting set implies the existence of a unique least small supporting set.

In the more general case where infinite permutations are allowed, it is possible to construct a set with a permutation action X and $x \in |\mathsf{X}|$ such that x has a supporting set but does not have a unique least small supporting set. See (Gabbay 2007b, Lemma 21) for an example.

An intermediate state is to admit infinite permutations but restrict the notion of support to consider only the finite ones. We do this in Definitions 3.1 and 3.2 and Remark 3.3 of Dowek and Gabbay (2012a).

For this paper, none of this will matter directly.

[5] Using possibly repeated powersets, arbitrarily complex structures may be constructed. Thus this example guarantees an inexhaustible supply of arbitrarily large and complex structures with which to model ... almost anything we can imagine. The survey Gabbay (2011b) explores this in detail.

Corollary 2.18. *Suppose* X *is a permissive-nominal set and* $x \in |\mathsf{X}|$. *Suppose* $b \notin supp(x)$. *Then* $(b\ a)\cdot x = x$ *if and only if* $a \notin supp(x)$.

Proof. Suppose $b \notin supp(x)$. The right-to-left implication is by the definition of support. For the left-to-right implication, we prove the contrapositive. Suppose $a \in supp(x)$. By Lemma 2.17 $supp((b\ a)\cdot x) = (b\ a)\cdot supp(x)$. By our suppositions, $(b\ a)\cdot supp(x) \neq supp(x)$. It follows that $(b\ a)\cdot x \neq x$. ■

2.2.1.3 Equivariance

Definition 2.19. Suppose X and Y are permissive-nominal sets.

Call $x \in |\mathsf{X}|$ **equivariant** when $supp(x) = \varnothing$. (So x is equivariant when $\pi\cdot x = x$ for all π.)

Call $F \in |\mathsf{X}| \to |\mathsf{Y}|$ **equivariant** when

$$\forall \pi \in \mathbb{P}.\forall x \in |\mathsf{X}|.\pi\cdot(F(x)) = F(\pi\cdot x).$$

F will range over equivariant functions between pairs of permissive-nominal sets.

Remark 2.20. The second notion of equivariance in Definition 2.19 is a special case of the first. For details, see e.g. Definition 9.3 and Lemma 9.4 of Gabbay (2011b).

Lemma 2.21. *If* F *from* $|\mathsf{X}|$ *to* $|\mathsf{Y}|$ *is equivariant then* $supp(F(x)) \subseteq supp(x)$ *for all* $x \in |\mathsf{X}|$.

Proof. Suppose $\pi \in fix(supp(x))$. By assumption $\pi\cdot F(x) = F(\pi\cdot x)$, and $\pi\cdot x = x$. ■

Definition 2.22. Write PmsPrm for the category with objects permissive-nominal sets and arrows equivariant functions between them.

So X, Y range over objects in PmsPrm (Definition 2.14).

2.2.1.4 Examples of Permissive-Nominal Sets

Throughout the rest of this document we will need the following examples of permissive-nominal sets: atoms, booleans, lists, product, equivariant elements, permutation orbits, and atoms-abstraction. We consider each in turn now.

Atoms, Booleans, infinite lists

Definition 2.23 (Atoms). $\mathbb{A}$ the set of all atoms can be considered a permissive-nominal set with a natural permutation action $\pi\cdot a = \pi(a)$. So can each $\mathbb{A}_\nu$.

Definition 2.24. If X is a permissive-nominal set say the permutation action is **trivial** when $\pi{\cdot}x = x$ for all $x \in |\mathsf{X}|$ and all $\pi \in \mathbb{P}$.

So X is trivial if and only if all its elements are equivariant.

Definition 2.25. Any 'ordinary' set can be made into a permissive-nominal set by giving it the trivial permutation action such that $\pi{\cdot}x = x$ always.

In particular, the set $\mathbb{B} = \{0,1\}$ can be considered a permissive-nominal set with the trivial permutation action; so can $\mathbb{N}$ and $\mathbb{Z}$ from Definition 2.1.

In the cases of $\mathbb{A}$ and $\{0,1\}$ only, we will be lax about the distinction between the set, and the permissive-nominal set with its natural permutation action.

Definition 2.26 (Infinite lists). Define a permissive-nominal set $\mathbb{L}$ by:

- $|\mathbb{L}|$ is the set of infinite sequences of distinct atoms $L = [a_1, a_2, a_3, \ldots]$ such that $\mathit{atms}(L) = \{a_1, a_2, a_3, \ldots\}$ is a permission set.
- $\pi{\cdot}L = [\pi(a_1), \pi(a_2), \pi(a_3), \ldots]$.

Product

Definition 2.27. Suppose I is an indexing set.[6] If X_i are permissive-nominal sets for $i \in I$ then define $\Pi_i \mathsf{X}_i$ by:

- $|\Pi_i \mathsf{X}_i|$ is the set of I-tuples $(x_i)_i$ such that $\forall i.x_i \in |\mathsf{X}_i|$ and there exists a permission set S such that $\forall i.\mathit{supp}(x_i) \subseteq S$.
- $\pi{\cdot}(x_i)_i = (\pi{\cdot}x_i)_i$ (the **elementwise** or **pointwise** action).

Permutation orbits

Permutation orbits will serve us later in Definition 2.59 (free unknowns of a term).

If X is a nominal set then $\mathit{orb}(\mathsf{X})$ is 'X quotiented by the permutation action'.

Definition 2.28. If X is a permissive-nominal set define $\mathit{orb}(\mathsf{X})$ by:

- If $x \in X$ then define its **permutation orbit** by $\mathit{orb}(x) = \{\pi{\cdot}x \mid \pi \in \mathbb{P}\}$.
- $|\mathit{orb}(\mathsf{X})| = \{\mathit{orb}(x) \mid x \in \mathsf{X}\}$.
- $\pi{\cdot}\mathit{orb}(x) = \mathit{orb}(x)$.

Lemma 2.29.

- $\mathit{supp}(\mathit{orb}(x)) = \varnothing$. *That is,* $\mathit{orb}(x)$ *is* equivariant *(Definition 2.19).*
- $\mathit{orb}(x) = \mathit{orb}(y)$ *if and only if* $y = \pi{\cdot}x$ *for some* π.

[6] For clarity, note that we intend this set to *not* have a permutation action. Or, we can take this to be a nominal set with the trivial action (Definition 2.24). We have in mind $\mathbb{N}$.

Atoms-abstraction

Definition 2.30. Suppose X is a permissive-nominal set and $\mathbb{A}_i$ is a set of atoms. Define **atoms-abstraction** $[\mathbb{A}_i]\mathsf{X}$ by:

$$\begin{array}{r} [a]x = \{(a,x)\} \cup \{(b,(b\ a)\cdot x \mid b \in \mathbb{A}_i \backslash supp(x)\} \\ |[\mathbb{A}_i]\mathsf{X}| = |[\mathbb{A}_i]\mathsf{X}| = \{[a]x \mid a \in \mathbb{A}_i,\ x \in |\mathsf{X}|\} \\ \pi\cdot[a]x = [\pi(a)]\pi\cdot x \end{array}$$

Lemma 2.31.

1. $[\mathbb{A}_i]\mathsf{X}$ *is a permissive-nominal set.*
2. $[a]x{=}[a]x'$ *if and only if* $x{=}x'$*, for* $a{\in}\mathbb{A}_i$ *and* $x{\in}|\mathsf{X}|$.
3. $[a]x{=}[a']x'$ *if and only if* $a'{\notin}supp(x)$ *and* $(a'\ a)\cdot x{=}x'$*, for* $a,a'{\in}\mathbb{A}_i$ *and* $x,x'{\in}|\mathsf{X}|$.

Lemma 2.32. *Suppose a function* F *from* $|\mathbb{A}\times\mathsf{X}|$ *to* $|\mathsf{Y}|$ *is equivariant and suppose* $\forall a,x.a \notin supp(F(a,x))$*. Then there is a unique equivariant function* $\hat{F}$ *from* $|[\mathbb{A}]\mathsf{X}|$ *to* $|\mathsf{Y}|$ *such that* $\forall a,x.\hat{F}([a]x) = F(a,x)$.

Proof. It suffices to show that if $b \notin supp(x) \cup supp(F(a,x))$ then $F(b,(b\ a)\cdot x) = F(a,x)$. By assumption $a \notin supp(F(a,x))$, so $(b\ a)\cdot F(a,x) = F(a,x)$. The result follows by equivariance. ■

Here are some basic properties of support:

Lemma 2.33.

- $supp(a) = \{a\}$.
- $supp([a]x) = supp(x) \setminus \{a\}$.
- $supp((x_1,\ldots,x_n)) = \bigcup\{supp(x_i) \mid 1 \le i \le n\}$.

Proof. Proofs are as in Gabbay and Pitts (2001) or Gabbay (2011b). ■

The fine design of PmsPrm

Studying PmsPrm (Definition 2.22) is not the point of this paper, but for the benefit of the interested reader we will discuss a few aspects of its behaviour.

- If $\mathbb{P}$ consists of finite permutations then PmsPrm is a Boolean topos, directly generalising the category of nominal sets (equivariant FM sets) from Gabbay and Pitts (2001), Gabbay (2011b). The proof proceeds much as in (Gabbay 2011b, Corollary 9.11).
- If $\mathbb{P}$ contains infinite permutations then PmsPrm is cartesian (has products) but is not necessarily cartesian closed (may not have exponentials). This is the *fuzzy support* observed in Gabbay (2007b); see (Gabbay 2007b, Lemma 21) for the concrete construction. This is reasonable, and it happens because it is possible to construct a function f on $\omega+\omega$ which satisfies $f(0) = 0$ and $f(i+1) = f(i)$

yet which is not a constant function (it returns 0 on finite cardinals and 1 on infinite ones).

- If $\mathbb{P}$ contains infinite permutations but we follow Dowek et al. (2010) and take the notions of support in Definition 2.13 and equivariance to consider only *finite* permutations, then the category we obtain is a Boolean topos but we only have $supp(x) \cap nontriv(\pi) = \varnothing$ implies $\pi \cdot x = x$ for finite π. In other words, an element can be fixed by all finite permutations and have empty support, but be shifted by some infinite permutation.

  Again, this is reasonable; it is no surprise that infinite permutations can 'observe' more than finite ones.
- If $\mathbb{P}$ contains infinite permutations and we work with presheaves (in essence, we lose the 'unique least supporting set' assumption in Definition 2.14), then we get a topos, though it is not Boolean.

In this paper we do not attempt to reason inside PmsPrm so we do not care whether it is a topos; and we do want the possibility of infinite permutations because these let us write nice algorithms and they give our logics some useful extra expressive power (see e.g. rule (**IF**) of Fig. 2.2, Sect. 2.2.2.6, and Remark 2.239).

So we admit the possibility of infinite permutations in Definition 2.6, we let Definition 2.13 consider all $\pi \in \mathbb{P}$ (even infinite ones), and we insist in Definition 2.14 that every x have a unique least small supporting set.

In another paper, another set of design decisions might be appropriate.

The reader who does not care about these considerations need not worry; they are all swept under the carpet henceforth.

2.2.1.5 Strong Support

Strong support exists in nominal terms, though this is implicit. Consider in Urban et al. (2004) the $\approx$-suspension rule in Figure 2, and Lemma 2.8. We call this *strong support*, following (Tzevelekos 2007, Definition 1).

A possibly useful intuition is that an element $x \in \mathsf{X}$ has strong support when the atoms in its support occur *in order* (a dedicated theoretical study of this is in Gabbay (2007b)). Formally, the notion of strong support enters into the mathematics in this paper via Proposition 2.38, Lemma 2.67, and Lemma 2.186.

Definition 2.34. Suppose X is a permissive-nominal set. Say $A \subseteq \mathbb{A}$ **strongly supports** $x \in |\mathsf{X}|$ when $\pi \cdot x = x$ if *and only if* $\forall a \in A.\pi(a) = a$.

If x has some strongly supporting set, call x **strongly supported**.

If every $x \in |\mathsf{X}|$ is strongly supported then call X **strongly supported**.

Lemma 2.35. *$x \in \mathsf{X}$ is strongly supported if and only if*

$$\forall \pi, \pi'.\big(\pi \cdot x = \pi' \cdot x \Leftrightarrow (\forall a \in supp(x).\pi(a) = \pi'(a))\big).$$

Proof. From Definition 2.34 by considering $\pi^{-1} \circ \pi'$. ■

Example 2.36.

- The pair $(a,b) \in \mathbb{A} \times \mathbb{A}$ is strongly supported by $\{a,b\}$.
- The unordered pair $\{a,b\} \subseteq \mathbb{A}$ with the pointwise permutation action (Definition 2.9) is not strongly supported, because $(a\ b)\cdot\{a,b\} = \{a,b\}$.
- The infinite sequences $[a_1,a_2,a_3,\ldots]$ in $\mathbb{L}$ from Definition 2.26 are strongly supported.

Definition 2.37. Suppose X and Y are permissive-nominal sets and X is strongly-supported. Suppose we are given the following data:

- For each $x \in |orb(\mathsf{X})|$ a fixed but arbitrary choice of representative $X_x \in x$.
- For each $x \in |orb(\mathsf{X})|$ a choice of $y_x \in |\mathsf{Y}|$ such that $supp(y_x) \subseteq supp(X_x)$.

Define the **equivariant extension** F of this data, which is a function from $|\mathsf{X}|$ to $|\mathsf{Y}|$, by:

$$F(\pi\cdot X_x) = \pi\cdot y_x$$

Proposition 2.38. *1. The equivariant extension is well-defined and is an equivariant function from* $|\mathsf{X}|$ *to* $|\mathsf{Y}|$.
2. Every equivariant f *is an equivariant extension.*

Proof. For the first part, by properties of orbits every $x \in |\mathsf{X}|$ has the form $\pi\cdot X_x$ for some π and for precisely one X_x. This is equivariant by construction, if it is well-defined. So suppose $\pi\cdot X_x = \pi'\cdot X_x$. By assumption X_x is strongly supported so $\pi(a) = \pi'(a)$ for every $a \in supp(X_x)$. By assumption $supp(y_x) \subseteq supp(X_x)$. The result follows by the definition of support.

The second part is easy, noting that $supp(F(x)) \subseteq supp(x)$ by Lemma 2.21. ■

2.2.2 The Syntax of Nominal Terms

Nominal terms were introduced in Urban et al. (2004). The development here is permissive, following Dowek et al. (2010), but with some additional ingredients: We allow non-equivariant constant symbols and we parameterise over a set of unknowns which is a *strongly-supported* Tzevelekos (2007).

Some example permissive-nominal terms are given in Example 2.52. See also how nominal terms are used in rewrite theories (Example 2.124), algebra (Example 2.170), and first-order logic (Sect. 2.4.2.1).

2.2.2.1 Signatures

Definition 2.39. A **sort-signature** is a tuple $(\mathcal{A},\mathcal{B})$ of **name** and **base** sorts $\mathcal{A} \subseteq \mathbb{N}$ and $\mathcal{B}$.

ν will range over name sorts; τ will range over base sorts.

A **sort language** is defined by

$$\alpha ::= \nu \mid \tau \mid (\alpha,\ldots,\alpha) \mid [\nu]\alpha.$$

Example 2.40. Example base sorts are: 'λ-terms', 'formulae', 'π-calculus processes', and 'program environments', 'functions', 'truth-values', 'behaviours', and 'valuations'.

Base sorts τ are arbitrary; later on when we build denotations they will be populated by elements of arbitrary permissive-nominal sets, see Definition 2.176.

Examples of name sorts are 'variable symbols', 'channel names', 'thread identifiers', or 'memory locations'. Name sorts ν are populated by the atoms we fixed in Definition 2.2 and which we used to build permutations and permissive-nominal sets.

Remark 2.41. $(\alpha,\ldots,\alpha)$ is a product sort and behaves as expected.

$[\nu]\alpha$ is an *atoms-abstraction sort*; this is different. The behaviour of a term of sort $[\nu]\alpha$ corresponds to 'α-abstract a name of sort ν in a term of sort α'. This is *binding without functions*: we will use atoms-abstractions (Definition 2.30) to populate atoms-abstraction sorts.

Remark 2.42. In Definition 2.39 we insist that a name sort ν is a natural number; this is not necessary but it makes it easier for us to identify name sorts with sets of atoms from Definition 2.2, which are also indexed by numbers.

Definition 2.43. A **(nominal) term-signature** over a sort-signature $(\mathcal{A},\mathcal{B})$ is a tuple $(\mathcal{C},\mathcal{X},\mathcal{F},ar)$ where:

- $\mathcal{C}$ is a permissive-nominal set of **constants**.
- $\mathcal{X}$ is a strongly supported (Definition 2.34) permissive-nominal set of **unknowns**.
- $\mathcal{F}$ is a set of equivariant **term-formers**.
- ar assigns
  - to each constant $C \in \mathcal{C}$ a base sort τ which may we write $sort(C)$,
  - to each unknown $X \in \mathcal{X}$ a sort α which we may write $sort(X)$, and
  - to each $\mathsf{f} \in \mathcal{F}$ a **term-former arity** $(\alpha)\tau$, where

  α and τ are in the sort-language determined by $(\mathcal{A},\mathcal{B})$.

A **(nominal terms) signature** Σ is then a tuple $(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{X},\mathcal{F},ar)$.

The support $supp(X)$ of an unknown $X \in \mathcal{X}$ is intuitively the atoms that may occur free in a term we substitute for that unknown, and $\mathbb{A} \setminus supp(X)$ is the atoms which may not occur free. See Proposition 2.64.

Notation 2.44. We may write $((\alpha_1,\ldots,\alpha_n))\tau$ just as $(\alpha_1,\ldots,\alpha_n)\tau$.

We write $\mathsf{f} : (\alpha)\tau$ for $ar(\mathsf{f}) = (\alpha)\tau$ and similarly we write $\mathsf{P} : \alpha$ for $ar(\mathsf{P}) = \alpha$.

Example 2.45. Here are some examples of suitable $\mathcal{X}$.

1. For each sort α and permission set S choose a disjoint countably infinite set of **unknown symbols** X^S_α, $\mathtt{Y}^S_\alpha$, ... Define $\pi{\cdot}\mathtt{X}^S_\alpha = \{(\pi',\mathtt{X}^S_\alpha) \mid \forall \mathtt{a}{\in}S.\pi(\mathtt{a}) = \pi'(\mathtt{a})\}$. Let $\mathcal{X} = \{\pi{\cdot}\mathtt{X}^S_\alpha \mid \text{all } \mathtt{X}^S_\alpha,\pi\}$ with permutation action $\pi{\cdot}(\pi'{\cdot}\mathtt{X}^S_\alpha) = (\pi\circ\pi'){\cdot}\mathtt{X}^S_\alpha$. Define $ar(\pi{\cdot}\mathtt{X}^S_\alpha) = \alpha$.

   Essentially this $\mathcal{X}$ was used in Dowek et al. (2010).
2. For each sort α choose a disjoint countably infinite set of **unknown symbols** $\mathtt{X}_\alpha$, $\mathtt{Y}_\alpha$, ... Define $\pi{\cdot}\mathtt{X}_\alpha = \{(\pi',\mathtt{X}_\alpha) \mid \forall \mathtt{a}{\in}\mathbb{A}^{<}.\pi(\mathtt{a}) = \pi'(\mathtt{a})\}$. Let $\mathcal{X} = \{\pi{\cdot}\mathtt{X}_\alpha \mid \text{all } \mathtt{X}_\alpha,\pi\}$ with permutation action $\pi{\cdot}(\pi'{\cdot}\mathtt{X}_\alpha) = (\pi\circ\pi'){\cdot}\mathtt{X}_\alpha$. Define $ar(\pi{\cdot}\mathtt{X}_\alpha) = \alpha$.
3. Take $X = (\alpha,(a_0,a_1,a_2,\ldots))$ where $\{a_i \mid i \in \mathbb{N}\}$ is a permission set and let $\mathcal{X}$ be the set of all possible X. Give this the pointwise permutation action $\pi{\cdot}X = (\alpha,(\pi(a_0),\pi(a_1),\ldots))$ and define $ar(X) = \alpha$.

   This $\mathcal{X}$ is mathematically simple, eliminating the need to take quotients over π.
4. Take $\mathcal{X} = \{0,1,2,\ldots\}$ with the trivial action $\pi{\cdot}x = x$, so every $x \in \mathcal{X}$ has $supp(x) = \varnothing$. This example illustrates that our framework is general enough to include the possibility of unknowns ranging over closed elements (a possibility also mooted in (Fernández and Gabbay 2007, Section 9.2)). By adding further structure to $\mathcal{X}$, further possibilities can be explored. See also Gabbay (2011c) and Gabbay (2012a).

In all cases it can be verified that $\mathcal{X}$ is strongly supported.

Remark 2.46. In the case that $\mathcal{X}$ the set of unknowns is as described in parts 1 or 2 of Example 2.45, $orb(X)$ (Definition 2.28) may be identified with $\mathtt{X}^S_\alpha$ or $\mathtt{X}_\alpha$ respectively.

The $\mathcal{X}$ of part 1 above may be equivalent to that of $\mathcal{X}$ of part 2, if there exists $\pi \in \mathbb{P}$ bijecting S with $S \setminus \{a\}$ for $a \in S$. This is a *shift*-permutation; see Definition 2.79 and subsequent discussion.

For the benefit of the reader familiar with 'vanilla' nominal terms as used e.g. in Urban et al. (2004), Fernández and Gabbay (2007), Gabbay and Mathijssen (2009), Fig. 2.1 gives a cheat sheet suggesting how concepts in those papers map to the 'permissive' context.

Example 2.47. A nominal terms signature for the λ-calculus would have one name sort ν, one base sort τ, and term-formers $\mathsf{lam} : ([\nu]\tau)\tau$, $\mathsf{app} : (\tau,\tau)\tau$, and $\mathsf{var} : (\nu)\tau$. The set of constants is empty, and for unknowns we can consider Example 2.45.

Vanilla	Permissive
X	Unknown with permission set $\mathbb{A}^<$
$a\#X$	$a \notin supp(X)$
$a\#r$	$a \notin fa(r)$
$\nabla \vdash r \to s$ or $\Delta \vdash r = s$	$r \to s$ or $r = s$
Extend freshness context	*shift*-permutation (approx)
Finite support	Small support

Fig. 2.1 Cheat sheet relating 'vanilla' nominal terms concepts with 'permissive' ones

Usually we assume 'plenty' of variable symbols. Definition 2.48 makes that formal:

Definition 2.48. Say that a signature $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ has **enough unknowns** when for every sort α in $(\mathcal{A}, \mathcal{B})$ and every permission set S, the set $\{orb(X) \mid X \in \mathcal{X},\ sort(X) = \alpha,\ supp(X) = S\}$ is infinite.

All the examples in Example 2.45 have enough unknowns.

2.2.2.2 Terms

Definition 2.49. For each signature $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ (Definition 2.43) define **(permissive-nominal) terms** over Σ by:

$$\frac{(a \in \mathbb{A}_\nu,\ \nu \in \mathcal{A})}{a : \nu} \qquad \frac{(sort(C) = \tau)}{C : \tau} \qquad \frac{(sort(X) = \alpha)}{X : \alpha}$$

$$\frac{r : \alpha \quad (ar(\mathsf{f}) = (\alpha)\tau)}{\mathsf{f}(r) : \tau} \qquad \frac{r_1 : \alpha_1 \ \ldots\ r_n : \alpha_n}{(r_1, \ldots, r_n) : (\alpha_1, \ldots, \alpha_n)} \qquad \frac{r : \alpha \ \ (a \in \mathbb{A}_\nu,\ \nu \in \mathcal{A})}{[a]r : [\nu]\alpha}$$

Notation 2.50. We may write $\mathsf{f}((r_1, \ldots, r_n))$ as $\mathsf{f}(r_1, \ldots, r_n)$.

Remark 2.51. Definition 2.49 is *nominal abstract syntax*: terms come pre-quotiented by α-equivalence by construction by virtue of our use of atoms-abstraction $[a]r$. That is, if $a \in \mathbb{A}_\nu$ and $r : \alpha$ then $[a]r$ is not a pair (a, r), it is a set $\{(a, r)\} \cup \{(b, (b\ a) \cdot r) \mid b \in \mathbb{A}_\nu \setminus supp(r)\}$ (Definition 2.30).

Example 2.52. Recall the signature for the λ-calculus from Example 2.47. In that signature we can form terms as illustrated in the following table, where $a : \nu$ and $X : \tau$:

$a : \nu$	This is not a λ-term.
$\mathsf{var}(a) : \tau$	If we want an atom to behave like a λ-term variable, we use var to 'inject' it into τ. This corresponds to 'x'.
$[a]a : [\nu]\nu$	An atoms-abstraction. This is not a λ-term.
$[a]\mathsf{var}(a) : [\nu]\tau$	An atoms-abstraction of a λ-term. This is not a λ-term.
$\mathsf{lam}([a]\mathsf{var}(a)) : \tau$	This corresponds to '$\lambda x.x$'.
$\mathsf{lam}([a]\mathsf{app}(X,\mathsf{var}(a))) : \tau$	An open nominal term. This corresponds to '$\lambda x.tx$, for some t'. Depending on whether $a \notin supp(X)$, we may add a side-condition 'where x is not free in t'.

Lemma 2.53. *Support and the permutation action are characterised on terms r as follows:*

$$\begin{array}{rr} supp(a) = \{a\} & supp(\mathsf{f}(r)) = supp(r) \\ supp(C) = supp(C) & supp((r_1,\ldots,r_n)) = \bigcup_{1\leq i\leq n} supp(r_i) \\ supp(X) = supp(X) & supp([a]r) = supp(r)\backslash\{a\} \end{array}$$

$$\begin{array}{rr} \pi{\cdot}a = \pi(a) & \pi{\cdot}\mathsf{f}(r) = \mathsf{f}(\pi{\cdot}r) \\ \pi{\cdot}C = \pi{\cdot}C & \pi{\cdot}(r_1,\ldots,r_n) = (\pi{\cdot}r_1,\ldots,\pi{\cdot}r_n) \\ \pi{\cdot}X = \pi{\cdot}X & \pi{\cdot}[a]r = [\pi(a)]\pi{\cdot}r \end{array}$$

Proof. By facts of the permutation action and Lemma 2.33. ■

Remark 2.54. Lemma 2.53 is important because it verifies that 'support of r' coincides with the usual definition of 'free variables (atoms) of r'. This is false of nominal terms; for instance the support of the structure $[a]\mathtt{X}$ as constructed in Urban et al. (2004) is $\{a\}$, and that of $(a\ b){\cdot}\mathtt{X}$ is $\{a,b\}$.

What makes Lemma 2.53 work is the very specific way in which we constructed our permissive-nominal terms syntax, so that it coincides with the nominal abstract syntax of Gabbay and Pitts (2001). In this sense, what Lemma 2.53 expresses is a unification (no pun intended) of the mathematics of Gabbay and Pitts (2001) and Urban et al. (2004).

In Lemma 2.53 the clauses for C and X are uninformative, of course. This is because support and the permutation action are determined by the choice of $\mathcal{C}$ and $\mathcal{X}$. If we assume further internal structure of $C \in \mathcal{C}$ or $X \in \mathcal{X}$ then we can be more specific: for instance in the case of part 1 of Example 2.45, $fa(\pi{\cdot}\mathtt{X}^S) = \{\pi(a) \mid a \in S\}$.

Because of Lemma 2.53, we are entitled to use the following notation:

Notation 2.55. In the case of syntax r, we may write $fa(r)$ for $supp(r)$ and call this the **free atoms** of r.

Lemma 2.56. $fa(\pi{\cdot}r) = \pi{\cdot}fa(r)$.

Proof. By a routine induction on r. ■

Lemma 2.57. *If $\pi(a) = \pi'(a)$ for all $a \in fa(r)$ then $\pi{\cdot}r = \pi'{\cdot}r$. The reverse implication also holds, provided that all constant symbols in r are strongly supported.*

Proof. The first part is immediate from Notation 2.55 and the definition of support in Definition 2.13.

The reverse implication is by a nominal abstract syntax induction on r. For the case of $r = [a]r'$ we α-convert a to be fresh so that $a \notin nontriv(\pi) \cup nontriv(\pi')$; by assumption 3 in Definition 2.6 we can do this. We then use part 2 of Lemma 2.31. The case of $r = X \in \mathcal{X}$ uses the assumption of strong support in Definition 2.43.[7] ■

2.2.2.3 Free Unknowns of a Term

Remark 2.58. Defining a notion of 'the free unknowns of r' is not entirely evident.

Consider for example $[a]X$ where $a \in supp(X)$. If 'X appears in $[a]X$' is true then so is '$(b\ a){\cdot}X$ appears in $[a]X$' for any $b \notin supp(X)$, since $[a]X = [b](b\ a){\cdot}X$. We deal with this in Definition 2.59 using *permutation orbits* from Definition 2.28; we simply quotient out all permutations. We take a more refined look at this later in Remark 2.93.

Definition 2.59. Define **(free) unknowns** $fU(r)$ by:

$$\begin{array}{ll} fU(a) = \varnothing & fU(\mathsf{f}(r)) = fU(r) \\ fU(C) = \varnothing & fU((r_1,\ldots,r_n)) = \bigcup_i fU(r_i) \\ fU(X) = \{orb(X)\} & fU([a]r) = fU(r) \end{array}$$

By abuse of notation we write $X \in fU(r)$ for $orb(X) \in fU(r)$ and $X \notin fU(r)$ for $orb(X) \notin fU(r)$, and so forth.

[7] Details of how induction on nominal abstract syntax allows us to α-convert and make freshness assumptions, are the topic of Gabbay (2011b). A less fancy proof of both implications by a standard induction—so not this new-fangled nominal nonsense—on terms not quotiented by α-equivalence, is in Appendix A of Dowek et al. (2010), proof of Lemma 4.15 on page 50. We leave it to the reader to judge which is the nicer proof.

Lemma 2.60. *$fU(r)$ is well-defined.*

Proof. Using Lemmas 2.29 and 2.32. ■

Notation 2.61. Call a term r **ground** when $fU(r) = \varnothing$. Otherwise, call r **open**.

2.2.2.4 Substitutions

Remark 2.62. Substitutions are of course how unknowns 'stand for' terms. Somewhat later we will develop a denotational theory for nominal terms, and so valuations for unknowns will appear, in Definition 2.178. Between now and then, substitutions are king.

The permissive-nominal framework we work with allows us an elegant definition:

Definition 2.63. Suppose Σ is a signature. A **substitution** θ in Σ is an equivariant function from $\mathcal{X}$ to terms in Σ such that $sort(\theta(X)) = sort(X)$ always.
θ will range over substitutions.
Write *id* for the **identity** substitution mapping X to X always. It will always be clear whether *id* means the identity substitution or permutation.

The reader familiar with nominal terms will expect a 'freshness' condition on substitutions corresponding to '$\nabla' \vdash \nabla\theta$', as in for example Equation (11) or Lemma 2.14 of Urban et al. (2004), or '$fa(\theta(X)) \subseteq supp(X)$' as in Definition 3.1 of Dowek et al. (2010). This follows immediately from equivariance:

Proposition 2.64. *If θ is a substitution then $\forall X{\in}\mathcal{X}.fa(\theta(X)) \subseteq supp(X)$.*

Proof. Direct from Lemma 2.21. ■

Putting Propositions 2.64 and 2.38 together with a concrete $\mathcal{X}$ recovers the notion of substitution used in Dowek et al. (2010):

Lemma 2.65. *If $\mathcal{X}$ is equal to example 1 of Example 2.45 then the construction in Definition 2.37 describes a 1–1 correspondence between substitutions and maps from unknowns $\mathtt{X}^S_\alpha$ to terms $t : \alpha$ such that $fa(t) \subseteq S$.*

Definition 2.66. Suppose $fa(t) \subseteq supp(X)$ and $sort(t) = sort(X)$. Write $[X{:=}t]$ for the **atomic substitution** equivariantly extending the assignment $X \mapsto t$, so that

$$\begin{array}{ll} [X{:=}t](\pi{\cdot}X) & = \pi{\cdot}t \text{ and} \\ [X{:=}t](Y) & = Y \quad \text{for all other } Y. \end{array}$$

By Proposition 2.38 we have:

Lemma 2.67. *Definition 2.66 is well-defined. That is, if $\pi{\cdot}X = \pi'{\cdot}X$ then $\pi{\cdot}t = \pi'{\cdot}t$.*

Remark 2.68. The 'moderated unknown' $\pi{\cdot}X$ in Definition 2.66 is an artefact of our writing $[X{:=}t]$ instead of a mathematically equal $[\pi{\cdot}X{:=}\pi{\cdot}t]$ for some other π.

Since θ is equivariant its behaviour on $\pi{\cdot}X$ is already determined by its behaviour on X and so we could unambiguously specify $[X{:=}t]$ succinctly as $[X{:=}t](X) = t$ and $[X{:=}t](Y) = Y$.

Definition 2.69. Define a **substitution action** on terms by:

$$\begin{array}{ll} a\theta = a & \mathsf{f}(r)\theta = \mathsf{f}(r\theta) \\ C\theta = C & (r_1,\ldots,r_n)\theta = (r_1\theta,\ldots,r_n\theta) \\ X\theta = \theta(X) & ([a]r)\theta = [a](r\theta) \end{array}$$

Note that $X\theta$ refers to θ acting on X as a term whereas $\theta(X)$ refers the value of the function θ at X. The substitution action is well-defined by Lemmas 2.32 and 2.33.

Remark 2.70. Famously, the nominal terms substitution is capturing (Urban et al. 2004, Definition 2.13). We spell out how this works in our permissive-nominal context: Suppose $supp(X)$ is equal to a permission set S and $a \in S$ and $b \notin S$ (where we assume appropriate sorts). Then:

- $([a]X)[X{:=}a] = [a]a$. The a in the substitution $[X{:=}a]$ has been captured by the $[a]X$.
- $([b]X)[X{:=}a] = [b]a$.
- It is impossible to even ask what $([b]X)[X{:=}b]$ is equal to because $[X{:=}b]$ is not even a substitution, since $b \notin S$. So $b \notin S$ cannot be captured by a substitution $[X{:=}b]$, because that substitution does not exist. This is no *ad hoc* restriction: by Proposition 2.64 it *cannot* exist.
- Also, $[b](b\ a){\cdot}X = [a]X$. By construction in Definition 2.66

$$([b](b\ a){\cdot}X)[X{:=}a] = [b](b\ a){\cdot}a = [b]b = [a]a.$$

  Also $[X{:=}a] = [(b\ a){\cdot}X{:=}b]$ and $([b](b\ a){\cdot}X)[(b\ a){\cdot}X{:=}b] = [b]b$.

  That is, the choice of representative of $[a]X$ and $[X{:=}a]$ does not matter for capture to occur.

It is interesting to note that in our setting, $[X{:=}a]$ is *equivariant* and that $a \notin supp([a]X)$. If a is fresh for both $[X{:=}a]$ and $[a]X$, how can it be captured?

What allows a to get captured is the *strong support property* of X. Because X is strongly supported, we can think of it as 'containing' a list of its supporting atoms in some order, so that the a in $[X{:=}a]$ is bound by $supp(X)$ but in being bound it points to a 'position' in X.

Viewed from this interesting perspective, the nominal substitution action is not capturing at all: it is simply a compact way to present an 'infinite raising' (terminology from higher-order logic), or a de Bruijn index.

Lemma 2.71. $\pi{\cdot}(r\theta){=}(\pi{\cdot}r)\theta$.

Proof. By a routine induction on r using equivariance. ■

Lemma 2.72. $fa(r\theta) \subseteq fa(r)$.

Proof. From Lemmas 2.21 and 2.71. ■

Lemma 2.73. $r\theta = r\theta'$ *if and only if* $\forall X{\in}fU(r).\theta(X) = \theta'(X)$.

Proof. By a routine induction on r. We consider two cases:

- *The case* $[a]r$. Suppose $\theta(X) = \theta'(X)$ for every $X \in fU([a]r)$. $fU([a]r) = fU(r)$ so by inductive hypothesis $r\theta = r\theta'$. The result follows from the definitions.
  The reverse implication is similar.
- *The case* X. Suppose $\theta(\pi{\cdot}X) = \theta'(\pi{\cdot}X)$ for all π. Then taking $\pi = id$ we have $X\theta = \theta(X) = \theta'(X) = X\theta'$.
  Conversely if $X\theta = X\theta'$ then using equivariance (Definition 2.63) $\theta(\pi{\cdot}X) = \theta'(\pi{\cdot}X)$ for all π. ■

Remark 2.74. Recall from Definition 2.59 that we write $X \in fU(r)$ for $orb(X) \in fU(r)$. It might seem that the condition $\forall X{\in}fU(r).\theta(X) = \theta'(X)$ in Lemma 2.73 would require checking $\theta(X) = \theta'(X)$ for infinitely many X provided that $fU(r) \neq \varnothing$. In fact, this is not the case: by equivariance of θ, we only need to check equality for one representative X of each permutation orbit: $X \in orb(X) \in fU(r)$.

2.2.2.5 Composition and Invertibility of Substitutions

Definition 2.75. Define **composition** of substitutions $\theta_1 \circ \theta_2$ by

$$(\theta_1 \circ \theta_2)(X) = (\theta_1(X))\theta_2.$$

Lemma 2.76. $(r\theta)\theta' = r(\theta \circ \theta')$.

Proof. By induction on r. ■

Definition 2.77. Call θ **invertible** when there exists θ^{-1} such that $\theta \circ \theta^{-1} = \theta^{-1} \circ \theta = id$.

Lemma 2.78. *θ is invertible if and only if θ is a bijection on $\mathcal{X}$ the set of all unknowns. Furthermore, if θ is invertible then $supp(\theta(X)) = supp(X)$ always.*

Proof. Substitution cannot make syntax smaller, or (by Lemma 2.72) make free atoms larger. ■

So an invertible θ must biject unknowns of a particular sort and permission set with other unknowns of that same sort and permission set. So, like atoms, we can rename unknowns to 'be fresh' (provided we have given ourselves enough of them).

Invertible substitutions will be useful later, and they are also one manifestation of a more general framework of *two-level nominal sets* Gabbay (2011c).

2.2.2.6 *Shift*-Permutations

The reader may be familiar with nominal freshness conditions $a\#\mathtt{X}$ from Urban et al. (2004). In that paper, $a\#\mathtt{X}$ indicated that $\mathtt{X}$ should be substituted only for terms for which a is fresh.

In Urban et al. (2004), Fernández and Gabbay (2007), we might have to extend a freshness context in order to give ourselves more fresh atoms. This is what rules like (**Fr**) from (Gabbay and Mathijssen 2008c, Figure 2) or (**fr**) from (Gabbay and Mathijssen 2009, Figure 2) do; see also Fernández and Gabbay (2010) where the issue of extending nominal freshness contexts is made very explicit.

In principle, permission sets guarantee an infinite supply of fresh atoms, so the problem of extending a freshness context should not arise. But this may rely on oracular knowledge of what the permission set should be, which we might prefer not to assume. The choice of nominal permutation group $\mathbb{P}$ gives us the power to implicitly parameterise over this decision.

Suppose we have some X such that $a \in supp(X)$ and we perhaps we are solving a unification problem and the information that a should be fresh for X has just been revealed by an algorithm; so we want to remove a from the permission set of X. This arises in the unification algorithm of Sect. 2.3.1.

Suppose alternatively we would like to make the permission set *larger*, e.g. if we know $\forall X.\phi$ and want to deduce $\phi[X{:=}t]$ where $fa(t) \not\subseteq supp(X)$, or we have a rewrite rule $X \to X$ and want to deduce $t \to t$ where again $fa(t) \not\subseteq supp(X)$. This arises in the nominal rewriting, algebra and permissive-nominal logic which we construct later.

This is where *shift*-permutations can help.

Definition 2.79. Call a permutation $\delta \in \mathbb{P}$ a ***shift*-permutation** when there exists a permission set S and atom $a \in S$ such that $S \setminus \{a\} = \delta \cdot S$.
Say that a nominal permutation group $\mathbb{P}$ has ***shift*-permutations** when for every permission set S and atom $a \in \mathbb{A}$ there exists a permutation $\pi \in \mathbb{P}$ such that $\pi \cdot S = S \setminus \{a\}$.

Remark 2.80. Another way to read Definition 2.79 is that $\mathbb{P}$ has *shift*-permutations when, if S is a permission set and A is finite, then $S \setminus A$ and $S \cup A$ are permission sets. Stronger versions allowing infinite A are certainly imaginable.

Example 2.81. The nominal permutation group in part 2 of Example 2.7 has *shift*-permutations.

δ_i bijects $\mathbb{A}_i^<$ with $\mathbb{A}_i^< \setminus \{f(0)\}$. Using swappings we can now generate a π to biject any permission set S with $S \setminus \{a\}$ for $a \in S$. We give the concrete constructions below, culminating with Lemma 2.88.

For the rest of this section we work concretely with the nominal permutation group from part 2 of Example 2.7; the reader only interested in the high-level picture can skip this. Recall the bijections f_i from integers to atoms from part 2 of Example 2.7. For simplicity drop the subscript i and consider just one set of atoms.

Notation 2.82. By abuse of notation write 0 for the atom $f(0)$.

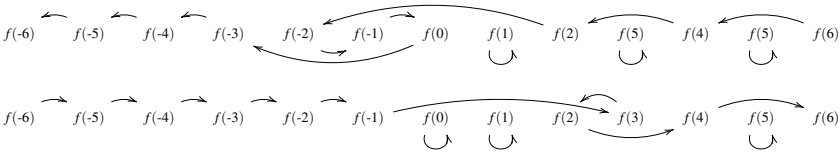
Definition 2.83. 1. If $a \in \mathbb{A}^<$ then define δ^{-a} by:

$$\delta^{-a} = (a\,0) \circ \delta \circ (a\,0)$$

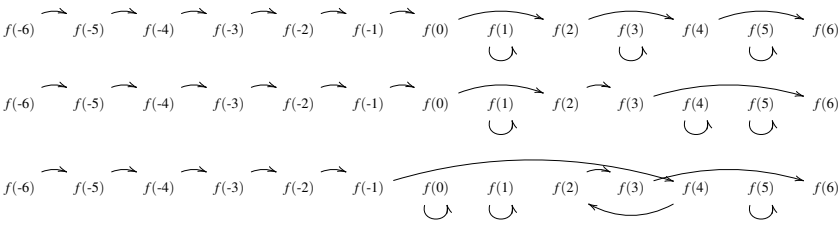
2. If $b \in \mathbb{A}^>$ then for some fixed but arbitrary choice of $c \in \mathbb{A}^>$ such that $\delta(c) = c$ (and so also $c \notin \mathbb{A}^<$), define δ^{+b} by:

$$\delta^{+b} = (b\,0) \circ (c\,b) \circ \delta^{-1} \circ (c\,b) \circ (b\,0)$$

Example 2.84. We illustrate δ^{-a} and δ^{+b} where $a = f(-2)$ and $b = f(3)$ and where we take $c = b$:



We also consider the slightly more complex example of δ^{+d} where $d = f(4)$, and again we take $c = f(3)$. We do this in three steps, where we illustrate δ^{-1}, then $(c\,d) \circ \delta^{-1} \circ (c\,d)$, and finally δ^{+d}:



Lemma 2.85. *1. If $a \in \mathbb{A}^<$ then δ^{-a} bijects $\mathbb{A}^<$ with $\mathbb{A}^< \setminus \{a\}$.*
2. If $b \in \mathbb{A}^>$ then δ^{+b} bijects $\mathbb{A}^<$ with $\mathbb{A}^< \cup \{b\}$.

Proof. For the first part, suppose $a \in \mathbb{A}^<$. Then $\mathbb{A}^< = (a\,0) \cdot \mathbb{A}^<$. We reason as follows:

$$\begin{aligned}\delta^{-a} \cdot ((a\,0) \cdot \mathbb{A}^<) &= ((a\,0) \circ \delta \circ (a\,0) \circ (a\,0)) \cdot \mathbb{A}^< \\ &= ((a\,0) \circ \delta) \cdot \mathbb{A}^< = (a\,0) \cdot (\mathbb{A}^< \setminus \{0\}) = \mathbb{A}^< \setminus \{a\}\end{aligned}$$

Now suppose $b \in \mathbb{A}^>$. It is easier to work with $(\delta^{+b})^{-1}$, to keep the parallel with the previous case. So $\mathbb{A}^< \cup \{b\} = ((b\,0) \cdot \mathbb{A}^<) \cup \{0\}$. We reason as follows:

$$
\begin{aligned}
(\delta^{+b})^{\text{-}1}\cdot(((b0)\cdot\mathbb{A}^{<})\cup\{0\}) &= ((b0)\circ(cb)\circ\delta\circ(cb)\circ(b0))\cdot(((b0)\cdot\mathbb{A}^{<})\cup\{0\}) && \text{Def. 2.83}\\
&= \big(((b0)\circ(cb)\circ\delta\circ(cb)\circ(b0))\cdot((b0)\cdot\mathbb{A}^{<})\big)\cup\{0\} && \delta(c){=}c\\
&= \big(((b0)\circ(cb)\circ\delta\circ(cb))\cdot\mathbb{A}^{<}\big)\cup\{0\} && \text{Fact}\\
&= \big(((b0)\circ(cb)\circ\delta)\cdot\mathbb{A}^{<}\big)\cup\{0\} && b,c\notin\mathbb{A}^{<}\\
&= \big(((b0)\circ(cb))\cdot(\mathbb{A}^{<}\setminus\{0\})\big)\cup\{0\} && \delta\cdot\mathbb{A}^{<}{=}\mathbb{A}^{<}\setminus\{0\}\\
&= (\mathbb{A}^{<}\setminus\{0\})\cup\{0\} && b,c\notin\mathbb{A}^{<}\setminus\{0\}\\
&= \mathbb{A}^{<}
\end{aligned}
$$

■

Recall from Definition 2.10 that each permission set S has the form $\pi\cdot\mathbb{A}^{<}$ for some permutation π.

Definition 2.86. For each S make some choice of permutation π_S such that $S = \pi_S^{\text{-}1}\cdot\mathbb{A}^{<}$.[8]

Definition 2.87. Suppose S is a permission set and $a\in S$ and $b\notin S$. Then we define:

$$\delta_{S\text{-}a} = \pi_S^{\text{-}1}\circ\delta^{\text{-}\pi_S(a)}\circ\pi_S \qquad \delta_{S+b} = \pi_S^{\text{-}1}\circ\delta^{+\pi_S(b)}\circ\pi_S$$

The concrete details of the construction are only interesting insofar as they give us Lemma 2.88. Other permutations are possible, but we only need that one exists.

Lemma 2.88.

1. *$\delta_{S\text{-}a}$ bijects S with $S\setminus\{a\}$.*
2. *δ_{S+b} bijects S with $S\cup\{b\}$.*

Proof. From Lemma 2.85. ■

Definition 2.89. Suppose S is a permission set and $supp(X) = S$. Suppose D is a finite list of atoms $d_1,\dots,d_n$ and E is a finite list of atoms $e_1,\dots,e_n$. Suppose $\{d_1,\dots,d_n\}\subseteq S$ and $\{e_1,\dots,e_n\}\cap S=\varnothing$. Then define $\delta_{S\text{-}D}$ and δ_{S+E}, and $X\text{-}D$ and $X{+}E$ by:

[8] Taking the inverse here saves writing $^{\text{-}1}$ quite so many times in Definition 2.87, and is harmless since permutations are invertible.

$$\begin{array}{rlrl} \delta_{S\text{-}[]} &= id & \delta_{S+[]} &= id \\ \delta_{S\text{-}[d]} &= \delta_{S\text{-}d} & \delta_{S+[e]} &= \delta_{S+e} \\ \delta_{S\text{-}d,D} &= \delta_{(S\setminus\{d\})\text{-}D} \circ \delta_{S\text{-}d} & \delta_{S+e,E} &= \delta_{(S\cup\{e\})+E} \circ \delta_{S+e} \\ X\text{-}D &= \delta_{supp(X)\text{-}D} \cdot X & X+E &= \delta_{supp(X)+E} \cdot X \end{array}$$

Lemma 2.90. *Suppose S is a permission set. Suppose D and E are finite lists of atoms $d_1,\ldots,d_n$ and $e_1,\ldots,e_n$. Suppose $\{d_1,\ldots,d_n\} \subseteq S$ and $\{e_1,\ldots,e_n\} \cap S = \varnothing$.*

Then $\delta_{S\text{-}D}$ bijects S with $S \setminus \{d_1,\ldots,d_n\}$ and δ_{S+E} bijects S with $S \cup \{e_1,\ldots,e_n\}$.

Proof. Using Lemma 2.88. ■

Corollary 2.91. *S is a permission set if and only if $S = (\mathbb{A}^< \setminus A) \cup B$ for some finite $A \subseteq \mathbb{A}^<$ and $B \subseteq \mathbb{A}^>$.*

Proof. If S is a permission set then by Definition 2.10 $S = \pi \cdot \mathbb{A}^<$ for some π and the result follows by a routine induction on the generators of π (swapping and δ; see part 2 of Example 2.7).

Conversely consider $S = (\mathbb{A}^< \setminus A) \cup B$. Let D be the atoms in A in some order, and E be the atoms in B in some order. Then we apply $\delta_{S\text{-}D}$ and then $\delta_{(S\setminus A)+E}$ and use Lemma 2.90. ■

Remark 2.92. The reader may be familiar with the de Bruijn *shift* function $\uparrow$ (Abadi et al. 1991, Section 2.2). This maps $\mathbb{N}$ to $\mathbb{N}\setminus\{0\}$ by mapping $j \in \mathbb{N}$ to $j+1 \in \mathbb{N}$, and in doing so it 'creates a fresh number' 0. The reader familiar with presheaf techniques may know of a functor δ and arrow up, which work the same way, as exemplified in (Fiore et al. 1999, Section 1).

δ_i from part 2 of Example 2.7 is in the same spirit. It shifts 'down' instead of 'up', but δ_i^{-1} shifts 'up'.

Note that δ is *invertible* ($\uparrow$ and up are not). This is consistent with the general preference of nominal techniques for using permutations where possible.

2.2.2.7 Occurrences

Remark 2.93. As discussed in Remark 2.58 we have to be careful if we wish to say 'X appears in r'; this might not quite mean what we think it does.

For example if 'X appears in $[a]X$' where $a \in supp(X)$ then also '$(b\ a)\cdot X$ appears in $[a]X$' for any $b \notin supp(X)$. We dealt with this in Definition 2.59 by quotienting out all permutations.

But this is a little drastic. For instance, '$(b\ a)\cdot X$ appears in $[a]X$' is not true for $b \in supp(X)$; it is not the case that if 'X appears in r' then '$\pi\cdot X$ appears in r' for any π.

We did not need to quotient out *all* permutations—only some of them—and so returning $orb(X)$ in Definition 2.59 throws out more information than necessary.

Definitions 2.94 and 2.95 develop a more refined notion of occurrence, based on an intuition of 'X appears in r under a list of abstractions D'. This will be useful later.

Definition 2.94. D will range over finite lists of distinct atoms. A **(level 2) occurrence** is a term of the form $[D]X$ where $[]X$ is X and $[a,D]X$ is $[a][D]X$.

Definition 2.95. Define the **occurrences in** r inductively by:

$$\begin{array}{rr} occ(a) = \varnothing & occ(\mathtt{f}(r)) = occ(r) \\ occ(C) = \varnothing & occ((r_1,\ldots,r_n)) = \bigcup occ(r_i) \\ occ(X) = X & occ([a]r) = \{[a]x \mid x{\in}occ(r)\} \end{array}$$

Example 2.96.

- X occurs in X.
- $[a]X$ occurs in $[a]X$ and also in $[a](X,Y)$; so does $[a]Y$. X does not occur in $[a]X$ or $[a](X,Y)$.
- $[a][b]X$ and $[a][a]X$ occur in $[a]([b]X,[a]X)$.

We write occurrences as $[D]X$ for D a finite list of distinct atoms. Note that $[a][a]X$ is an occurrence since it is equal to $[a][b](b\ a){\cdot}X$ where $b \notin supp(X)$. This is an equality, not an equivalence imposed on terms after they are constructed, because of our use of atoms-abstraction (Definition 2.30) in syntax (Definition 2.49).

2.3 Rewrites, Equations, and Algebras

2.3.1 Unification

We want to write rewrite rules and equality axioms using nominal terms. In order to do this, we have to unify nominal terms (answer the question: "given r and s what substitutions θ make them equal?"). Unification makes unknowns 'come alive' and represent unknown terms.

Therefore, we now create a nominal unification algorithm. One notable property of nominal unification is that it has most general (principal) unifiers Theorem 2.118. Contrast this with higher-order unification, which does not (Dowek 2001, Section 4). This is one reason we say that the nominal approach to names and binding has a 'first-order' flavour.

The algorithm we use follows the spirit of Urban et al. (2004) but the design is different. In Urban et al. (2004) a solution to $[a]\mathtt{X} \stackrel{?}{=} [b]\mathtt{Y}$ would be $(b\#\mathtt{X}, [\mathtt{Y}{:=}(b\ a){\cdot}\mathtt{X}])$; that is, the unification algorithm returns a pair of some freshness side-conditions and some equalities.[9]

[9]We write typewriter font to avoid confusion between the symbols used in Urban et al. (2004) (which have no support) and the elements $X \in \mathcal{X}$ used in this paper (which do have support). To see how to travel between these two worlds see part 2 of Example 2.45, or Dowek et al. (2010).

Here, solutions are equalities only, without freshness conditions. The extra power resides in the notion of an *shift*-permutation (Definition 2.79).

A solution to $[a]X = [b]Y$ where $b \in supp(X) = supp(Y)$ would be

$$[X{:=}\delta' \cdot X,\ Y{:=}((b\ a) \circ \delta') \cdot X]$$

where δ' bijects $supp(X)$ with $supp(X) \setminus \{b\}$ (and by this bijection 'internally freshens' X with respect to b).

In another design (Dowek et al. 2010, Section 5) we use permission sets and fresh unknowns; a solution to $[a]X = [b]Y$ where $b \in supp(X) = supp(Y)$ is $[X{:=}Z,\ Y{:=}(b\ a) \cdot Z]$ where $supp(Z) = supp(X) \setminus \{b\}$. Generating Z fresh requires us to solve problems in a context of 'known unknowns' $\mathcal{V}$. This introduces a notion of state and sequentiality into the algorithm of Dowek et al. (2010) which we avoid here.

Nothing forces us to feed the unification algorithm syntax with *shift*-permutations, even if the solutions it returns might mention them; similarly in Urban et al. (2004) we may obtain a solution with freshness side-conditions to a unification problem with only equalities. So use of *shift*-permutation in Definition 2.97 should not be read as a commitment to using them everywhere (though we do note empirically that *shift* seems to be useful elsewhere too).

The main definition of this section is Definition 2.104. The main result is Theorem 2.118.

Definition 2.97. Throughout this Section we fix some signature Σ and we work with syntax over Σ. We assume a nominal permutation group $\mathbb{P}$ with *shift*-permutations and a set of unknowns $\mathcal{X}$ such that every unknown is supported by a permission set (see e.g. part 2 of Example 2.45).

2.3.1.1 The Unification Algorithm

Definition 2.98. A **(unification) equality** is a unordered pair $r \stackrel{?}{=} s$ (so $r \stackrel{?}{=} s$ is identical to $s \stackrel{?}{=} r$) such that:

1. $sort(r) = sort(s)$.
2. If $[D]X$ and $[D']\pi \cdot X$ are both in $occ(r) \cup occ(s)$ then π is finite.
   So we exclude an equality like $X \stackrel{?}{=} \delta \cdot X$, where δ is a shift permutation and $nontriv(\delta) \cap supp(X)$ is not finite.

A **(unification) freshness** is an ordered pair $a\#_? r$.

Let ef range over equalities or freshnesses and define $ef\theta$ by:

- $(r \stackrel{?}{=} s)\theta = (r\theta \stackrel{?}{=} s\theta)$.
- $(a\#_? r)\theta = (a\#_?(r\theta))$.

$$
\begin{array}{lll}
(\stackrel{?}{=}\mathbf{a}) & a \stackrel{?}{=} a, Pr & \Longrightarrow \quad Pr \\
(\stackrel{?}{=}\mathbf{C}) & C \stackrel{?}{=} C, Pr & \Longrightarrow \quad Pr \\
(\stackrel{?}{=}\mathsf{f}) & \mathsf{f}(r) \stackrel{?}{=} \mathsf{f}(s), Pr & \Longrightarrow \quad r \stackrel{?}{=} s, Pr \\
(\stackrel{?}{=}()) & (r_1,\ldots,r_n) \stackrel{?}{=} (s_1,\ldots,s_n), Pr & \Longrightarrow \quad r_1 \stackrel{?}{=} s_1,\ldots,r_n \stackrel{?}{=} s_n, Pr \\
(\stackrel{?}{=}[]) & [a]r \stackrel{?}{=} [a]s, Pr & \Longrightarrow \quad r \stackrel{?}{=} s, Pr \\
(\stackrel{?}{=}\mathbf{X}) & X \stackrel{?}{=} \pi\cdot X, Pr & \Longrightarrow \quad a_1\#_? X,\ldots,a_n\#_? X, Pr \\
 & & (\{a_1,\ldots,a_n\} = nontriv(\pi)\cap supp(X)) \\
(\mathbf{F}) & r \stackrel{?}{=} X, Pr & \Longrightarrow \quad a\#_? r,\ r \stackrel{?}{=} X,\ Pr \\
 & & (a \in fa(r)\backslash supp(X)) \\
(\mathbf{F}\#) & a\#_? r, Pr & \Longrightarrow \quad Pr \qquad (a \notin fa(r)) \\
(\mathbf{F}\mathsf{f}) & a\#_? \mathsf{f}(r), Pr & \Longrightarrow \quad a\#_? r, Pr \\
(\mathbf{F}()) & a\#_? (r_1,\ldots,r_n), Pr & \Longrightarrow \quad a\#_? r_1,\ldots,a\#_? r_n, Pr \\
(\mathbf{F}[]) & a\#_? [b]r, Pr & \Longrightarrow \quad a\#_? r,\ Pr \\
(\mathbf{IE}) & r \stackrel{?}{=} X, Pr & \stackrel{[X:=r]}{\Longrightarrow} \quad Pr[X:=r] \\
 & & (X \notin fU(r), fa(r) \subseteq supp(X)) \\
(\mathbf{IF}) & a\#_? X, Pr & \stackrel{[X:=\delta_{X\text{-}a}\cdot X]}{\Longrightarrow} \quad Pr[X:=\delta_{X\text{-}a}\cdot X]
\end{array}
$$

Fig. 2.2 Simplification rules for problems

A **nominal unification problem** Pr is a finite list $ef_1,\ldots,ef_n$.

We (ab)use standard sets notation and write $ef \in Pr$ as shorthand for 'ef appears in the list Pr'.

Remark 2.99. Condition 2 in Definition 2.98 protects $(\stackrel{?}{=}\mathbf{X})$ in Fig. 2.2 from an 'infinite freshness explosion', if $nontriv(\pi)\cap supp(X)$ is not finite. This condition exists implicitly in Urban et al. (2004), in the sense that all permutations there are finite. However, condition 2 is not only computationally motivated. Given constants C and D with $supp(C) = \varnothing = supp(D)$, $X \stackrel{?}{=} \delta\cdot X$ may have solutions C and D but have no principal solution. We discuss the implications of this condition to nominal rewriting, at the end of Sect. 2.3.3.

Definition 2.100. If $Pr = ef_1,\ldots,ef_n$ is a problem then define $Pr\theta$ by:

$$Pr\theta = ef_1\theta,\ldots,ef_n\theta$$

Say θ **solves** Pr and call θ a **solution** to Pr when

$$
\begin{array}{llll}
r\theta = s\theta & \text{for every} & r \stackrel{?}{=} s \in Pr, & \text{and} \\
a \notin fa(r\theta) & \text{for every} & a\#_? r \in Pr. &
\end{array}
$$

Write $Sol(Pr)$ for the set of solutions to Pr and call Pr **solvable** when $Sol(Pr)$ is non-empty.

Recall the definition of $\theta \circ \theta'$ from Definition 2.75.

Lemma 2.101. *$\theta \circ \theta' \in Sol(Pr)$ if and only if $\theta' \in Sol(Pr\theta)$.*

Proof. By unpacking Definition 2.100 and using Lemma 2.76. ■

Definition 2.102. Define a **simplification** rewrite relation $Pr \Longrightarrow Pr'$ on unification problems by the rules in Fig. 2.2.

We call rules (**IF**) and (**IE**) **instantiating rules**. We call all the other rules **non-instantiating rules**.

In (**IF**) $\delta_{X\text{-}a}$ is some permutation bijecting $supp(X)$ with $supp(X) \setminus \{a\}$. We can do this because we assumed *shift*-permutations in Definition 2.97.[10]

Write $\Longrightarrow^*$ for the transitive and reflexive closure of $\Longrightarrow$.

Remark 2.103. Compare Fig. 2.2 with Figure 3 of Urban et al. (2004). Note of $(\stackrel{?}{=}[])$ that we do not consider the case $[a]r \stackrel{?}{=} [b]s$. This is because α-equivalence is handled automatically by nominal abstract syntax, specifically by Definition 2.30. So α-renaming is pushed into the background (just as is usually the case for first-order syntax) and these rules are somewhat higher-level than those of Urban et al. (2004).

We also do not require a rule $a\#_?[a]r, Pr \Longrightarrow Pr$ because the abstracted atom in $[a]r$ is α-convertible; more formally, $[a]r = [b](b\ a)\cdot r$ for some/any fresh b (so $b \notin fa(r)$).

Finally, in $(\stackrel{?}{=}\mathbf{X})$ we do not need to write $\pi\cdot X \stackrel{?}{=} \pi'\cdot X$ (though we could) because unknowns are just a strongly-supported nominal set. We know that $nontriv(\pi) \cap supp(X)$ is finite by a routine argument based on condition 2 of Definition 2.98. It is not hard to check that the instantiating rules (**IF**) and (**IE**) do indeed preserve these conditions—(**IF**) involves a *shift* permutation, but in a manner that is applied uniformly to the whole problem.

Definition 2.104. If Pr is a problem, define a **unification algorithm** by:

1. Rewrite Pr using the rules of Definition 2.102 where possible, with top-down precedence (so apply $(\stackrel{?}{=}\mathbf{a})$ before $(\stackrel{?}{=}\mathsf{f})$, and so on).
2. If we reduce to $\varnothing$ then we succeed and return θ where θ is the composition of all the substitutions labelling rewrites (we take $\theta = id$ if there are none). Otherwise, we fail.

[10] The specific choice does not matter. Intuitively this is because permutations are invertible so any one choice and be undone and redone at will. A more formal statement of this is Theorem 2.112. For an example of a *shift*-permutation concretely constructed, see Definition 2.87.

This algorithm generates *shifts* just like in Urban et al. (2004) we generated freshness conditions, and for the same reason.

Remark 2.105. Note in Definition 2.104 that we apply each rule to the head of the list Pr. This is to prevent 'unfair' looping, e.g. repeatedly applying (**F**) to some equality $r \stackrel{?}{=} X$ wherever it appears in Pr.

Note also that the rule (**F#**) is equivalent—in the presence of the other rules—to three rules as follows:

$$\begin{array}{lll} (\mathbf{Fa}) & a\#_? b, Pr \Longrightarrow Pr & \\ (\mathbf{FC}) & a\#_? C, Pr \Longrightarrow Pr & (a \notin supp(C)) \\ (\mathbf{FX}) & a\#_? X, Pr \Longrightarrow Pr & (a \notin supp(X)) \end{array}$$

Proposition 2.106. *The algorithm of Definition 2.104 always terminates.*

Proof. It is not hard to generate an inductive quantity which is reduced by the reductions in Fig. 2.2. ■

2.3.1.2 Examples of the Algorithm

We assume the permutation group from part 2 of Example 2.7 and we recall the definition of X-D from Definition 2.87.

Example one (succeeds).

Suppose $a, c \in \mathbb{A}^<$ and $d \notin \mathbb{A}^<$. Take $supp(X) = \mathbb{A}^<$ and suppose a term-former $\mathtt{g}$. We apply the algorithm to $\{\mathtt{g}([a]X, [a]a) \stackrel{?}{=} \mathtt{g}([d]c, [d]d)\}$:

$$\begin{array}{lll} \mathtt{g}([a]X, [a]a) \stackrel{?}{=} \mathtt{g}([d]c, [d]d) & \Longrightarrow & (\stackrel{?}{=}\mathtt{g}), (\stackrel{?}{=}()) \\ [a]X \stackrel{?}{=} [d]c\ , [a]a \stackrel{?}{=} [d]d & \Longrightarrow & (\stackrel{?}{=}[]), [a]X = [d](d\ a)\cdot X \\ (d\ a)\cdot X \stackrel{?}{=} c\ , [a]a \stackrel{?}{=} [d]d & \stackrel{[X:=c]}{\Longrightarrow} & (\mathbf{IE}) \\ [a]a \stackrel{?}{=} [d]d & \Longrightarrow & (\stackrel{?}{=}[]), [a]a = [d]d \\ d \stackrel{?}{=} d & \Longrightarrow & (\stackrel{?}{=}\mathbf{a}) \\ \varnothing \quad \text{Success, with } [X{:=}c] & & \end{array}$$

Example two (succeeds).

Suppose $a, c \in \mathbb{A}^<$ and $b, d \notin \mathbb{A}^<$. Take $supp(X) = \mathbb{A}^< \cup \{b, d\}$, $supp(Y) = \mathbb{A}^< \cup \{f\}$, and $supp(Z) = \mathbb{A}^<$. Suppose a term-former $\mathtt{f}$.

We apply the algorithm to $\{\mathsf{f}([a]b,Z,X) \stackrel{?}{=} \mathsf{f}([d]b,[a]a,Y)\}$:

$\mathsf{f}([a]b,Z,X) \stackrel{?}{=} \mathsf{f}([d]b,[a]a,Y)$	$\Longrightarrow$	$(\stackrel{?}{=}\mathsf{f}),(\stackrel{?}{=}())$
$[a]b \stackrel{?}{=} [d]b$, $Z \stackrel{?}{=} [a]a$, $X \stackrel{?}{=} Y$	$\Longrightarrow$	$(\stackrel{?}{=}[]),[a]b=[d]b$
$b \stackrel{?}{=} b$, $Z \stackrel{?}{=} [a]a$, $X \stackrel{?}{=} Y$	$\Longrightarrow$	$(\stackrel{?}{=}\mathbf{a})$
$Z \stackrel{?}{=} [a]a$, $X \stackrel{?}{=} Y$	$\stackrel{[Z:=[a]a]}{\Longrightarrow}$	$(\mathbf{IE})$
$X \stackrel{?}{=} Y$	$\Longrightarrow$	$(\mathbf{F})$
$b\#_?X$, $X \stackrel{?}{=} Y$	$\stackrel{[X:=X\text{-}b]}{\Longrightarrow}$	$(\mathbf{IF})$
$X\text{-}b \stackrel{?}{=} Y$	$\Longrightarrow$	$(\mathbf{F})$
$d\#_?X\text{-}b$, $X\text{-}b \stackrel{?}{=} Y$	$\stackrel{[X\text{-}b:=X\text{-}b,d]}{\Longrightarrow}$	$(\mathbf{IF})$
$X\text{-}b,d \stackrel{?}{=} Y$	$\Longrightarrow$	$(\mathbf{F})$
$f\#_?Y$, $X\text{-}b,d \stackrel{?}{=} Y$	$\stackrel{[Y:=Y\text{-}f]}{\Longrightarrow}$	$(\mathbf{IF})$
$X\text{-}b,d \stackrel{?}{=} Y\text{-}f$	$\stackrel{[Y\text{-}f:=X\text{-}b,d]}{\Longrightarrow}$	$(\mathbf{IE})$
$\varnothing$ Success, with $[X:=X\text{-}b,d,\ Y:=X\text{-}b,d,\ Z:=[a]a]$		

Example three (fails).

Take $supp(X) = \mathbb{A}^<$. We run the algorithm on $\{[a][b]X \stackrel{?}{=} [a]X\}$:

$[a][b]X \stackrel{?}{=} [a]X$	$\Longrightarrow$	$(\stackrel{?}{=}[])$
$[b]X \stackrel{?}{=} X$ Failure		

The algorithm fails because the precondition of rule $(\mathbf{IE})$, $X \notin fU([b]X)$ is not satisfied.

Example four (succeeds).

Take $supp(X) = \mathbb{A}^<$ and take $a,b \in \mathbb{A}^<$. We run the algorithm on $\{X \stackrel{?}{=} (a\ b)\cdot X\}$:

$X \stackrel{?}{=} (a\ b)\cdot X$	$\Longrightarrow$	$(\stackrel{?}{=}\mathbf{X})$
$a\#_?X$, $b\#_?X$	$\stackrel{[X:=X\text{-}a]}{\Longrightarrow}$	$(\mathbf{IF})$
$b\#_?X\text{-}a$	$\stackrel{[X\text{-}a:=(X\text{-}a)\text{-}b]}{\Longrightarrow}$	
$\varnothing$ Success, with $[X:=(X\text{-}a)\text{-}b]$		

Later we will prove Theorem 2.118, which tells us that failure here implies that no solution to the unification problem exists.

2.3.1.3 Preservation of Solutions

...under non-instantiating rules

Lemma 2.107. *If $Pr \Longrightarrow Pr'$ by a non-instantiating rule (Definition 2.102) then $Sol(Pr) = Sol(Pr')$.*

Proof. The empty set cannot be simplified, so suppose $Pr = r \stackrel{?}{=} s, Pr'$ where the simplification rule acts on $r \stackrel{?}{=} s$. We consider two cases:

- *The case* $(\stackrel{?}{=}[])$. Suppose $Pr = [a]r \stackrel{?}{=} [a]s, Pr'$ and $[a]r \stackrel{?}{=} [a]s, Pr' \Longrightarrow r \stackrel{?}{=} s, Pr'$ by $(\stackrel{?}{=}[])$. By Definition 2.69 and properties of equality, $[a](r\theta) = [a](s\theta)$ if and only if $r\theta = s\theta$.
- *The case* $(\mathbf{F}())$. Suppose $Pr = a\#_?(r_1, \ldots, r_n), Pr'$ and suppose that $a\#_?(r_1, \ldots, r_n), Pr' \Longrightarrow a\#_?r_1, \ldots, a\#_?r_n, Pr'$ by $(\mathbf{F}())$. By Definition 2.69 and Lemma 2.53, $a \notin fa((r_1, \ldots, r_n)\theta)$ if and only if $a \notin fa(r_1\theta), \ldots, a \notin fa(r_n\theta)$. ■

Lemma 2.108. *Suppose $\theta(X) = \theta'(X)$ for all $X \in fU(Pr)$. Then $\theta \in Sol(Pr)$ if and only if $\theta' \in Sol(Pr)$.*

Proof. From Definition 2.100 it suffices to show that $r\theta = s\theta$ if and only if $r\theta' = s\theta'$, for every $(r \stackrel{?}{=} s) \in Pr$, and $a \notin fa(r\theta)$ if and only if $a \notin fa(r\theta')$, for every $(a\#_?r) \in Pr$. This is immediate using Lemma 2.73. ■

...under $(\mathbf{IE})$

Recall from Remark 2.68 the discussion of why we write $\pi{\cdot}X$ when we have chosen a representative element X of an equivalence class of unknowns under permutations.

Definition 2.109. Write $\theta{-}X$ for the substitution such that

$$\begin{array}{l}(\theta{-}X)(\pi{\cdot}X) = \pi{\cdot}X \\ (\theta{-}X)(Y) = \theta(Y) \quad \text{for all other } Y.\end{array}$$

In the right circumstances, a substitution θ can be factored as 'a part of θ that does not touch X' and 'a single substitution for X':

Theorem 2.110. *If $X\theta{=}s\theta$ and $X \notin fU(s)$ then*

$$\theta = [X{:=}s] \circ (\theta{-}X).$$

That is:

$$\theta(X) = X([X:=s]\circ(\theta - X)) \quad \textit{and}$$
$$\theta(Y) = Y([X:=s]\circ(\theta - X)).$$

Proof. We reason as follows:

$$\begin{array}{lll}
(\pi\cdot X)([X:=s]\circ(\theta - X)) = (\pi\cdot s)(\theta - X) & & \text{Definition 2.69, Lemma 2.76} \\
= (\pi\cdot s)\theta & & X \notin fU(s),\ \text{Lemma 2.73} \\
= (\pi\cdot X)\theta & & \text{Assumption} \\
 & & \\
Y([X:=s]\circ(\theta - X)) = Y(\theta - X) & & \text{Definition 2.69, Lemma 2.76} \\
= Y\theta & & \text{Definition 2.109}
\end{array}$$

■

...under (**IF**)

Definition 2.111. Suppose θ is a substitution. Suppose $a \in supp(X)$ and $a \notin fa(\theta(X))$. Let $\delta_{X\text{-}a}$ be a *shift* permutation bijecting $supp(X)$ with $supp(X)\setminus\{a\}$.

Define a substitution $\theta_{[X\text{-}a:=X]}(X)$ by:

- $(\theta_{[X\text{-}a:=X]})(\pi\cdot X) = (\pi\circ\delta^{-1}_{X\text{-}a})\cdot\theta(X)$.
- $(\theta_{[X\text{-}a:=X]})(Y) = \theta(Y)$ for all other Y.

It is routine to verify that Definition 2.111 is well-defined and a substitution.

Theorem 2.112. *Suppose $a \in supp(X)$ and $a \notin fa(\theta(X))$. Then*

$$\theta = [X:=X\text{-}a]\circ(\theta_{[X\text{-}a:=X]}).$$

That is:

$$\theta(\pi\cdot X) = ([X:=X\text{-}a]\circ\theta_{[X\text{-}a:=X]})(\pi\cdot X) \quad \textit{and}$$
$$\theta(Y) = ([X:=X\text{-}a]\circ\theta_{[X\text{-}a:=X]})(Y).$$

Proof. We unpack definitions:

$$\begin{array}{lll}
([X:=X\text{-}a]\circ(\theta_{[X\text{-}a:=X]}))(\pi\cdot X) = (\pi\cdot(X\text{-}a))\theta_{[X\text{-}a:=X]} & & \text{Definition 2.75} \\
= ((\pi\circ\delta_{X\text{-}a})\cdot X)\theta_{[X\text{-}a:=X]} & & \text{Def. } X\text{-}a \\
= (\pi\circ\delta_{X\text{-}a}\circ\delta^{-1}_{X\text{-}a})\cdot X & & \text{Definition 2.111} \\
= \pi\cdot X & & \text{Group action}
\end{array}$$

The result follows. ■

2.3.1.4 Simplification Rewrites Calculate Principal Solutions

Definition 2.113. Write $\theta_1 \leq \theta_2$ when there exists some θ' such that $X\theta_2{=}X(\theta_1 \circ \theta')$ always. Call $\leq$ the **instantiation ordering**.

Definition 2.114. A **principal** (or **most general**) solution to a problem Pr is a solution $\theta \in Sol(Pr)$ such that $\theta \leq \theta'$ for all other $\theta' \in Sol(Pr)$.

Our main result is Theorem 2.117: the unification algorithm from Definition 2.104 calculates a principal solution.

Lemma 2.115. *If $\theta_1 \leq \theta_2$ then $\theta \circ \theta_1 \leq \theta \circ \theta_2$.*

Proof. By Definition 2.113, θ' exists such that $X\theta_2{=}X(\theta_1 \circ \theta')$ always. Then:

$$\begin{array}{rll} X(\theta \circ \theta_2) & = (X\theta)\theta_2 & \text{Lemma 2.76} \\ & = (X\theta)(\theta_1 \circ \theta') & \text{Lemma 2.73} \\ & = X((\theta \circ \theta_1) \circ \theta') & \text{Lemma 2.76} \end{array}$$

■

Lemma 2.116.

1. *Suppose $fa(s) \subseteq supp(X)$ and $X \notin fU(s)$. Write $\chi{=}[X{:=}s]$. If $Pr \stackrel{\chi}{\Longrightarrow} Pr'$ with* (**IE**) *then $\theta \in Sol(Pr)$ implies $\theta{-}X \in Sol(Pr')$.*
2. *Suppose $a \in supp(X)$. Write $\rho{=}[X{:=}X\text{-}a]$. If $Pr \stackrel{\rho}{\Longrightarrow} Pr'$ with* (**IF**) *then $\theta \in Sol(Pr)$ implies $\theta_{[X\text{-}a:=X]} \in Sol(Pr')$.*

Proof.

1. Suppose $Pr = X \stackrel{?}{=} s,\ Pr''$ so that $X \stackrel{?}{=} s,\ Pr'' \stackrel{\chi}{\Longrightarrow} Pr''\chi$. Now suppose $\theta \in Sol(Pr)$. By Theorem 2.110 $\chi \circ (\theta{-}X) \in Sol(Pr)$. By Lemma 2.101, $\theta{-}X \in Sol(Pr\chi)$. It follows that $\theta{-}X \in Sol(Pr''\chi)$ as required.
2. Suppose $Pr = a\#_? X,\ Pr''$ and $a \in supp(X)$ so that $Pr \stackrel{\rho}{\Longrightarrow} Pr\rho$. Now suppose $\theta \in Sol(Pr)$. By Theorem 2.112 $\rho \circ \theta_{[X\text{-}a:=X]} \in Sol(Pr)$. By Lemma 2.101, $\theta_{[X\text{-}a:=X]} \in Sol(Pr\rho)$ as required.

■

Theorem 2.117. *If $Pr \stackrel{\theta}{\Longrightarrow}{}^* \varnothing$ then θ is a principal solution to Pr (Definition 2.114).*

Proof. By induction on the path of $Pr \stackrel{\theta}{\Longrightarrow}{}^* \varnothing$.

- *The empty path.* So $Pr = \varnothing$ and $\theta = id$. By Definition 2.113, $id \leq \theta'$.
- *The non-instantiating case.* Suppose

$$Pr \Longrightarrow Pr' \stackrel{\theta}{\Longrightarrow}{}^* \varnothing$$

where $Pr \Longrightarrow Pr'$ by a non-instantiating rule. By inductive hypothesis θ is a principal solution of Pr'. It follows from Lemma 2.107 that θ is also a principal solution of Pr.

- *The case* (**IE**). Suppose $fa(r) \subseteq supp(X)$ and $X \notin fU(r)$. Write $\chi = [X{:=}r]$. Suppose $Pr = r \stackrel{?}{=} X, Pr''$ so that

$$r \stackrel{?}{=} X,\ Pr'' \stackrel{\chi}{\Longrightarrow} Pr''\chi \stackrel{\theta''}{\Longrightarrow}{}^{*} \varnothing.$$

  Further, consider any other $\theta' \in Sol(Pr)$.

  By Lemma 2.116 $(\theta' {-} X) \in Sol(Pr''\chi)$ and by inductive hypothesis $\theta'' \in Sol(Pr''\chi)$ and $\theta'' \leq \theta' {-} X$. By Lemma 2.115, $\chi \circ \theta'' \leq \chi \circ (\theta' {-} X)$. By Theorem 2.110 $\chi \circ (\theta' {-} X) = \theta'$.
- *The case* (**IF**). Suppose $a \in supp(X)$. Write $\rho = [X{:=}X\text{-}a]$, so that

$$Pr \stackrel{\rho}{\Longrightarrow} Pr\rho \stackrel{\theta''}{\Longrightarrow}{}^{*} \varnothing,$$

  Further, consider any other $\theta' \in Sol(Pr)$.

  By Lemma 2.116, $\theta'_{[X\text{-}a:=X]} \in Sol(Pr\rho)$ and by inductive hypothesis $\theta'' \in Sol(Pr\rho)$ and $\theta'' \leq \theta'_{[X\text{-}a:=X]}$. By Lemma 2.115, $\rho \circ \theta'' \leq \rho \circ \theta'_{[X\text{-}a:=X]}$. By Theorem 2.112 $\rho \circ \theta'_{[X\text{-}a:=X]} = \theta'$.

■

Theorem 2.118 (Correctness of algorithm). *Given a problem Pr, if the algorithm of Definition 2.104 succeeds then it returns a principal solution; if it fails then there is no solution.*

Proof. If the algorithm succeeds we use Theorem 2.117. Otherwise, the algorithm generates an element of the form $\mathsf{f}(r) \stackrel{?}{=} \mathsf{g}(s)$, $a \stackrel{?}{=} b$, $a\#_? a$, $a\#_? C$ where $a \in supp(C)$, or $X \stackrel{?}{=} s$ where $X \in fU(s)$ and s is not of the form $\pi \cdot X$. By arguments on syntax and size of syntax, no solution to the reduced problem exists. It follows by Lemma 2.116 that no solution to Pr exists. ■

Definition 2.119. Fix terms r and s.

- Call **nominal unification** the problem of finding a θ to make $r\theta = s\theta$.
- Call **nominal matching** the problem of finding a θ to make $r\theta = s$.

Corollary 2.120. *Providing that equality of $\mathcal{C}$ (constants), $\mathcal{X}$ (unknowns), and $\mathbb{P}$ (permutations) are decidable, nominal unification and nominal matching over signatures using them are also decidable.*

Proof. An algorithm for unification is sketched in Definition 2.104; furthermore by Theorem 2.118 it calculates a most general θ which represents all other solutions.

For matching, we substitute unknowns in s with fresh (non-equivariant) constants of the same sorts and permission sets—we extend the signature if we need

to—and run the unification algorithm. We then replace the constants by the original unknowns.[11] It is not hard to see that this calculates a most general matching solution. ■

Remark 2.121. The matching and unification algorithms might generate solutions with *shift*-permutations. If we prefer to eliminate them then—provided that $\mathcal{X}$ has enough unknowns (Definition 2.48)—we may do so by appending an invertible substitution (Definition 2.77) mapping each shifted $\delta{\cdot}X$ in the solution to a fresh unknown Y such that $supp(Y) = \delta{\cdot}supp(X)$.

2.3.2 Rewriting

Nominal rewriting was the first logical system designed to study theories (sets of axioms, i.e. *rewrite rules*) over nominal terms. It was introduced by Fernández and the author in Fernández et al. (2004), Fernández and Gabbay (2007). Nominal terms allow us to express rewrite rules involving binding, like substitution and the λ-calculus (see Example 2.124).

The presentation of nominal rewriting here differs from that in Fernández and Gabbay (2007), and is more concise. Partly this is optimisation, but this is also due to the permissive-nominal approach. We compare and contrast nominal rewriting from Fernández and Gabbay (2007) with nominal rewriting here, in Sect. 2.3.2.6.

2.3.2.1 Rewrite Rules

Definition 2.122. A **rewrite rule** in a signature $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, ar)$ is a pair of terms $l \to m$ in Σ such that $sort(l) = sort(m) \in \mathcal{B}$ and $fU(m) \subseteq fU(l)$.
R will range over rewrite rules.
A **rewrite theory** $\mathsf{R} = (\Sigma, Rew)$ is a pair of a signature Σ (Definition 2.43) and a (possibly infinite) set of rewrite rules Rew in Σ.

Notation 2.123. Write $(l \to m) \in \mathsf{R}$ to mean 'l and m are terms in Σ and $(l \to m) \in Rew$'.

The notion of rewrite rule and rewrite theory in Definition 2.122 is much like the first-order case, but because of the 'nominal' aspects of our syntax we can handle names and binding.

[11] We do not make this formal, but since constants are structurally just like unknowns the definitions can easily be constructed by proceeding exactly as we did when we defined substitution for unknowns.

Example 2.124. Here are some example rewrite theories:

- nrSUB expresses the usual capture-avoiding substitution action on λ-calculus terms.

  Let Σ have a base sort τ and the following term-formers:

$$\mathsf{sub} : ([\nu]\tau,\tau)\tau \quad \mathsf{lam} : ([\nu]\tau)\tau \quad \mathsf{app} : (\tau,\tau)\tau \quad \mathsf{var} : (\nu)\tau$$

  Rewrite rules are as follows:

$$\begin{array}{llll}
(\mathbf{var}\rightarrow) & \mathsf{var}(a)[a{\mapsto}X] & \rightarrow\ X & \\
(\mathbf{var}\rightarrow') & \mathsf{var}(b)[a{\mapsto}X] & \rightarrow\ \mathsf{var}(b) & \\
(\mathsf{lam}\rightarrow) & \mathsf{lam}([a]X)[b{\mapsto}Y] & \rightarrow\ \mathsf{lam}([a](X[b{\mapsto}Y])) & (a{\not\in}supp(Y)) \\
(\mathsf{app}\rightarrow) & \mathsf{app}(X,X')[b{\mapsto}Y] & \rightarrow\ \mathsf{app}(X[b{\mapsto}Y],X'[b{\mapsto}Y]) &
\end{array}$$

  Here and in the next example we sugar $\mathsf{sub}([a]r,t)$ to $r[a{\mapsto}t]$. Every permission set contains b and every permission set contains a except for $supp(Y)$, as indicated above.
- nrLAM extends the previous theory with two more rewrites:

$$\begin{array}{llll}
(\beta\rightarrow) & (\lambda[a]Z)X & \rightarrow\ Z[a{\mapsto}X] & \\
(\eta\rightarrow) & \lambda[a](Ya) & \rightarrow\ Y & (a{\not\in}supp(Y))
\end{array}$$

Sugar $\mathsf{lam}(r)$ to λr, $\mathsf{app}(r,s)$ to rs, and $\mathsf{var}(a)$ to a. We anticipate Sect. 2.3.2.2 and sketch how one might rewrite $(\lambda[b](\lambda[a]ab))a$ to $\lambda[a']a'a$:

$$\begin{array}{rl}
(\lambda[b](\lambda[a]ab))a \rightarrow & (\lambda[a]ab)[b{\mapsto}a] \\
= & (\lambda[a']a'b)[b{\mapsto}a] \\
\rightarrow & \lambda[a']((a'b)[b{\mapsto}a]) \\
\rightarrow^* & \lambda[a']a'a
\end{array}$$

2.3.2.2 Rewrite Steps

Definition 2.125. Define the terms s in which X occurs **only once** by:

$$s ::= \pi{\cdot}X \mid [a]s \mid \mathsf{f}(r_1,\ldots,r_{i-1},s,r_{i+1},\ldots,r_n)$$
$$(X \not\in fU(r_1),\ldots,fU(r_{i-1}),fU(r_{i+1}),\ldots,fU(r_n))$$

A **position** P is a pair (s,X) of a nominal term and an unknown X which occurs only once in s.

Our notion of *position* is also sometimes called a **context**; the idea goes back to at least Felleisen and Hieb (1992).

In Definition 2.125, $\pi{\cdot}X$ denotes an unknown in the same permutation orbit as X.

Notation 2.126. If $P=(s,X)$ is a position write $supp(P)$ for $supp(X)$ and $sort(P)$ for $sort(X)$.

If $sort(r) = sort(P)$ and $fa(r) \subseteq supp(P)$ (so that $[X{:=}r]$ is a substitution) write $P[r]$ for $s[X{:=}r]$.

Definition 2.127. The **one-step rewrite relation** $r \xrightarrow{\mathsf{R}} s$ is the least relation such that for every $(l \to m) \in \mathsf{R}$, position P, and substitution θ, if $sort(r) = sort(P)$ and $fa(l\theta) \cup fa(m\theta) \subseteq supp(P)$ (so that $P[l\theta]$ and $P[m\theta]$ are well-defined) then

$$P[l\theta] \xrightarrow{R} P[m\theta].$$

The **multi-step rewrite relation** $r \xrightarrow{\mathsf{R}}\!\!\!\!\!\!\ast\ \ s$ is the reflexive transitive closure of the one-step rewrite relation.

We consider decidability and complexity of the rewrite relation in Sect. 2.3.3.

Example 2.128. Let T have one name sort ν, one base sort τ, one term-former triv and one axiom $\mathsf{triv}(a) \to \mathsf{triv}(b)$.

Then $\mathsf{triv}(a) \to \mathsf{triv}(b)$ but also (using positions $(\pi{\cdot}X, X)$ for any π) $\mathsf{triv}(b) \to \mathsf{triv}(a)$ and $\mathsf{triv}(a') \to \mathsf{triv}(b')$ for any pair of distinct atoms a' and b'.

Thus atoms in rewrite rules range over 'any atom' analogously to how unknowns in rewrite rules range over 'any term'.

Example 2.129. Recall the rule $(\eta{\to}) = (\lambda[a](Ya) \to Y)$ where $a \notin supp(Y)$ from Example 2.124. Suppose also $b \notin supp(Y)$.

1. To deduce $\lambda[a](ba) \to b$ we take $P = ((b\ c){\cdot}Y, Y)$ for some $c \in supp(Y)$ and we take $\theta = [Y{:=}c]$.
2. To deduce $\lambda[a'](ba') \to b$ for any other a' we *also* take $P = ((b\ c){\cdot}Y, Y)$ and $\theta = [Y{:=}c]$. This is because $\lambda[a'](ba')$ and $\lambda[a](ba)$ are the same term (Lemma 2.31).
3. To deduce $\lambda[a](Ya) \to Y$ we take $P = (Y, Y)$ and $\theta = id$.
4. Suppose $supp(Y') = supp(Y) \cup \{a\}$.

   Suppose we have *shift*-permutations so there exists a permutation, write it $\delta_{Y'\text{-}a}$, bijecting $supp(Y')$ with $supp(Y)$. To deduce $\lambda[a](Y'a) \to Y'$ we take $P = ((\delta_{Y'\text{-}a})^{-1}{\cdot}Y, Y)$ and $\theta = [Y{:=}\delta_{Y'\text{-}a}{\cdot}Y']$.

   Without *shift* we cannot deduce $\lambda[a](Y'a) \to Y'$; we can still deduce $\lambda[a](Ya) \to Y$.
5. We cannot deduce $\lambda[a](aa) \to a$, because $[Y{:=}a]$ is not a substitution: no function mapping Y to a can be equivariant, since $(b\ a){\cdot}Y = Y$ but $(b\ a){\cdot}a = b \neq a$ (also $a \notin supp(Y)$: see Proposition 2.64).
6. A rewrite $X \to X$ only entails rewrites for t with $fa(t) \subseteq \pi{\cdot}supp(X)$ for some π. With *shift*, the effect of this may be that we can deduce $t \to t$ from $X \to X$ for any t. We make no claim to there being a 'right' or 'wrong' answer here: the issue is purely a design question of how much expressivity we want permutations to have. Our results are parameterised over this choice.

Definition 2.130.

- Call R **locally confluent** when $r \xrightarrow{\mathsf{R}} s_1$ and $r \xrightarrow{\mathsf{R}} s_2$ implies there exists some s' such that $s_1 \xrightarrow{\mathsf{R}}{}^{*} s'$ and $s_2 \xrightarrow{\mathsf{R}}{}^{*} s'$.
- Call R **confluent** when $r \xrightarrow{\mathsf{R}}{}^{*} s_1$ and $r \xrightarrow{\mathsf{R}}{}^{*} s_2$ implies there exists some s' such that $s_1 \xrightarrow{\mathsf{R}}{}^{*} s'$ and $s_2 \xrightarrow{\mathsf{R}}{}^{*} s'$.

2.3.2.3 Peaks, Critical Pairs, Joinability

We now begin to investigate criteria for deducing confluence of nominal rewrite systems. Our first observation is that things are not quite as simple as in first-order rewriting (Baader and Nipkow 1998, Section 6.2): by Lemma 2.135, trivial critical pairs are not always joinable.

Definition 2.131. Write $r \to s_1, s_2$ when $r \to s_1$ and $r \to s_2$ and call this a **peak**. Call this peak **joinable** when there exists a t such that $s_1 \twoheadrightarrow t$ and $s_2 \twoheadrightarrow t$.

So R is locally confluent when every peak is joinable.

Definition 2.132. Consider two rewrite rules $R_1 = (l_1 \to m_1)$ and $R_2 = (l_2 \to m_2)$. Call R_1 a **copy** of R_2 when there exists an invertible substitution θ such that $(l_2\theta \to m_2\theta) = R_1$.

Clearly, if R_1 is a copy of R_2 then R_2 is also a copy of R_1. Furthermore:

Lemma 2.133. *If R_1 and R_2 are copies of the same rule then $l \xrightarrow{R_1} m$ if and only if $l \xrightarrow{R_2} m$.*

Proof. Unpacking Definition 2.127 and exploiting the existence of an inverse θ^{-1}. ■

Definition 2.134. Suppose that $R_i = (l_i \to m_i)$ for $i = 1,2$ and $fU(R_1) \cap fU(R_2) = \varnothing$. Suppose $l_1 = P[l_1']$ for some l_1', and suppose $l_1' \stackrel{?}{=} l_2$ has a principal solution θ. Call the pair $(m_1\theta, P[m_2]\theta)$ a **critical pair**.

Call $(m_1\theta, P[m_2]\theta)$ **trivial** when at least one of the following hold:

1. $P = (\pi \cdot X, X)$ and R_1 and R_2 are copies of the same rule.
2. $l_1' = X$ for some unknown X.

Lemma 2.135. *Peaks that are instances of trivial critical pairs, are not always joinable.*

Proof. It suffices to provide a counterexample. Fix term-formers $\mathsf{0}$ and f and take $R_1 = (\mathsf{0} \to a)$ and $R_2 = (X \to \mathsf{f}(a))$ where $a \notin supp(X)$.

There is a critical pair $(a, \mathsf{f}(a))$ between R_1 and R_2.

Also, $\mathsf{0} \xrightarrow{R_1} a$ and $\mathsf{0} \xrightarrow{R_2} \mathsf{f}(a)$ and it is a fact that this peak cannot be joined—we 'want' to close this peak by rewriting a to $\mathsf{f}(a)$ using R_2, but the fact that $a \notin supp(X)$ blocks this. ■

2.3.2.4 Uniform Rewriting

The proof of Lemma 2.135 suggests a simple cure:

Definition 2.136. Call a rule $R = (l \to m)$ **uniform** when

$$fa(m) \subseteq fa(l).$$

Call a rewrite theory R **uniform** when every $R \in \mathsf{R}$ is uniform.

Definition 2.136 mirrors the condition in Definition 2.122 that $fU(m) \subseteq fU(l)$, but for atoms instead of unknowns. This condition is sufficient to obtain Theorem 2.142, which is a nominal rewriting version of the well-known critical pair lemma from first-order rewriting (Baader and Nipkow 1998, Theorem 6.2.4).

Example 2.137. Let R have one name sort ν, one base sort τ, two term-formers $\mathsf{triv} : (\nu)\tau$ and $\mathsf{abs} : ([\nu]\tau)\tau$, and rewrite rules

$$\mathsf{triv}(a) \to \mathsf{triv}(a) \qquad \mathsf{triv}(a) \to \mathsf{triv}(b) \qquad \mathsf{abs}([a]X) \to X.$$

$fa(\mathsf{triv}(a)) \subseteq fa(\mathsf{triv}(a))$ and $fa(\mathsf{triv}(b)) \not\subseteq fa(\mathsf{triv}(a))$. Also $fa(X) \not\subseteq fa(\mathsf{abs}([a]X))$ if and only if $a \notin supp(X)$.

So the first rule is uniform, the second is not, and the third is uniform if and only if $a \notin supp(X)$.

The rewrite rules of nrSUB and nrLAM in Example 2.124 are uniform.[12]

Lemma 2.138. *If $fa(m) \subseteq fa(l)$ then $fa(P[m]) \subseteq fa(P[l])$.*

Proof. Routine induction using Lemmas 2.56 and 2.53. ■

Corollary 2.139. *$R = (l \to m)$ is uniform if and only if $\forall r,s.\big(r \xrightarrow{R} s \Rightarrow fa(s) \subseteq fa(r)\big)$.*

Proof. From Lemmas 2.56 and 2.72. ■

Lemma 2.140. *Suppose $R = (l \to m)$ is uniform and $X \notin fU(R)$. Suppose $\theta(X) = l\theta$. Specify θ' by $\theta'(\pi{\cdot}X) = \pi{\cdot}(m\theta)$ and $\theta'(Y) = \theta(Y)$. Then $r\theta \twoheadrightarrow r\theta'$ for any r.*

Proof. θ' is a substitution by Lemmas 2.72 and 2.56. The result follows by a routine induction on r. ■

Because of Lemma 2.133, we can be relaxed about the particular (orbits of) unknowns that are used in a rewrite rule, if we only care about the rewrites that they generate. We do this in Theorems 2.141 and 2.142. This can always be made formal by inserting invertible 'freshening' substitutions as appropriate.

[12] There is a deeper reason for this: they are also closed. See Example 2.164 and Theorem 2.165.

Theorem 2.141. *If a rewrite theory* R *(Definition 2.122) is uniform then peaks that are instances of trivial critical pairs, are joinable.*

Proof. Consider two rules $R_i = (l_i \to m_i) \in \mathsf{R}$ for $i = 1,2$. Taking copies if necessary, suppose $fU(R_1) \cap fU(R_2)$. Suppose they have a critical pair $(m_1\theta, P[m_2]\theta)$. That is, there exists l_1' such that $l_1 = P[l_1']$ and θ is a principal solution to $l_1' \stackrel{?}{=} l_2$.

There are two cases:

- *The case* $P = (\pi{\cdot}X, X)$ *and* R_1 *and* R_2 *are copies of the same rule* $l \to m$. The peak we want to join is $l_1\theta = \pi{\cdot}l_2\theta \to m_1\theta,\ \pi{\cdot}m_2\theta$, where the rules $l_1 \to m_1$ and $l_2 \to m_2$ are identical aside from their free unknowns which are renamed disjoint. We use Lemma 2.73 and the assumption in Definition 2.122 that $fU(m) \subseteq fU(l)$.
- *The case of* $(m_1\theta, P[m_2]\theta)$ *where* $l_1 = P[X]$ *and* $\theta(X) = l_2$. Specify θ' by $\theta'(\pi{\cdot}X) = \pi{\cdot}m_2$ and $\theta'(Y) = \theta(Y)$ for all other Y; note that θ' *is* a substitution since $fa(m_2) \subseteq fa(l_2)$ by uniformity and $fa(l_2) \subseteq supp(X)$ by our assumption that θ is a substitution.

  By Lemma 2.140 $m_1\theta \twoheadrightarrow m_1\theta'$. By definition $P[m_2]\theta = l_1\theta' \xrightarrow{R_1} m_1\theta'$, so we have joined the peak.

■

Theorem 2.142. *Suppose all non-trivial critical pairs of* R *are joinable* and *suppose* R *is uniform. Then* R *is locally confluent.*

Proof. Suppose $r \xrightarrow{R_1} s_1$ and $r \xrightarrow{R_2} s_2$. Write P_1 and P_2 for the positions at which the two rewrites occur. Taking copies if necessary, suppose $fU(R_1) \cap fU(R_2) = \varnothing$.

If P_1 and P_2 identify distinct subterms of r then local confluence holds by a standard diagrammatic argument (see for instance Baader and Nipkow (1998)).

Otherwise it must be that $P_2 = (P_1[P], X)$ for some position P; that is, P_2 identifies a point in r beneath the point identified by P_1 (or the symmetric case that $P_1 = (P_2[P], X)$, which is similar and we elide). There are now three possibilities:

1. X in P_2 replaces an unknown in r. This is an instance of a trivial critical pair; we use Theorem 2.141.
2. $P = (\pi{\cdot}X, X)$ and R_1 and R_2 are copies of the same rule. Then again this is an instance of a trivial critical pair and we use Theorem 2.141.
3. Otherwise, this is an instance of a non-trivial critical pair at it may be joined using our assumption that non-trivial critical pairs are joinable.

■

Definition 2.143. Call a rewrite system R **terminating** when all rewrite sequences are finite. Call a term r a **normal form** (with respect to a rewrite system R) when $\forall s.\neg(r \xrightarrow{\mathsf{R}} s)$, that is, when r does not R-rewrite to anything.

Example 2.144. It can be proved that nrSUB in Example 2.124 is terminating. nrLAM (famously) is not terminating, because of $(\beta\mapsto)$.

Corollary 2.145. *Suppose* R *is terminating, uniform, and suppose non-trivial critical pairs in* R *are joinable. Then:*

1. R *is confluent.*
2. *If* $r \twoheadrightarrow s$ *and* $r \twoheadrightarrow s'$ *and* s *and* s' *are normal forms, then* $s = s'$.

2.3.2.5 Orthogonal Rewrite Systems

We now treat another standard criterion in rewriting: orthogonality Dershowitz and Jouannaud (1989), Baader and Nipkow (1998). By Theorem 2.152 orthogonality implies not only local confluence, but the stronger property of confluence (Definition 2.130). The proof is not direct: it turns out that it is easier to consider an auxilliary *parallel reduction* relation $\Rightarrow$ (Definition 2.149). The reflexive transitive closure of $\Rightarrow$ is equal to that of $\rightarrow$ (Lemma 2.150), but $\Rightarrow$ allows (intuitively) multiple reductions provided that they do not occur 'one after the other, in the same position'. This is the kind of multiple reduction generated in the second case of the proof of Theorem 2.141, when we rewrite $m_1\theta$ to $m_1\theta'$.

Definition 2.146. Call $R = (l \rightarrow m)$ **left-linear** when each unknown occurring in l occurs only once (Definition 2.125).

For example $\mathsf{f}(X) \rightarrow \mathsf{g}(X,X)$ is left-linear but $\mathsf{g}(X,X) \rightarrow \mathsf{f}(X)$ and $\mathsf{g}(\pi{\cdot}X,x) \rightarrow \mathsf{f}(X)$ are not. Note that $(a,a) \rightarrow a$ is left-linear.

Definition 2.147. Call R **orthogonal** when every $R \in \mathsf{R}$ is uniform and left-linear, and all critical pairs are trivial.

(Note that we insist that R is uniform, as well as the standard condition that it be left-linear.)

Definition 2.148. Suppose $R = (l \rightarrow m)$. Write $r \xrightarrow[R]{}_{\varepsilon} s$ when $r \xrightarrow[R]{} s$ and the rewrite occurs at a position $P = (\pi{\cdot}X, X)$. We say that the rewrite with R occurs at **root position**.

Expanding Definition 2.148, $r \xrightarrow[R]{}_{\varepsilon} s$ when there exists θ and π such that $r = \pi{\cdot}(l\theta)$ and $s = \pi{\cdot}(m\theta)$. For example: if $R = (a \rightarrow a)$ then $a \xrightarrow[R]{}_{\varepsilon} a$ but not $[a]a \xrightarrow[R]{}_{\varepsilon} [a]a$.

Definition 2.149. We define a **parallel reduction** relation $\Rightarrow$ by the rules in Fig. 2.3.

Lemma 2.150. $r \twoheadrightarrow s$ *if and only if* $r \Rightarrow s$.

Proof. By routine inductions. ■

$$\frac{r_1 \Rightarrow s_1 \ \cdots \ r_n \Rightarrow s_n}{\mathsf{f}(r_1,\ldots,r_n) \Rightarrow \mathsf{f}(s_1,\ldots,s_n)}\ (\Rightarrow\mathsf{f})$$

$$\frac{r_1 \Rightarrow s_1 \ \cdots \ r_n \Rightarrow s_n \quad \mathsf{f}(s_1,\ldots,s_n) \xrightarrow{R}_{\varepsilon} s'}{\mathsf{f}(r_1,\ldots,r_n) \Rightarrow s'}\ (\Rightarrow\mathsf{f}')$$

$$\frac{s \Rightarrow t}{[a]s \Rightarrow [a]t}\ (\Rightarrow\mathbf{abs}) \qquad \frac{r \Rightarrow s \quad [a]s \xrightarrow{R}_{\varepsilon} s'}{[a]r \Rightarrow s'}\ (\Rightarrow\mathbf{abs}')$$

$$\frac{}{r \Rightarrow r}\ (\mathbf{refl}) \qquad \frac{a \xrightarrow{R}_{\varepsilon} s'}{a \Rightarrow s'}\ (\Rightarrow\mathbf{a}') \qquad \frac{X \xrightarrow{R}_{\varepsilon} s'}{X \Rightarrow s'}\ (\Rightarrow\mathbf{X}')$$

Fig. 2.3 Parallel reduction relation

Lemma 2.151. *If* R *is orthogonal then* $\Rightarrow$ *is confluent.*

Proof. We prove by induction on the derivation of $r \Rightarrow s$ that a stronger property holds, often called the **diamond property**: for all s' if $r \Rightarrow s'$ then there exists some s'' such that $s \Rightarrow s''$ and $s' \Rightarrow s''$. From this, confluence easily follows by a standard diagrammatic argument.

We consider a selection of cases:

- *The derivations of* $r \Rightarrow s$ *and* $r \Rightarrow s'$ *both end in* $(\Rightarrow\mathsf{f})$. We use the inductive hypotheses and $(\Rightarrow\mathsf{f})$.
- *The derivation of* $r \Rightarrow s$ *ends in* $(\Rightarrow\mathsf{f})$ *and that of* $r \Rightarrow s'$ *ends in* $(\Rightarrow\mathsf{f}')$. So $r_i \Rightarrow s_i$ and $r_i \Rightarrow s'_i$ for $1 \leq i \leq n$, and $\mathsf{f}(s'_1,\ldots,s'_n) = \pi\cdot(l\theta) \xrightarrow{R}_{\varepsilon} \pi\cdot(m\theta)$ for some π and $R = (l \to m) \in \mathsf{R}$. By inductive hypothesis there exist s''_i such that $s_i \Rightarrow s''_i$ and $s'_i \Rightarrow s''_i$. We now proceed as illustrated and explained below:

$$\begin{array}{ccccc}
\mathsf{f}(r_1,\ldots,r_n) & \Longrightarrow & \mathsf{f}(s'_1,\ldots,s'_n) & = \pi\cdot(l\theta) & \xrightarrow{R_\varepsilon} \pi\cdot(m\theta) \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{f}(s_1,\ldots,s_n) & \Longrightarrow & \mathsf{f}(s''_1,\ldots,s''_n) & = \pi\cdot(l\theta') & \xrightarrow{R_\varepsilon} \pi\cdot(m\theta')
\end{array}$$

Either l is an unknown X or the rewrite $\mathsf{f}(s'_1,\ldots,s'_n) \Rightarrow \mathsf{f}(s''_1,\ldots,s''_n)$ takes place in the substitution θ.

If l is an unknown then by uniformity we may rewrite $f(s''_1,\ldots,s''_n)$ using R and close the diagram by rewriting corresponding instances of $\theta(X)$ in $\pi\cdot(m\theta)$.

Otherwise, by uniformity there is a substitution θ' such that $\theta(X) \Rightarrow \theta'(X)$ for every X and $\mathsf{f}(s''_1,\ldots,s''_n) = \pi\cdot(l\theta')$. Rules are also left-linear so R still applies to $\pi\cdot(l\theta)$: $\mathsf{f}(s''_1,\ldots,s''_n) \xrightarrow{R}_{\varepsilon} \pi\cdot(m\theta')$ and therefore $\mathsf{f}(s_1,\ldots,s_n) \Rightarrow s\theta'$ by $(\Rightarrow\mathsf{f}')$ for R.

The other cases are no harder. ■

Theorem 2.152. *If a theory* R *is orthogonal (Definition 2.147) then* R *is confluent (Definition 2.130).*

Proof. If the uniform rewrite system has only left-linear rules and only trivial critical pairs, then $\Rightarrow$ is confluent by Lemma 2.151. It follows that $\Rightarrow^*$ is confluent. By Lemma 2.150 the result follows. ■

2.3.2.6 Nominal Rewriting with Freshness Contexts Versus Permissive-Nominal Rewriting

As mentioned in the introduction to this Section, the presentation of this paper differs from that of Fernández and Gabbay (2007) in being permissive-nominal.

For clarity, let us call the nominal rewrite framework from Fernández and Gabbay (2007) 'System ∇' and the nominal rewrite framework here 'System S'.

In system ∇ a rewrite rule takes the form $\nabla \vdash t \to u$ where ∇ is a set of assumptions $a\#\mathtt{X}$ called a *freshness context*. $\mathtt{X}$ is an *unknown*. This is not typed by a permission set; freshness information is given by ∇.

Here are $(\lambda\to)$ from Example 2.124, and how it would look in System ∇:

$$\begin{array}{lllll} \text{System S} & \mathsf{lam}([a]X)[b{\mapsto}Y] & \to & \mathsf{lam}([a](X[b{\mapsto}Y])) & (a{\notin}supp(Y)) \\ \text{System } \nabla \quad a\#\mathtt{Y}\vdash & \mathsf{lam}([a]\mathtt{X})[b{\mapsto}\mathtt{Y}] & \to & \mathsf{lam}([a](\mathtt{X}[b{\mapsto}\mathtt{Y}])) & \end{array}$$

$a \notin supp(Y)$ is a fact (we must choose Y so that this is true). It does not matter *which* permission set we give Y because using δ and swappings we can build a π to map $supp(Y)$ to every other permission set $\pi\cdot supp(Y)$—which will contain $\pi(a)$.

Conversely $a\#\mathtt{Y}$ is a freshness condition. It directly controls the terms to which we may instantiate $\mathtt{Y}$; they must not contain a free. Here we attain this effect using Proposition 2.64.

Freshness conditions are elementary: they mean what they say and what they mean be quickly understood. Permission sets are still finitely representable, but somewhat harder to understand. So from the point of view of keeping a gentle learning curve, System ∇ may be preferable to System S.

However, System S rewards us with some advantages: we can use nominal abstract syntax and the freshness conditions which must be explicitly stated (repeatedly) in Fernández and Gabbay (2007) are handled in the background by equivariance of substitutions (as Proposition 2.64 makes formal).

This also has some effects on mathematical properties. In System ∇ from Fernández and Gabbay (2007) it was not in general the case that if $\nabla \vdash r \approx_\alpha r'$ and $\nabla \vdash r \xrightarrow{R} s$ then $\nabla \vdash r' \xrightarrow{R} s$ (see the end of Section 5.2 in Fernández and

Gabbay (2007)). It was also not in general the case that nominal rewriting coincides with nominal algebra (Sect. 2.3.4), essentially because any fixed freshness contexts might not be 'big enough'. Fernández and the author wrote a paper on how to adjust for this Fernández and Gabbay (2010). In a permissive-nominal context, these issues do not arise in the first place.

This author's feeling is that nominal-terms-with-freshness-contexts and permissive-nominal terms can be considered as essentially the same thing. However, if our goal is to prove theorems then we get closer to what is 'really going on' via the permissive-nominal presentation.

2.3.3 Closed Terms

Equivariant unification—the problem of finding θ *and* π such that $\pi{\cdot}(r\theta) = s\theta$—is NP complete Cheney (2004, 2010). The same applies to corresponding matching problems. This matters to us because the rewrite relation in Definition 2.127 is equivariant; to determine whether r rewrites with a rule $(l \to r)$, we must solve an equivariant matching problem.

Fernández and the author introduced a notion of *closed term* such that for closed terms, equivariant matching/unification coincides with 'ordinary' matching/unification Fernández and Gabbay (2007). That is, for closed terms we can throw away the π.

We now develop corresponding definitions and results. The definitions and proofs in this paper are significantly different from those in Fernández and Gabbay (2007).[13]

2.3.3.1 The Definition

Definition 2.153. Define **explicit atoms** $ea(r)$ inductively by:

$$\begin{array}{ccc} ea(a) = \{a\} & ea(C) = supp(C) & ea(X) = \varnothing \\ ea(\mathtt{f}(r)) = ea(r) & ea((r_1,\ldots,r_n)) = \bigcup ea(r_i) & ea([a]r) = ea(r)\backslash\{a\} \end{array}$$

Remark 2.154. Intuitions for $ea(r)$ versus $fa(r)$ are as follows:

[13]The interested reader can begin by comparing our notion of closed terms in Definition 2.159, based on two simpler inductive definitions, with that used in (Fernández and Gabbay 2007, Definition 68), based on a renamed variant of a term and an equality derivable in an extended freshness context. See also an inductive characterisation of closed terms in unpublished notes Clouston (2007).

- The explicit atoms of r are the atoms that actually appear in r (unbound). That is, we can read '$a \in ea(r)$' as 'a appears in r'.
- The free atoms of r are the atoms that can appear in $r\theta$ for some θ.

For instance, $ea(X) = \varnothing \neq supp(X) = fa(X)$.

This is an intuition, not a fact. $fa(r) = \bigcup_\theta ea(r\theta)$ is not true in general (but see Lemma 2.157). For instance in a signature with one base sort τ and no term formers, terms containing atoms simply do not populate the sort τ.

Recall the notion of *occurrences* $occ(r)$ from Definition 2.95.

Notation 2.155. Write $\pi{\cdot}occ(r) = \{\pi{\cdot}x \mid x \in occ(r)\}$. Also if $D = [d_1,\ldots,d_n]$ and S is a permission set define $S \setminus D = S \setminus \{d_1,\ldots,d_n\}$.

Lemma 2.156. *$ea(\pi{\cdot}r) = \pi{\cdot}ea(r)$ and $occ(\pi{\cdot}r) = \pi{\cdot}occ(r)$. In addition, $ea(r) \subseteq ea(r\theta)$.*

Proof. By routine inductions on r. ■

Lemma 2.157. *$fa(r) = ea(r) \cup \bigcup\{supp(x) \mid x \in occ(r)\}$.*

As an easy corollary using Lemma 2.53, $fa(r) = ea(r) \cup \bigcup\{supp(X) \setminus D \mid [D]X \in occ(r)\}$.

Proof. By a routine induction on r. We consider one case:

- *The case $[a]r$.* Suppose $fa(r) = ea(r) \cup \bigcup\{supp(x) \mid x \in occ(r)\}$. By definition $fa([a]r) = fa(r) \setminus \{a\}$, and $ea([a]r) = ea(r) \setminus \{a\}$ and $occ([a]r) = \{[a]x \mid x \in occ(r)\}$. The result follows by an easy sets calculation.

■

Definition 2.158. Call r *fa*-**functional** when if $[D_1]X \in occ(r)$ and $[D_2]X \in occ(r)$ then $fa([D_1]X) = fa([D_2]X)$ (equivalently, when D_1 and D_2 contain the same atoms but not necessarily in the same order).

Definition 2.159. Call r **closed** when r is *fa*-functional and $ea(r) = \varnothing$.

Example 2.160.
- a is not closed (*ea* is non-empty).
- X is closed, so note that 'closed' does not mean '$fU(r) = \varnothing$'. Our terminology is consistent with Fernández and Gabbay (2007) and the subsequent literature.
- $([a]X, X)$ is not closed (*occ* is not *fa*-functional).
- $[a](X, a)$ is closed.

Lemma 2.161. *Suppose $ea(r) = \varnothing$. Then $\pi{\cdot}(r\theta) = r\theta'$ if and only if $\pi{\cdot}(([D]X)\theta) = ([D]X)\theta'$ for every $[D]X \in occ(r)$.*

Proof. By a routine induction on r. ■

Theorem 2.162. *r is closed if and only if*

$$\exists S.fa(r) \subseteq S \wedge \forall \pi, \theta.\pi{\cdot}fa(r\theta) \subseteq S \Rightarrow \exists \theta'.\pi{\cdot}(r\theta) = r\theta'.$$

Proof. Suppose there is a permission set $S \supseteq fa(r)$ such that if $\pi{\cdot}fa(r\theta) \subseteq S$ then there exists θ' such that $\pi{\cdot}(r\theta) = r\theta'$. There are two things to prove:

- *$ea(r)$ is empty.* Suppose there exists $a \in ea(r)$. Pick $b \in S \setminus ea(r)$. By assumption taking $\theta = id$ there exists θ' such that $(b\ a){\cdot}(r\theta) = r\theta'$. By Lemma 2.156 $ea((b\ a){\cdot}r) = (b\ a){\cdot}ea(r) \not\ni a$ and $a \in ea(r) \subseteq ea(r\theta')$, a contradiction.
- *$occ(r)$ is fa-functional.* Consider $[D_1]X$ and $[D_2]X$ in $occ(r)$; choose D_i such that $D_i \cap fa(r) = \varnothing$ for $i = 1,2$. Suppose there exists $a \in fa([D_2]X) \setminus fa([D_1]X)$, and choose any $b \in fa([D_1]X)$ (since $supp(X)$ is infinite and D_1 is finite, such a b exists).

  By Lemma 2.157 $a,b \in fa(r)$ so by assumption taking $\theta = id$ there exists θ' such that $(b\ a){\cdot}r = r\theta'$. We proved above that $ea(r) = \varnothing$, so by Lemma 2.161 $(b\ a){\cdot}[D_1]X = ([D_1]X)\theta$. By Lemma 2.56 a is free in the left-hand side, and by Lemma 2.72 a is not free in the right-hand side; a contradiction.

Suppose $occ(r)$ is *fa*-functional and $ea(r) = \varnothing$ and choose some permutation π and substitution θ.

If $occ(r) = \varnothing$ then by Lemma 2.157 $fa(r) = \varnothing$ so by Lemmas 2.57 and 2.73 $\pi{\cdot}(r\theta) = r$ and $r\theta' = r$, so there is nothing to prove.

Otherwise take $S = fa(r)$. For every element of in $occ(r)$ make a fixed but arbitrary choice of representation as $[D]X$ where the atoms in D are disjoint from the atoms in $nontriv(\pi)$. We take θ' to equivariantly extend this choice (Definition 2.37), so we map $\pi'{\cdot}X$ to $(\pi' \circ \pi){\cdot}\theta(X)$ for the choice of representing X above, and otherwise to map Y to Y. Using Proposition 2.38 this is a substitution and $\pi{\cdot}(([D]X)\theta) = ([D]X)\theta'$ for every $[D]X \in occ(r)$. We use Lemma 2.161. ■

2.3.3.2 Closed Rewrite Rules

Definition 2.163. Call a rewrite rule $l \to m$ **closed** when (l,m) is closed.

Example 2.164. Let R have one name sort ν, one base sort τ, two term-formers $\mathsf{triv} : (\nu)\tau$ and $\mathsf{abs} : ([\nu]\tau)\tau$, and rewrite rules

$$\mathsf{triv}(a) \to \mathsf{triv}(a) \qquad \mathsf{triv}(a) \to \mathsf{triv}(b) \qquad \mathsf{abs}([a]X) \to X.$$

The terms $\mathsf{triv}(a)$ and $\mathsf{triv}(b)$ are not closed; the terms $\mathsf{abs}([a]X)$ and X are closed. The terms $(\mathsf{triv}(a),\mathsf{triv}(a))$ and $(\mathsf{triv}(a),\mathsf{triv}(b))$ are not closed. The term $(\mathsf{abs}([a]X),X)$ is closed if and only if $a \notin supp(X)$. So the first two rules are not closed and the third is closed if and only if $a \notin supp(X)$.

The rewrite rules of nrSUB and nrLAM in Example 2.124 are closed.

Recall that uniform rules have good properties like Theorems 2.142 and 2.152. Closed rules inherit these good properties, because:

Theorem 2.165. *If $R = (l \to m)$ is closed then it is uniform.*

Proof. By assumption $fU(m) \subseteq fU(l)$. Also (l,m) is fa-functional; it follows that $occ(m) \subseteq occ(l)$. The result follows from Lemma 2.157. ■

Lemma 2.166. *Suppose r and l are terms and l is closed. Then*

1. $\exists \pi, \theta . r = \pi \cdot (l\theta)$ *implies*
2. $\forall \pi . fa(r) \subseteq \pi \cdot fa(l) \Rightarrow \exists \theta . r = \pi \cdot (l\theta)$

Proof. Suppose $fa(r) \subseteq \pi \cdot fa(l)$ and $fa(r) \subseteq \pi' \cdot fa(l)$ and $r = \pi \cdot (l\theta)$.

We need a θ' such that $r = \pi' \cdot (l\theta')$. It follows from the above that $(\pi'^{-1} \circ \pi) \cdot fa(l\theta) \subseteq fa(l)$. We use Theorem 2.162. ■

Theorem 2.167. *If* R *is closed then* $\xrightarrow{\mathsf{R}}$ *can be checked as follows, where for simplicity we suppose* $\mathsf{R} = \{(l \to m)\}$*:*

1. *We try to match r against $\pi \cdot l$ for some π such that $fa(r) \subseteq \pi \cdot fa(l)$, if such a π exists.*
2. *If we fail then by Lemma 2.166 we must fail for instantiating for any $\pi \cdot l$. We descend into subterms of r and repeat the previous step.*

Whether step 1 of the algorithm above is decidable depends on the decidability of $\mathbb{P}$, $\mathcal{X}$, and $\mathcal{C}$; obviously, if equality of the syntax is undecidable then matching will also be undecidable. So assuming that we have not been *silly*, closed rules are useful because we only need to compute one π and consider matching, rather than consider an equivariant matching problem.

To use the matching algorithm of Sect. 2.3.1, we need terms to satisfy condition 2 of Definition 2.98. So, we could forbid *shift* permutations altogether. The algorithm might reintroduce them but as noted in Remark 2.121, *shift* can be eliminated once a solution is found. Thus, if we care about decidability and not so much about infinite permutations—which was the case e.g. in Fernández et al. (2004), Fernández and Gabbay (2007)—then *shift* can be viewed as an internal mechanism of our unification/matching algorithm. However we have designed the mathematics to allow the possibility of exploring other, more liberal (and perhaps still decidable) choices, if we wish. More on this in Gabbay (2012a).

2.3.4 Equality: (Permissive-)Nominal Algebra

Permissive-nominal algebra has one judgement form: an equality $r = s$. This is just an unoriented nominal rewriting rule, so what makes algebra different from rewriting is not so much the judgement form as the properties we care about: instead of confluence and decidability, we primarily care about soundness and completeness. These are Theorems 2.188 and Corollary 2.200.

This different emphasis affects the axioms we write. The rewrites in Example 2.124 are designed to work on λ-terms without unknowns (since we expect to 'evaluate' closed terms using rewrites). The analogous axioms in Example 2.170 are designed to work also on open terms (since we expect to reason about arbitrary denotations).

Permissive-nominal algebra simplifies and streamlines the nominal algebra logic of Gabbay and Mathijssen (2009) (which was based on nominal terms). Essentially, these two logics do the same thing, but there are significant differences which we discuss in Sect. 2.3.4.7. Nominal Algebra (NA) was presented in Gabbay (2005); Gabbay and Mathijssen (2006b); see also Gabbay and Mathijssen (2007, 2009). It was first used to axiomatise substitution, first-order logic, and the λ-calculus Gabbay and Mathijssen (2006a,c, 2008a,c,b, 2010). The interest of these papers was not merely to write down the axioms—which all take advantage of atoms-abstraction to axiomatise various binding operators—but also to prove these axioms sound and complete. These proofs are not included here; see the presentations in Gabbay and Mathijssen (2008a,c, 2010). Or, to see a much more sophisticated instance of the same general idea, the reader can examine the permissive-nominal logic axiomatisation of arithmetic which is proved correct in the case study of Sect. 2.4.2.

2.3.4.1 Judgement Form, Axioms, Theories

Definition 2.168. A (nominal algebra) **equality judgement** is a pair $r = s$.

Definition 2.169. A **theory** $\mathsf{T} = (\Sigma, Ax)$ is a pair of a signature Σ and a possibly infinite set of *equality* judgements Ax in that signature; we call them the **axioms**.

Example 2.170. Here are some example nominal algebra theories:

- naSUB axiomatises capture-avoiding substitution (on the λ-calculus).
  Let Σ have a base sort τ and the following term-formers:

$$\mathsf{sub} : ([\nu]\tau, \tau)\tau \quad \mathsf{lam} : ([\nu]\tau)\tau \quad \mathsf{app} : (\tau, \tau)\tau \quad \mathsf{var} : (\nu)\tau$$

  Axioms are as follows:

$$\begin{array}{lllll}
(\mathbf{var}{\mapsto}) & \mathsf{var}(a)[a{\mapsto}X] & = X & \\
(\#{\mapsto}) & Y[a{\mapsto}X] & = Y & (a \# supp(Y)) \\
(\mathsf{f}{\mapsto}) & \mathsf{f}(Y)[a{\mapsto}X] & = \mathsf{f}(Y[a{\mapsto}X]) & (\mathsf{f} \in \{\mathsf{lam}, \mathsf{app}, \mathsf{var}, \mathsf{sub}\}) \\
(\mathbf{tup}{\mapsto}) & (X_1, \ldots, X_n)[a{\mapsto}X] & = (X_1[a{\mapsto}X], \ldots, X_n[a{\mapsto}X]) & \\
(\mathbf{abs}{\mapsto}) & ([b]Y)[a{\mapsto}X] & = [b](Y[a{\mapsto}X]) & (a \# supp(Y)) \\
(\mathbf{id}{\mapsto}) & Y[b{\mapsto}\mathsf{var}(b)] & = Y & \\
(\eta{\mapsto}) & [a]\mathsf{sub}(Y, \mathsf{var}(a)) & = Y & (a \# supp(Y))
\end{array}$$

  Here and in the next example we sugar $\mathsf{sub}([a]r, t)$ to $r[a{\mapsto}t]$. Every permission set contains b and every permission set contains a except for $supp(Y)$, as indicated above. Sorts are filled in as appropriate.

naSUB is based on the nominal algebra axioms of Gabbay and Mathijssen (2006a, 2008a) (which were parameterised over the signature Σ).

There, we proved the axioms sound and complete for a specific syntactic model in which $Z[a{:=}X]$ really is interpreted as capture-avoiding substitution. The completeness result from Corollary 2.200 remains valid but is weaker because it holds not for the specific syntactic model, but the class of all nominal algebra models of the axioms.

- naLAM extends the previous theory with two more axioms:

$$(\beta) \qquad (\lambda[a]Y)X \qquad = Y[a{\mapsto}X]$$
$$(\eta) \qquad \lambda[a](Xa) \qquad = X \qquad (a{\notin}supp(X))$$

This theory is studied in Gabbay and Mathijssen (2008b, 2010). Analogously to naSUB, we prove the axioms sound and complete for a syntactic model where substitution *is* substitution and β- and η-conversion *are* β- and η-conversion.

Remark 2.171. Compare and contrast Example 2.170 with Example 2.124. Clearly, one is an equality theory and another a rewrite theory, but we obtain a nominal algebra theory from Example 2.124 by replacing $\rightarrow$ by $=$, and conversely we can replace $=$ with $\rightarrow$ in Example 2.170.

So why are they different? They demonstrate different design priorities.

The rewrites in Example 2.124 are designed to operate on ground terms ($fU(r) = \varnothing$), following an intuition that rewriting is about 'executing programs'. The equalities in Example 2.170 are designed to operate on possibly open terms, following an intuition that algebra is about models, not all of whose elements need be referenced by ground terms.

What we gain in deductive power we lose in computational properties. For instance, nrSUB is terminating whereas (an oriented version of) naSUB is not terminating, because explicit substitutions can 'churn' by distributing repeatedly over one another (this is essentially the idea behind Melliès's counterexample in Melliès (1995)). On the other hand while the effect of $(\#{\mapsto})$ from naSUB can be obtained on ground terms using the rules in nrSUB, by pushing the substitution down to the atoms, the rules of nrSUB are not deductively powerful enough to do this for open terms (or arbitrary models). More on this in Gabbay and Mathijssen (2008a, 2010).

2.3.4.2 Derivable Equality

Definition 2.172. Suppose T is a theory. **Derivable equality** $\mathsf{T} \vdash r = s$ is the least transitive reflexive symmetric relation such that for every $(r = s) \in \mathsf{T}$, position P, and substitution θ, if $sort(r) = sort(P)$ and $fa(r\theta) \cup fa(s\theta) \subseteq supp(P)$ (so that $P[l\theta]$ and $P[s\theta]$ are well-defined) then

$$\mathsf{T} \vdash P[r\theta] = P[s\theta].$$

$$\frac{}{r=r}\ (\mathbf{Refl}) \qquad \frac{r=s \quad s=t}{r=t}\ (\mathbf{Trans})$$

$$\frac{r=s}{s=r}\ (\mathbf{Symm}) \qquad \frac{r_1=r'_1 \quad \dots \quad r_n=r'_n}{(r_1,\dots,r_n)=(r'_1,\dots,r'_n)}\ (\mathbf{Cong1})$$

$$\frac{r=r'}{\mathsf{f}(r)=\mathsf{f}(r')}\ (\mathbf{Cong2}) \qquad \frac{r=r'}{[a]r=[a]r'}\ (\mathbf{Cong3})$$

$$\frac{((r{=}s){\in}\mathsf{T})}{\pi{\cdot}(r\theta)=\pi{\cdot}(s\theta)}\ (\mathbf{Ax_{r=s}})$$

Fig. 2.4 Derivable entailment in Permissive-Nominal Algebra (PNA)

Remark 2.173. Definition 2.172 is rather compact; it might be useful to expand it a little. This is Fig. 2.4, given in natural deduction style.

The reader familiar with nominal terms (see for instance Figure 2 of Urban et al. (2004)) should note of (**Cong3**) that we do not need to consider the case $[a]r=[b]s$, because α-equivalence is handled automatically for us by nominal abstract syntax. It is built in by Definition 2.30. In other words, thanks to how we set up our permissive-nominal terms syntax, we can always rename abstracted atoms so that they are equal. We noted an analogous point earlier on, in Remark 2.103.

Lemma 2.174. *Suppose* $\mathsf{T}=(\Sigma,Ax)$ *is a theory. Then:*

- $\mathsf{T}\vdash a=b$ *is impossible.*
- $\mathsf{T}\vdash [a]r=[b]s$ *if and only if* $b\notin fa(r)$ *and* $(b\ a){\cdot}r=s$.
- $\mathsf{T}\vdash (r_1,\dots,r_n)=(s_1,\dots,s_n)$ *if and only if* $\mathsf{T}\vdash r_i=s_i$ *for* $1\leq i\leq n$.

Proof. In axiom $(r=s)\in Ax$, r and s must have base sort τ; thus it is not possible to assert equalities between atoms, abstractions, or tuples (unless wrapped in a term-former and so injected into a base sort). The second part additionally uses Lemma 2.31. ■

Lemma 2.175. *Suppose* $\mathsf{T}\vdash r=s$. *Then:*

1. $\mathsf{T}\vdash \pi{\cdot}r=\pi{\cdot}s$.
2. $\mathsf{T}\vdash r\theta=s\theta$.

Proof. Both parts are by a routine argument on derivations. We consider one case:

- *The case* $(r'=s')\in\mathsf{T}$ *and* $r=P[r'\theta']$ *and* $s=P[s'\theta']$ *and* $P=(t,X)$. For the first part we use a position $(\pi{\cdot}t,X)$.

  For the second part we consider a position $P'=(t(\theta{-}X),X)$ and consider $P'[r'\theta'\theta]$ and $S'[r'\theta'\theta]$ ($\theta{-}X$ defined in Definition 2.109). It is not hard to check that $P'[r'\theta'\theta]=P[r'\theta']\theta$ and $P'[s'\theta'\theta]=P[s'\theta']\theta$, and the result follows.

■

2.3.4.3 Interpretation of Signatures and Terms

Definition 2.176. Suppose $(\mathcal{A},\mathcal{B})$ is a sort-signature (Definition 2.39).

An **interpretation** $\mathscr{I}$ for $(\mathcal{A},\mathcal{B})$ consists of an assignment of a nonempty permissive-nominal set $[\![\alpha]\!]^{\mathscr{I}}$ to each sort α in $(\mathcal{A},\mathcal{B})$, along with equivariant maps

- for each $\nu \in \mathcal{A}$ an equivariant and injective map $\mathbb{A}_\nu \to [\![\nu]\!]^{\mathscr{I}}$ which we write $a^{\mathscr{I}}$,
- for each $\nu \in \mathcal{A}$ and α an equivariant and injective map $[\mathbb{A}_\nu][\![\alpha]\!]^{\mathscr{I}} \to [\![[\nu]\alpha]\!]^{\mathscr{I}}$ which we write $[a]^{\mathscr{I}}x$, and
- for each α_i for $1 \leq i \leq n$ an equivariant and injective map $\Pi_i[\![\alpha_i]\!]^{\mathscr{I}} \to [\![(\alpha_1,\ldots,\alpha_n)]\!]^{\mathscr{I}}$ which we write $(x_1,\ldots,x_n)^{\mathscr{I}}$.

Definition 2.177. Suppose $\Sigma = (\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{F},ar)$ is a signature (Definition 2.43).

An **interpretation** $\mathscr{I}$ for Σ, or **Σ-algebra**, consists of the following data:

- An interpretation for the sort-signature $(\mathcal{A},\mathcal{B})$ (Definition 2.176).
- For every $\mathsf{f} \in \mathcal{F}$ with $ar(\mathsf{f}) = (\alpha)\tau$ an equivariant function $\mathsf{f}^{\mathscr{I}}$ from $[\![\alpha]\!]^{\mathscr{I}}$ to $[\![\tau]\!]^{\mathscr{I}}$.
- An equivariant assignment from $C \in \mathcal{C}$ to $C^{\mathscr{I}} \in [\![sort(C)]\!]^{\mathscr{I}}$. (That is, $(\pi\cdot C)^{\mathscr{I}} = \pi\cdot(C^{\mathscr{I}})$.)

Definition 2.178. Suppose $\mathscr{I}$ is a Σ-algebra. A **valuation** ς to $\mathscr{I}$ is an equivariant function on unknowns $\mathcal{X}$ such that for each unknown X, $\varsigma(X) \in [\![sort(X)]\!]^{\mathscr{I}}$.

ς will range over valuations.

Definition 2.179. Suppose $\mathscr{I}$ is a Σ-algebra. Suppose ς is a valuation to $\mathscr{I}$.

Extend $\mathscr{I}$ to an **interpretation** on terms $[\![r]\!]^{\mathscr{I}}_{\varsigma}$ (where of course r is a term in the signature Σ) by:

$$\begin{array}{cc} [\![a]\!]^{\mathscr{I}}_{\varsigma} = a^{\mathscr{I}} & [\![\mathsf{f}(r)]\!]^{\mathscr{I}}_{\varsigma} = \mathsf{f}^{\mathscr{I}}([\![r]\!]^{\mathscr{I}}_{\varsigma}) \\ [\![C]\!]^{\mathscr{I}}_{\varsigma} = C^{\mathscr{I}} & [\![(r_1,\ldots,r_n)]\!]^{\mathscr{I}}_{\varsigma} = ([\![r_1]\!]^{\mathscr{I}}_{\varsigma},\ldots,[\![r_n]\!]^{\mathscr{I}}_{\varsigma})^{\mathscr{I}} \\ [\![X]\!]^{\mathscr{I}}_{\varsigma} = \varsigma(X) & [\![[a]r]\!]^{\mathscr{I}}_{\varsigma} = [a]^{\mathscr{I}}[\![r]\!]^{\mathscr{I}}_{\varsigma} \end{array}$$

Lemma 2.180 is a basic sanity check and an important soundness result:

Lemma 2.180. *If $r : \alpha$ then $[\![r]\!]^{\mathscr{I}}_{\varsigma} \in [\![\alpha]\!]^{\mathscr{I}}$.*

Proof. By a routine induction on r. ■

Lemma 2.181. $\pi\cdot[\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![\pi\cdot r]\!]^{\mathscr{I}}_{\varsigma}$.

Proof. By a routine induction on r. We consider one case:

- *The case X.* By Definition 2.179 $[\![X]\!]^{\mathscr{I}}_{\varsigma} = \varsigma(X)$. Therefore $\pi\cdot[\![X]\!]^{\mathscr{I}}_{\varsigma} = \pi\cdot\varsigma(X)$. By assumption $\pi\cdot\varsigma(X) = \varsigma(\pi\cdot X) = [\![\pi\cdot X]\!]^{\mathscr{I}}_{\varsigma}$.

■

Lemma 2.182. $supp([\![r]\!]^{\mathscr{I}}_{\varsigma}) \subseteq fa(r)$.

Proof. From Lemmas 2.21 and 2.181. ■

2.3.4.4 Models and Soundness

Definition 2.183. For a theory $\mathsf{T} = (\Sigma, Ax)$ and interpretation $\mathscr{I}$ of T call $(r = s)$ **valid** in $\mathscr{I}$ when $[\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![s]\!]^{\mathscr{I}}_{\varsigma}$ for every valuation ς to $\mathscr{I}$.
Call $\mathscr{I}$ a **model** of T when every axiom $(r = s) \in Ax$ is valid in $\mathscr{I}$.
Write $\mathsf{T} \models r = s$ when $(r = s)$ is valid in every model of T.

Lemma 2.184. *If $\varsigma(X) = \varsigma'(X)$ for all $X \in fU(r)$ then $[\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![r]\!]^{\mathscr{I}}_{\varsigma'}$.*

Proof. By a routine induction on r. ■

Definition 2.185. Suppose ς is a valuation to $\mathscr{I}$. Suppose X is an unknown and $x \in [\![sort(X)]\!]^{\mathscr{I}}$ is such that $supp(x) \subseteq supp(X)$. Define a function $\varsigma[X{:=}x]$ by

$$(\varsigma[X{:=}x])(\pi{\cdot}X) = \pi{\cdot}x \quad \text{and} \quad (\varsigma[X{:=}x])(Y) = \varsigma(Y) \quad \text{all other } Y$$

Lemma 2.186. *$\varsigma[X{:=}x]$ in Definition 2.185 is well-defined and a valuation to $\mathscr{I}$.*

Proof. As that of Proposition 2.38. ■

Lemma 2.187. *$[\![r]\!]^{\mathscr{I}}_{\varsigma[X:=[\![t]\!]^{\mathscr{I}}_{\varsigma}]} = [\![r[X{:=}t]]\!]^{\mathscr{I}}_{\varsigma}$. As corollaries we have:*

1. *If $[\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![s]\!]^{\mathscr{I}}_{\varsigma}$ then $[\![P[r]]\!]^{\mathscr{I}}_{\varsigma} = [\![P[s]]\!]^{\mathscr{I}}_{\varsigma}$.*
2. *If $[\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![s]\!]^{\mathscr{I}}_{\varsigma}$ then $[\![r\theta]\!]^{\mathscr{I}}_{\varsigma} = [\![s\theta]\!]^{\mathscr{I}}_{\varsigma}$.*

Proof. By a routine induction on the definition of $[\![r]\!]^{\mathscr{I}}_{\varsigma}$. We consider one case:

- *The case of $[\![\pi{\cdot}X]\!]^{\mathscr{I}}_{\varsigma[X:=t]}$.* We reason as follows:

$$\begin{array}{lll} [\![\pi{\cdot}X]\!]^{\mathscr{I}}_{\varsigma[X:=[\![t]\!]^{\mathscr{I}}_{\varsigma}]} = \pi{\cdot}[\![t]\!]^{\mathscr{I}}_{\varsigma} & & \text{Definition 2.179} \\ = [\![\pi{\cdot}t]\!]^{\mathscr{I}}_{\varsigma} & & \text{Lemma 2.181} \\ = [\![(\pi{\cdot}X)[X{:=}t]]\!]^{\mathscr{I}}_{\varsigma} & & \text{Definition 2.69.} \end{array}$$

For the two corollaries we reason as follows:

1. By definition where $P = (t, X)$, $P[r] = t[X{:=}r]$ and $P[s] = t[X{:=}s]$. Using the assumptions,

$$[\![t[X{:=}r]]\!]^{\mathscr{I}}_{\varsigma} = [\![t]\!]^{\mathscr{I}}_{\varsigma[X:=[\![r]\!]^{\mathscr{I}}_{\varsigma}]} = [\![t]\!]^{\mathscr{I}}_{\varsigma[X:=[\![s]\!]^{\mathscr{I}}_{\varsigma}]} = [\![t[X{:=}s]]\!]^{\mathscr{I}}_{\varsigma}.$$

2. It is a fact of syntax that $fU(r)$ and $fU(s)$ are finite. Using Lemma 2.73 we may represent the effect of θ on r and s as a sequence of atomic substitutions (Definition 2.66). The result follows.

■

Theorem 2.188 (Soundness). *For any* $\mathsf{T} = (\Sigma, Ax)$ *if* $\mathsf{T} \vdash r = s$ *then* $\mathsf{T} \models r = s$.

Proof. Let $\mathscr{I}$ be a model of T and ς be a valuation to $\mathscr{I}$.

Identity in the denotation is reflexive, transitive, and symmetric so it suffices to check the theorem for axioms. That is, suppose $(r = s) \in Ax$ and assume a position P and substitution θ such that $sort(r) = sort(P)$ and $fa(r\theta) \cup fa(s\theta) \subseteq supp(P)$. We must show that $[\![P[r\theta]]\!]^{\mathscr{I}}_{\varsigma} = [\![P[s\theta]]\!]^{\mathscr{I}}_{\varsigma}$.

$\mathscr{I}$ is a model so $[\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![s]\!]^{\mathscr{I}}_{\varsigma}$. We use parts 1 and 2 of Lemma 2.187. ■

2.3.4.5 Free Term Models and Completeness

In this section fix a signature Σ and a theory $\mathsf{T} = (\Sigma, Ax)$.

The proof of completeness follows a standard method: we construct a model out of syntax in which by construction two terms denote equal elements if and only if they are derivably equal.

The subtlety occurs in Lemma 2.197. We want to eliminate ς in $[\![r]\!]^{\mathscr{F}(\mathsf{T})}_{\varsigma}$ by converting it into a substitution θ. This 'should' be easy, since for each X, $\varsigma(X)$ is a provably equivalent class of terms. We need only choose some representative term in $\varsigma(X)$ for each X and set $\theta(X)$ to be that representative.

If we are naive in our construction then this could be impossible, as outlined in Example 2.198: there might be 'too many atoms' in the available representatives. We enrich our syntax with 'enough' extra constant symbols, to guarantee 'enough' representatives of every element of the model. Nominal algebra without the constant symbols is complete for the same semantics, but the proof would be more complex.

Definition 2.189. For each sort α in Σ define $[r]_{\mathsf{T}}$ and $\mathscr{F}(\mathsf{T})_\alpha$ by

$$[r]_{\mathsf{T}} = \{r' : \alpha \mid \mathsf{T} \vdash r = r'\} \qquad (r : \alpha)$$
$$\mathscr{F}(\mathsf{T})_\alpha = \{[r]_{\mathsf{T}} \mid r : \alpha\}.$$

Make each $\mathscr{F}(\mathsf{T})_\alpha$ into a permissive-nominal set by giving it a permutation action

$$\pi \cdot [r]_{\mathsf{T}} = [\pi \cdot r]_{\mathsf{T}}.$$

$\mathscr{F}(\mathsf{T})$ stands for '$\mathscr{F}$ree terms in the signature of T, up to derivable equality in T'. Lemmas 2.190 and 2.191 relate permutation and support to the natural notions from nominal sets:

Lemma 2.190. *The permutation action on* $[r]_{\mathsf{T}}$ *is pointwise on* $[r]_{\mathsf{T}}$ *as a set: that is,* $\pi \cdot [r]_{\mathsf{T}} = \{\pi \cdot r' \mid r' \in [r]_{\mathsf{T}}\}$.

Proof. From Definition 2.189 and Lemma 2.175. ■

Lemma 2.191. $supp([r]_{\mathsf{T}}) \subseteq fa(r)$.

Proof. From Definition 2.189 and Lemma 2.21. ■

Definition 2.192. We construct the **free term interpretation** $\mathscr{F}(\mathsf{T})$ of T as follows:

- Take $\mathscr{F}(\mathsf{T})_\alpha$ as in Definition 2.189.
- $a^{\mathscr{F}(\mathsf{T})} = [a]_\mathsf{T}$, $[a]^{\mathscr{F}(\mathsf{T})}[r]_\mathsf{T} = [[a]r]_\mathsf{T}$, and $([r_1]_\mathsf{T},\ldots,[r_n]_\mathsf{T})^{\mathscr{F}(\mathsf{T})} = [(r_1,\ldots,r_n)]_\mathsf{T}$.
- $\mathsf{f}^{\mathscr{F}(\mathsf{T})}([r]_\mathsf{T}) = [\mathsf{f}(r)]_\mathsf{T}$ for each term-former $\mathsf{f} : (\alpha)\tau$ in Σ and each $r : \alpha$.
- $C^{\mathscr{F}(\mathsf{T})} = [C]_\mathsf{T}$ for each constant in Σ.

Lemma 2.193. *Definition 2.192 is well-defined and is an interpretation. That is:*

- *The choice of representative of $[r]_\mathsf{T}$ does not matter in any of the clauses.*
- *The choice of abstracted atom in the clause for $[a]^{\mathscr{F}(\mathsf{T})}[r]_\mathsf{T}$ does not matter.*
- *The maps $a^{\mathscr{F}(\mathsf{T})}$, $[a]^{\mathscr{F}(\mathsf{T})}[r]_\mathsf{T}$, and $([r_1]_\mathsf{T},\ldots,[r_n]_\mathsf{T})^{\mathscr{F}(\mathsf{T})}$ are injective.*

Proof. The first part follows by congruence properties of derivable equality. The second part additionally uses Lemmas 2.32 and 2.33. The third part uses Lemma 2.174. ■

Definition 2.194. Define a theory $\mathsf{T}^+ = (\Sigma^+, Ax^+)$ to be equal to T except that we adjoin $\bigcup_\tau \mathscr{F}(\mathsf{T})_\tau$ to the set of constants in Σ, and we add axioms equating r with $[r]_\mathsf{T}$ in Ax.

That is, for every $r : \tau$ there is a constant $C_r = [r]_\mathsf{T} \in \Sigma^+$, and an axiom $(C_r = r) \in \mathscr{F}(\mathsf{T})^+$.

Lemma 2.195. *$\mathscr{F}(\mathsf{T})$ extends to an interpretation $\mathscr{F}(\mathsf{T})^+$ of T^+, where for each $r : \tau$ we take $C_r^{\mathscr{F}(\mathsf{T})^+} = [r]_\mathsf{T}$. Furthermore, $\mathscr{F}(\mathsf{T})^+$ is a model of T^+.*

Definition 2.196. Write ς_{id} for the valuation to $\mathscr{F}(\mathsf{T})$ mapping each X to $C_X = [X]_\mathsf{T}$.

Lemma 2.197. *For every valuation ς to $\mathscr{F}(\mathsf{T})$ there exists a substitution θ in T^+ such that $[\![r]\!]_\varsigma^{\mathscr{F}(\mathsf{T})} = [\![r\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+}$.*

Proof. For each orbit $x \in |orb(\mathcal{X})|$ choose a representative $X \in x$. Define θ by $\theta(\pi{\cdot}X) = \pi{\cdot}C_X$. Recall that $C_X = [X]_\mathsf{T}$ and by Lemma 2.191 $supp([X]_\mathsf{T}) \subseteq supp(X)$. By Proposition 2.38 θ is well-defined and is a substitution

It is not hard to check by induction on r that $[\![r]\!]_\varsigma^{\mathscr{F}(\mathsf{T})} = [\![r\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+}$. ■

Example 2.198. To see why Lemma 2.197 is non-trivial and how T^+ helps, suppose T has one name sort ν, two base sorts τ and τ', one term-former $\mathsf{abs} : (\nu,\tau)\tau'$, and one axiom $\mathsf{abs}(b,(b\ a){\cdot}X) = \mathsf{abs}(a,X)$ where $a \in supp(X)$ and $b \notin supp(X)$.

Then it is a fact that there is no $r \in [\mathsf{abs}(a,X)]_\mathsf{T}$ such that $fa(r) \subseteq supp([\mathsf{abs}(a,X)]_\mathsf{T})$ and it follows that there is no θ such that $[\![X']\!]_{[X':=[\mathsf{abs}(a,X)]_\mathsf{T}]}^{\mathscr{F}(\mathsf{T})} = [\![X'\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+}$ (recall that substitutions must be equivariant).

Theorem 2.199. *$\mathscr{F}(\mathsf{T})$ is a model of T.*

Proof. We must show that $\mathscr{F}(\mathsf{T})$ validates the axioms.

Suppose $(r = s) \in Ax$. Suppose ς is a valuation to $\mathscr{F}(\mathsf{T})$. We must show that $[\![r]\!]_\varsigma^{\mathscr{F}(\mathsf{T})} = [\![s]\!]_\varsigma^{\mathscr{F}(\mathsf{T})}$.

By Lemma 2.197 there exists θ to T^+ such that $[\![r]\!]_\varsigma^{\mathscr{F}(\mathsf{T})} = [\![r\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+}$ and $[\![s]\!]_\varsigma^{\mathscr{F}(\mathsf{T})} = [\![s\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+}$.

By assumption $\mathsf{T}^+ \vdash r\theta = s\theta$. By Lemma 2.195, $[\![r\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+} = [\![s\theta]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})^+}$. The result follows. ■

Corollary 2.200 (Completeness). *If* $\mathsf{T} \models r = s$ *then* $\mathsf{T} \vdash r = s$.

Proof. Suppose $\mathsf{T} \models r = s$. By Theorem 2.199 $[\![r]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})} = [\![s]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})}$ (ς_{id} is defined in Definition 2.196).

It is not hard to prove by induction that $[\![r]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})} = [r]_\mathsf{T}$ and $[\![s]\!]_{\varsigma_{id}}^{\mathscr{F}(\mathsf{T})} = [s]_\mathsf{T}$. It follows that $\mathsf{T} \vdash r = s$ as required. ■

2.3.4.6 Freshness

Nominal terms freshness conditions $a\#X$ and $a\#r$ from Urban et al. (2004) correspond in this paper to 'free atoms of' $a \notin supp(X)$ and $a \notin fa(r)$. See Notation 2.55 and Lemma 2.53. Call this **syntactic** freshness.

Nominal sets freshness $a \notin supp([\![r]\!])$ is a distinct notion which can be expressed using equality; call this **semantic** freshness. The two are not identical, but they are connected in various ways which we briefly explore.

Proposition 2.201 corresponds to Theorem 5.5 from Gabbay and Mathijssen (2007) and Lemma 4.51 from Gabbay and Mathijssen (2009):

Proposition 2.201. *Suppose* $b \notin fa(r)$.

Then $\mathsf{T} \vdash (b\ a)\cdot r = r$ *if and only if for every model* $\mathscr{I}$ *of* T *and valuation* ς *to* $\mathscr{I}$, $a \notin supp([\![r]\!]_\varsigma^\mathscr{I})$.

Proof. By Theorem 2.188 and Corollary 2.200 $\mathsf{T} \vdash (b\ a)\cdot r = r$ if and only if $\mathsf{T} \models (b\ a)\cdot r = r$, which unpacking definitions means that for every $\mathscr{I}$ and ς, $[\![(b\ a)\cdot r]\!]_\varsigma^\mathscr{I} = [\![r]\!]_\varsigma^\mathscr{I}$. By Lemma 2.181 $[\![(b\ a)\cdot r]\!]_\varsigma^\mathscr{I} = (b\ a)\cdot[\![r]\!]_\varsigma^\mathscr{I}$, and by Lemma 2.182 $b \notin supp([\![r]\!]_\varsigma^\mathscr{I})$. The result follows by Corollary 2.18. ■

Lemmas 2.191 (and also Lemma 2.182) express that syntactic freshness implies semantic freshness. A partial converse is Proposition 2.203, which is based on a technical property of nominal sets:

Lemma 2.202. *Suppose* X *is a nominal set and* $U \subseteq |\mathsf{X}|$ *is finitely-supported (so* $U \in |pow(\mathsf{X})|$ *from Example 2.16) and nonempty.*

Then if $a\#U$ *then there exists some* $x \in U$ *with* $a\#x$.

Proof. U is nonempty so choose any $x' \in U$. Choose fresh b (so $b \notin supp(U) \cup supp(x')$) and set $x = (b\ a)\cdot x'$. By the definition of support $(b\ a)\cdot U = U$. By the pointwise action (Example 2.16) $x \in U$. By Lemma 2.17 $a \notin supp(x)$. ■

Proposition 2.203. *$a\#[r]_{\mathsf{T}}$ implies there exists some r' such that $\mathsf{T} \vdash r = r'$ and $a \notin fa(r')$.*

Proof. By Lemmas 2.190 and Lemma 2.202. ■

2.3.4.7 Design of Nominal Algebra

We designed nominal algebra originally to axiomatise substitution, first-order logic, and the λ-calculus Gabbay and Mathijssen (2006a,c, 2008a,c, 2009).

We encountered two design decisions: whether to include freshness axioms, and whether to include atoms-abstraction as primitive.

We disallowed freshness axioms because they are a definitional extension of the system without them, and we chose to include atoms-abstraction as primitive because—even though they too are a definitional extension (see next paragraph)—they make for more compact derivations and proofs and we knew that the reader would expect to see them in a 'nominal' paper. These decisions do not matter for expressivity because of the following two equalities from Gabbay and Pitts (2001), here written in the language of FM sets:

$$(\textbf{Freshness}) \qquad a\#x \Leftrightarrow И b.(b\ a)\cdot x = x$$

$$(\textbf{Abstraction}) \qquad И b.([b](b\ a)\cdot x = [a]x)$$

In Sect. 2.4.3.1 we express the equalities above in PNL. In Gabbay (2012b), we do the same in nominal algebra, showing how to compile nominal algebra with semantic freshness judgements and atoms-abstraction down to the core logic without it.

$И$ above is the *new*-quantifier meaning 'for some/any fresh atom' Gabbay and Pitts (2001), Gabbay (2011b).[14] $И$ does not care which fresh atom we choose (the some/any property (Gabbay 2011b, Theorem 6.5)). So, we do not have to be exact about $supp(x)$ when we choose fresh b; *any* will do, and for instance Proposition 2.201 is an 'if and only if' even though we chose $b \notin fa(r)$ (syntactic freshness) instead of $b \notin supp([\![r]\!])$ (semantic freshness), and it may be that $supp([\![r]\!]) \subsetneqq fa(r)$. More on this Sect. 2.4.3.1.

Note that including atoms-abstraction is orthogonal to the rest of the logic in the sense that it is isolated by the sort system: if we provide no term-formers injecting atoms-abstraction into base sorts, then it cannot interact with the rest of the logic.

The permissive-nominal algebra of this paper differs from the nominal algebra of Gabbay and Mathijssen (2009) in the following respects:

- The system here is sorted, the system in Gabbay and Mathijssen (2009) is not.

[14] In words: 'a is fresh for x if for some/any fresh b, $(b\ a)\cdot x = x$' and 'for some/any fresh b, $[b](b\ a)\cdot x = [a]x$'.

- We use permissive-nominal terms and semantics here, and 'vanilla' nominal terms and nominal sets in Gabbay and Mathijssen (2009). That is, the logic here is *permissive*-nominal algebra. Freshness conditions $a\#X$ and $a\#r$ translate to $a \notin supp(X)$ and $a \notin fa(r)$ here.
- Axioms are exactly equalities, with no freshness contexts: permission sets play this role instead.
- The syntax here admits non-equivariant constant symbols, that of Gabbay and Mathijssen (2009) does not. That does not matter if we are using finitely-supported models (as is the case in Gabbay and Mathijssen (2009)) because finite non-equivariance can be emulated using term-formers applied to finitely many atoms. Here, elements can have infinite support, which cannot be emulated using (finite) equivariant term-formers.
- The syntax here admits the possibility of unknowns with empty support ranging over closed elements (so it includes the $\bullet t$ freshness constraint of (Fernández and Gabbay 2007, Section 9.2)), unknowns with finite support ranging over finitely-supported elements, unknowns with support equal to a permission set, and whatever else we can imagine in-between.
- The development is parameterised over the set of unknowns $\mathcal{X}$ and *also* the group of permutations $\mathbb{P}$. In particular we admit (but do not insist on) the possibility of infinite permutations, including the *shift*-permutations considered in Sect. 2.2.2.6.
- Substitutions and valuations are—rather elegantly—treated as equivariant functions on $\mathcal{X}$ the set of unknowns.

In spite of these many differences, the spirit of the proofs remains the same. The details become simpler, and in particular the non-equivariant constants make construction of the free term model easier.[15]

2.3.5 The Nominal HSP Theorem

The HSP theorem states that a class of Σ-algebras is equational if and only if it is closed under Homomorphism, Subobject, and Product. Definitions follow below, and the main result is Theorem 2.228.

The result was first proved for the case of 'ordinary' algebra (using first-order terms and not over nominal sets) by Birkhoff (1935). It is also called *Birkhoff's theorem* (Burris and Sankappanavar 1981, Theorem 11.12). We prefer 'HSP' since this is more descriptive and Birkhoff's name is attached to several other results.

The result was first proved for nominal algebra by the author Gabbay (2009), and an alternative proof was provided by Kurz and Petrişan (2010). The new proof presented here is also rather short.

[15] In Gabbay and Mathijssen (2009) to build the free term model we enriched syntax with n-ary term-formers applied to atoms. This idea goes back to a completeness proof in Gabbay (2007a).

HSP was interesting for two reasons: first, it is not obvious that nominal algebra is a true logic of equality, because of the freshness side-conditions which give the nominal algebra as presented e.g. in Gabbay and Mathijssen (2009) or in Mathijssen's thesis (2007) a *prima facie* flavour of conditional equalities. The HSP result holding for nominal algebra was a way of making formal that this *is* a logic of equality.

The use of permission sets to phrase the logic entirely in terms of equality (freshness migrates to the types, as permission sets) is a step forward from this point of view: the nominal algebra of this paper is more *visibly* an equational logic. Still, HSP along with soundness and completeness (Theorem 2.188 and Corollary 2.200) form a triumvirate of results of interest for an algebraic reasoning framework.

The proofs here are much shorter and clearer than those of Gabbay (2009)—and the final result is strictly stronger than Gabbay (2009), Kurz and Petrişan (2010), which actually proved an HSPA theorem that a class of Σ-algebras is equational if and only if it is closed under Homomorphism, Subobject, Product, and Atoms-abstraction.

That is, we have dropped the 'atoms-abstraction' from the closure conditions. How can this be? The use of permission sets gives us finer control over the support of valuations; we needed atoms-abstraction in the proof of (Gabbay 2009, Theorem 9.8) to eliminate 'extra' atoms introduced by a valuation ς—'extra' relative to the freshness information in a freshness context Δ. Here, because freshness contexts/permission sets are fixed, this cannot happen.

2.3.5.1 Algebra Homomorphisms

Definition 2.204. Suppose $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar)$ is a signature and suppose $\mathscr{X}$ and $\mathscr{Y}$ are interpretations of Σ. A **Σ-homomorphism** Θ from $\mathscr{X}$ to $\mathscr{Y}$ is a family of equivariant functions Θ_α from $[\![\alpha]\!]^{\mathscr{X}}$ to $[\![\alpha]\!]^{\mathscr{Y}}$ for each sort α in the sort-signature $(\mathcal{A}, \mathcal{B})$ such that:

- $\Theta_\nu(a^{\mathscr{X}}) = a^{\mathscr{Y}}$.
- $\Theta_{(\alpha_1,\ldots,\alpha_n)}(x_1,\ldots,x_n)^{\mathscr{X}} = (\Theta_{\alpha_1}(x_1),\ldots,\Theta_{\alpha_n}(x_n))^{\mathscr{Y}}$.
- $\Theta_{[\nu]\alpha}([a]^{\mathscr{X}} x) = [a]^{\mathscr{Y}} \Theta_\alpha(x)$.
- $\Theta_\tau(\mathsf{f}^{\mathscr{X}}(x)) = \mathsf{f}^{\mathscr{Y}}(\Theta_\alpha(x))$ where $\mathsf{f} : (\alpha)\tau$ is in $\mathcal{F}$.

Definition 2.205. Call $\mathscr{Y}$ a **homomorphic image** of $\mathscr{X}$ when there is a Σ-homomorphism Θ from $\mathscr{X}$ to $\mathscr{Y}$ such that Θ_α is surjective for every sort α in $(\mathcal{A}, \mathcal{B})$.

Call Θ **injective** when Θ_α is injective for every sort α in $(\mathcal{A}, \mathcal{B})$.

Lemma 2.206. *Suppose Σ is a signature and $\mathscr{X}$ and $\mathscr{Y}$ are Σ-algebras. Suppose Θ is a Σ-algebra homomorphism from $\mathscr{X}$ to $\mathscr{Y}$.*

Suppose that ς is a valuation to $\mathscr{X}$. Define $\Theta(\varsigma)$ a valuation to $\mathscr{Y}$ by $\Theta(\varsigma)(X) = \Theta_{sort(X)}(\varsigma(X))$ for every $X \in \mathcal{X}$.

Then for every $r : \alpha$, $\Theta_\alpha([\![r]\!]^{\mathscr{X}}_\varsigma) = [\![r]\!]^{\mathscr{Y}}_{\Theta(\varsigma)}$.

Proof. By an easy induction on r. ■

Lemma 2.207. *Suppose* Σ *is a signature and* $\mathsf{T} = (\Sigma, Ax)$ *is a theory. Suppose* $\mathscr{X}$ *and* $\mathscr{Y}$ *are* Σ*-algebras and* $\mathscr{Y}$ *is a homomorphic image of* $\mathscr{X}$ *under* Θ.

Then if $\mathscr{X}$ *is a model of* T, *then so is* $\mathscr{Y}$.

Proof. Choose $(r = s) \in Ax$ and a valuation ς to $\mathscr{Y}$. It suffices to show that $[\![r]\!]^{\mathscr{Y}}_\varsigma = [\![s]\!]^{\mathscr{Y}}_\varsigma$.

We construct a valuation ς' to $\mathscr{X}$ as an equivariant extension (Definition 2.37) of the following data. For each unknown $X : \alpha$ let $\mathcal{X}_X = \{x \in |\mathscr{X}_\alpha| \mid \Theta(x) = \varsigma(X)\}$. We construct a valuation ς' to $\mathscr{X}$ by for each orbit and representative $X \in orb(X) \in \mathcal{X}$ setting $\varsigma'(X) = x$ for some choice of $x \in \mathcal{X}_X$.

By construction $\Theta\varsigma' = \varsigma$. By assumption $[\![r]\!]^{\mathscr{X}}_{\varsigma'} = [\![s]\!]^{\mathscr{X}}_{\varsigma'}$. We apply Θ to both sides and use Lemma 2.206. ■

2.3.5.2 Subalgebras

Definition 2.208. For Σ-algebras $\mathscr{X}$ and $\mathscr{Y}$, call $\mathscr{X}$ a **subalgebra** of $\mathscr{Y}$ when:

- $|\tau^{\mathscr{X}}| \subseteq |\tau^{\mathscr{Y}}|$ for every $\tau \in \mathcal{B}$.
- The subset inclusion maps form a Σ-algebra homomorphism (Definition 2.204).[16]

Lemma 2.209. *For* Σ*-algebras* $\mathscr{X}$, $\mathscr{Y}$ *and a theory* $\mathsf{T} = (\Sigma, Ax)$, *if* $\mathscr{Y}$ *is a model of* T *and* $\mathscr{X}$ *is a subalgebra of* $\mathscr{Y}$ *then* $\mathscr{X}$ *is a model of* T.

2.3.5.3 Products

Definition 2.210. Let I be a (possibly countably infinite) indexing set and $(\mathscr{X}_i)_{i\in I}$ be an I-indexed collection of Σ-algebras. The **product algebra** $\Pi_{i\in I}\mathscr{X}_i$ is the Σ-algebra such that:

- For each α in Σ, $\alpha^{\Pi_{i\in I}\mathscr{X}_i} = \Pi_{i\in I}\alpha^{\mathscr{X}_i}$ as defined in Definition 2.27.
- The ith projection map to $\mathscr{X}_i$ is a Σ-algebra homomorphism for every $i \in I$.

Lemma 2.211. *For any* I*-indexed collection of* Σ*-algebras* $(\mathscr{X}_i)_{i\in I}$, *if* $\mathscr{X}_i$ *is a model of* $\mathsf{T} = (\Sigma, Ax)$ *for every* $i \in I$ *then so is* $\Pi_{i\in I}\mathscr{X}_i$.

---

[16] That is:

- $a^{\mathscr{X}} = a^{\mathscr{Y}}$ for every atom a.
- $(x_1, \ldots, x_n)^{\mathscr{X}} = (x_1, \ldots, x_n)^{\mathscr{Y}}$ for every $x_1 \in |[\![\alpha_1]\!]^{\mathscr{X}}|, \ldots, x_n \in |[\![\alpha_n]\!]^{\mathscr{Y}}|$.
- $[a]^{\mathscr{X}} x = [a]^{\mathscr{Y}} x$ for every $x \in |[\![\alpha]\!]^{\mathscr{X}}|$.
- For every term-former f in $\mathcal{F}$, $\mathsf{f}^{\mathscr{X}}(x) = \mathsf{f}^{\mathscr{Y}}(x)$ for every $x \in |[\![\alpha]\!]^{\mathscr{X}}|$ where $ar(\mathsf{f}) = (\alpha)\tau$.

Proof. Suppose $(r = s) \in Ax$. Suppose ς is a valuation to $\Pi_{i\in I}\mathscr{X}_i$. For each $i \in I$ we obtain a valuation ς_i to $\mathscr{X}_i$ by projecting to the ith component. It follows that $[\![r]\!]^{\mathscr{X}_i}_{\varsigma_i} = [\![s]\!]^{\mathscr{X}_i}_{\varsigma_i}$, and thus $[\![r]\!]^{\Pi_{i\in I}\mathscr{X}_i}_{\varsigma} = [\![s]\!]^{\Pi_{i\in I}\mathscr{X}_i}_{\varsigma}$. ■

2.3.5.4 Ground Term Models and Extending a Signature

Definition 2.212. Call r **ground** when $fU(r) = \varnothing$.

Definition 2.213 exactly follows Definition 2.189 (cf. Remark 2.217):

Definition 2.213. Suppose $\mathsf{T} = (\Sigma, Ax)$ is a theory. For each sort α in Σ define $[r]^{gnd}_{\mathsf{T}}$ and $\mathscr{G}(\mathsf{T})_\alpha$ by

$$\begin{array}{rl} [r]^{gnd}_{\mathsf{T}} = \{r' : \alpha \mid \mathsf{T} \vdash r = r'\} & (r : \alpha, r \text{ ground}) \\ \mathscr{G}(\mathsf{T})_\alpha = \{[r]^{gnd}_{\mathsf{T}} \mid r : \alpha,\ r \text{ ground}\}. & \end{array}$$

Make each $\mathscr{G}(\mathsf{T})_\alpha$ into a permissive-nominal set by giving it a permutation action

$$\pi\cdot[r]^{gnd}_{\mathsf{T}} = [\pi\cdot r]^{gnd}_{\mathsf{T}}.$$

Lemma 2.214. $supp([r]^{gnd}_{\mathsf{T}}) \subseteq fa(r)$.

Proof. From Definition 2.189 and Lemma 2.21. ■

Definition 2.215. We construct the **ground free term interpretation** $\mathscr{G}(\mathsf{T})$ of T as follows:

We take $\mathscr{G}(\mathsf{T})_\alpha$ as in Definition 2.213. We define:

$$\begin{array}{rl} a^{\mathscr{G}(\mathsf{T})} & = [a]^{gnd}_{\mathsf{T}} \\ [a]^{\mathscr{G}(\mathsf{T})}[r]^{gnd}_{\mathsf{T}} & = [[a]r]^{gnd}_{\mathsf{T}} \\ ([r_1]^{gnd}_{\mathsf{T}}, \ldots, [r_n]^{gnd}_{\mathsf{T}})^{\mathscr{G}(\mathsf{T})} & = [(r_1, \ldots, r_n)]^{gnd}_{\mathsf{T}} \\ \mathsf{f}^{\mathscr{G}(\mathsf{T})}([r]^{gnd}_{\mathsf{T}}) & = [\mathsf{f}(r)]^{gnd}_{\mathsf{T}} \\ C^{\mathscr{G}(\mathsf{T})} & = [C]^{gnd}_{\mathsf{T}} \end{array}$$

Above, f ranges over each term-former $\mathsf{f} : (\alpha)\tau$ in Σ and C ranges over each constant in Σ.

Lemma 2.216. *Definition 2.215 is well-defined and is an interpretation.*

Proof. As the proof of Lemma 2.193. ■

Remark 2.217. Definition 2.189 is a special case of Definition 2.189. We obtain $\mathscr{F}(\mathsf{T})$ as $\mathscr{G}(\mathsf{T}')$ where T' is obtained from T by extending its signature with a copy of $\mathcal{X}$ as constants (the construction is made formal in Definition 2.219 below).

Doing this in Definition 2.189 would have complicated the presentation for no immediate gain, so it seemed kinder on the reader to build the special case first by hand.

Note that we *need* to use ground terms now, for the proof of Theorem 2.220 to work. The reason is that $\mathscr{F}(\mathsf{T})$ has elements in each sort given by the elements $[X]_{\mathsf{T}}$, whereas $\mathscr{G}(\mathsf{T})$ lacks these elements.

2.3.5.5 Surjective Maps onto Algebras

Fix a signature Σ and any collection of Σ-algebras $\mathcal{V}$.

Definition 2.218. Suppose $\mathsf{T} = (\Sigma, Ax)$ and suppose $\mathscr{X}$ and $\mathscr{Y}_i$ for $i \in I$ are models of T. Suppose $\theta_i \in \mathscr{X} \to \mathscr{Y}_i$ is a family of homomorphisms.

Write $\Pi_i \theta_i$ for the natural map from $\mathscr{X}$ to $\Pi_i \mathscr{Y}_i$, mapping $x \in |\mathscr{X}_\alpha|$ to $(\theta_i(x))_i \in |\Pi_i \mathscr{Y}_i|$.

It is easy to verify that $\Pi_i \theta_i$ is a Σ-algebra homomorphism.

Definition 2.219. Suppose Σ and Σ' are signatures. Say Σ' **extends** Σ **with fresh constants** when $\Sigma = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{X}, \mathcal{F}, ar')$ and $\Sigma' = (\mathcal{A}, \mathcal{B}, \mathcal{C} \cup \mathcal{D}, \mathcal{X}, \mathcal{F}, ar')$ where $\mathcal{D} \cap \mathcal{C} = \varnothing$ and $ar'(C) = ar(C)$ for every $C \in \mathcal{C}$.

Theorem 2.220. *Suppose $\mathsf{T} = (\Sigma, Ax)$ is a theory and $\mathscr{V}$ is a model of T. Then there exists a theory $\mathsf{T}' = (\Sigma', Ax)$ where Σ' extends Σ with some fresh constants $\mathcal{D}$ such that $\mathscr{V}$ is a homomorphic image of $\mathscr{G}(\mathsf{T}')$.*

Proof. We take $\mathcal{D} = \bigcup_\alpha |\mathscr{V}_\alpha|$ and construct a homomorphism based on mapping $x \in \mathscr{V}_\alpha$ (as a constant in $\mathcal{D}$) to itself (as an element of $|\mathscr{V}_\alpha|$). ■

2.3.5.6 Injections Out of Free Algebras

Definition 2.221. Suppose Σ is a signature and $\mathcal{V}$ is a set of Σ-algebras. Let $\mathsf{T} = (\Sigma, Ax)$ where Ax is the collection of judgements valid in all $\mathscr{V} \in \mathcal{V}$ for all valuations. Call T the (Σ-)theory **generated by** $\mathcal{V}$.

Remark 2.222. So $(r = s) \in Ax$ in Definition 2.221 when for every $\mathscr{V} \in \mathcal{V}$ and every valuation ς to $\mathscr{V}$, it is the case that $[\![r]\!]^{\mathscr{V}}_{\varsigma} = [\![s]\!]^{\mathscr{V}}_{\varsigma}$.

Definition 2.223. Define the **constants** of a term $consts(r)$ just as Definition 2.59 except that we take $consts(C) = \{orb(C)\}$ and $consts(X) = \varnothing$.

Lemma 2.224. *Suppose Σ is a signature and Σ' extends Σ with some fresh constants $\mathcal{D}$. Suppose Σ has enough unknowns (Definition 2.48).*

If g is a ground term in Σ' then there exists a term g^{-1} in Σ and substitution θ such that $g^{-1}\theta = g$.

Proof. For each orbit in $consts(r)$ choose a representative $C \in orb(C) \in consts(r)$, and some distinct unknown X_C with $sort(X_C) = sort(C)$ and $supp(C) \subseteq supp(X_C)$—we can do this because we have assumed enough unknowns and it is a fact that $consts(r)$ is finite. Define θ to be the equivariant extension of this choice, so $\theta(\pi \cdot X_C) = \pi \cdot C$ and (for all the other unknowns) $\theta(Y) = Y$. This is well-defined by Proposition 2.38.

It is now easy to generate g^{-1} by replacing each C in g with X_C (modulo some permutations). ■

Theorem 2.225. *Suppose $\mathcal{V}$ is a collection of Σ-algebras and Σ has enough unknowns. Let $\mathsf{T} = (\Sigma, Ax)$ be the Σ-theory generated by $\mathcal{V}$. Suppose Σ' extends Σ with some fresh constants $\mathcal{D}$ and write $\mathsf{T}' = (\Sigma', Ax)$.*

Then there exists some indexing set I, set of algebras $\{\mathscr{V}_i \in \mathcal{V} \mid i \in I\}$, and an injective Σ-algebra homomorphism Θ from $\mathscr{G}(\Sigma')$ to $\Pi_{i \in I} \mathscr{V}_i$.

Proof. Take I to be the set of all pairs (g,h) of ground terms in Σ' such that $\mathsf{T}' \not\vdash g = h$.

Consider some $i = (g,h) \in I$. By Lemma 2.224 there exist g^{-1}, h^{-1}, and θ_i such that $g^{-1}\theta_i = g$ and $h^{-1}\theta_i = h$. We assumed that $\mathsf{T}' \not\vdash g = h$ and it follows using Lemma 2.175 that $\mathsf{T} \not\vdash g^{-1} = h^{-1}$. Since T is the theory generated by $\mathcal{V}$ there exists a model $\mathscr{V}_i \in \mathcal{V}$ and a valuation ς such that $[\![g^{-1}]\!]^{\mathscr{V}_i}_{\varsigma} \neq [\![h^{-1}]\!]^{\mathscr{V}_i}_{\varsigma}$. We define a Σ-homomorphism Θ_i from $\mathscr{G}(\mathsf{T}')$ to $\mathscr{V}_i$ as an equivariant extension of mapping $C \in \mathcal{D}$ to $\varsigma(X_C)$, where C and X_C are as chosen in the proof of Lemma 2.224.

It follows by the choice of $\mathscr{V}_i$ that $\Pi_{i \in I}\theta_i$ from $\mathscr{G}(\mathsf{T}')$ to $\Pi_{i \in I}\mathscr{V}_i$ is injective as a map on underlying sets. ■

2.3.5.7 Proof of the HSP Theorem

We can now prove Theorem 2.228; a similar result for nominal algebra is proved in Gabbay (2009).

Definition 2.226. Suppose Σ is a signature. Suppose $\mathcal{V}$ is a collection of Σ-algebras. Then:

- Call $\mathcal{V}$ a **(Σ-)variety** when it is closed under Homomorphic image (Definition 2.204), Subalgebra (Definition 2.208), and countable Product (Definition 2.210).
- Call $\mathcal{V}$ **(Σ-)equational** when it is the collection of Σ-algebras that are models of $\mathsf{T} = (\Sigma, Ax)$ for some set of axioms Ax.

Lemma 2.227. *Suppose Σ is a signature with enough unknowns. Suppose $\mathcal{V}$ is a Σ-variety and let $\mathsf{T} = (\Sigma, Ax)$ be the Σ-theory generated by $\mathcal{V}$. Suppose Σ' extends Σ with some fresh constants $\mathcal{D}$ and write $\mathsf{T}' = (\Sigma', Ax)$. Then $\mathscr{G}(\mathsf{T}') \in \mathcal{V}$.*

Proof. By Theorem 2.225 there is some indexing set I, set of Σ-algebras $\{\mathscr{V}_i \in \mathcal{V} \mid i \in I\}$, and injective Σ-algebra homomorphism Θ from $\mathscr{G}(\mathsf{T}')$ to $\Pi_{i \in I}\mathscr{V}_i$. $\mathcal{V}$ is closed

under products so $\Pi_{i\in I}\mathscr{V}_i \in \mathcal{V}$. The image of $|\mathscr{G}(\mathsf{T}')|$ is a subalgebra of $\Pi_{i\in I}\mathscr{V}_i$, and is a homomorphic image of that subalgebra (by inverting Θ). $\mathcal{V}$ is closed under subalgebras and homomorphic images, so the result follows. ■

Theorem 2.228. *Suppose Σ is a signature with enough unknowns. A collection of Σ-algebras $\mathcal{V}$ is equational if and only if it is a variety*

Proof. Suppose $\mathcal{V}$ is equational. By Lemma 2.211 $\mathcal{V}$ is closed under products. By Lemma 2.207 $\mathcal{V}$ is closed under homomorphic images. By Lemma 2.209 $\mathcal{V}$ is closed under subalgebras. Therefore $\mathcal{V}$ is a variety.

Conversely, suppose $\mathcal{V}$ is a variety. Let T be the theory on Σ generated by $\mathcal{V}$ as described in Definition 2.221. Let $\mathscr{V}$ be any model of T. By Theorem 2.220 there exists a signature Σ' extending Σ with some fresh constants $\mathcal{D}$ such that $\mathscr{V}$ is a homomorphic image of $\mathscr{G}(\mathsf{T}')$. By Lemma 2.227 $\mathscr{G}(\mathsf{T}') \in \mathcal{V}$. Since $\mathcal{V}$ is closed under homomorphisms, $\mathscr{V} \in \mathcal{V}$ as required. Therefore $\mathcal{V}$ is equational. ■

2.4 Permissive-Nominal Logic: $\forall X$

2.4.1 *Permissive-Nominal Logic*

We now add quantification over unknowns—that is, $\forall X$—to permissive-nominal terms. Permissive nominal techniques makes the theory of α-equivalence easy here: if we used 'vanilla' nominal terms, then the development below might not be impossible, but we believe that it would be harder. We obtain a first-order logic which we call *permissive-nominal logic*.

Permissive-nominal logic (PNL) is just first-order logic, enriched with nominal-style names. Thus, the derivation rules in Fig. 2.5 are (virtually) identical to those of first-order logic. Only the term language is really changed.

$$\frac{}{\Phi, \phi \vdash \pi\cdot\phi, \Psi}\,(\mathbf{Ax}) \qquad \frac{}{\Phi, \bot \vdash \Psi}\,(\bot\mathbf{L})$$

$$\frac{\Phi \vdash \phi, \Psi \quad \Phi, \psi \vdash \Psi}{\Phi, \phi \Rightarrow \psi \vdash \Psi}\,(\Rightarrow\mathbf{L}) \qquad \frac{\Phi, \phi \vdash \psi, \Psi}{\Phi \vdash \phi \Rightarrow \psi, \Psi}\,(\Rightarrow\mathbf{R})$$

$$\frac{\Phi, \phi[X{:=}r] \vdash \Psi \quad (fa(r){\subseteq}supp(X),\ r{:}sort(X))}{\Phi, \forall X.\phi \vdash \Psi}\,(\forall\mathbf{L}) \qquad \frac{\Phi \vdash \phi, \Psi \quad (X \notin fU(\Phi, \Psi))}{\Phi \vdash \forall X.\phi, \Psi}\,(\forall\mathbf{R})$$

$$\frac{\Phi \vdash \phi, \Psi \qquad \Phi, \phi \vdash \Psi}{\Phi \vdash \Psi}\,(\mathbf{Cut})$$

Fig. 2.5 Sequent derivation rules of Permissive-Nominal Logic

In this section we set up PNL as a sequent derivation system (Fig. 2.5), interpret it in permissive-nominal sets (Definition 2.241), and prove soundness and completeness (Theorems 2.245 and 2.261).

In Sect. 2.4.2 we undertake as an extended case study a sound and complete finite axiomatisation of arithmetic inside PNL.

2.4.1.1 Syntax

The notions of sort-signature $(\mathcal{A},\mathcal{B})$ and sort language are as in Definition 2.39. An interpretation $\mathscr{I}$ for $(\mathcal{A},\mathcal{B})$ consists of an assignment of a permissive-nominal set $\tau^{\mathscr{I}}$ to each $\tau\in\mathcal{B}$, and we extend $\mathscr{I}$ to sorts as in Definition 2.176.

Definition 2.229. For this section it is convenient to take $\mathcal{X}$ to be specifically example 2 of Example 2.45.

Remark 2.230. So an unknown X takes the form

$$\pi\cdot\mathtt{X}_\alpha=\{(\pi',\mathtt{X}_\alpha)\mid\forall a{\in}\mathbb{A}^{<}.\pi(a)=\pi'(a)\}.$$

In this case, in the light of Remark 2.46, we may take $fU(r)$ to be equal to the set of $\mathtt{X}_\alpha$ occurring in r.

It is now easy to define binding of unknowns (level 2 variables) in terms. A more abstract account of level 2 binding is also available Gabbay (2011c).

Definition 2.231. A **PNL signature** over a sort-signature $(\mathcal{A},\mathcal{B})$ is a tuple $(\mathcal{C},\mathcal{F},\mathcal{P},ar)$ where:

- $\mathcal{C}$ is a permissive-nominal set of **constants**.
- $\mathcal{F}$ is a set of equivariant **term-formers**.
- $\mathcal{P}$ is a set of equivariant **predicate-formers**.
- ar assigns
  - to each constant $C\in\mathcal{C}$ an arity τ,
  - to each $\mathsf{f}\in\mathcal{F}$ a **term-former arity** $(\alpha)\tau$, and
  - to each $\mathsf{P}\in\mathcal{P}$ a **proposition-former arity** α, where

  α and τ are in the sort-language determined by $(\mathcal{A},\mathcal{B})$.

A **(PNL) signature** $\mathcal{S}$ is then a tuple $(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{F},\mathcal{P},ar)$.

Definition 2.232. Suppose $\mathcal{S}=(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{F},\mathcal{P},ar)$ is a PNL signature.

Terms are defined as in Definition 2.49 for the signature $(\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{X},\mathcal{F},ar)$.[17]

[17]The reader who would answer 'Can you pass the salt?' with 'Yes.' should note that we have to adjust ar to remove $\mathcal{P}$ and add $\mathcal{X}$ mapping X to α where $(\pi,\mathtt{X}_\alpha)\in X$.

Propositions are defined as follows:

$$\frac{}{\bot \text{ proposition}} \qquad \frac{\phi \text{ proposition} \quad \psi \text{ proposition}}{\phi \Rightarrow \psi \text{ proposition}}$$

$$\frac{r : \alpha \quad (ar(\mathsf{P}) = \alpha)}{\mathsf{P}(r) \text{ proposition}} \qquad \frac{\phi \text{ proposition}}{\forall \mathtt{X}_\alpha.\phi \text{ proposition}}$$

Here $\forall \mathtt{X}_\alpha$ binds $\mathtt{X}_\alpha$ in ϕ. We can use nominal abstract syntax to do this.

Notation 2.233. Write $\forall X.\phi$ as shorthand for $\forall \mathtt{X}_\alpha.\phi$ where $X = \{(\pi', \mathtt{X}_\alpha) \mid \forall a \in \mathbb{A}^{<}.\pi'(a) = \pi(a)\}$ for some π.

Lemma 2.234. *Support and the permutation action as characterised for terms on Lemma 2.53 extend to propositions as follows:*

$$supp(\bot) = \varnothing \qquad supp(\mathsf{P}(r)) = supp(r)$$
$$supp(\phi \Rightarrow \psi) = supp(\phi) \cup supp(\psi) \qquad supp(\forall X.\phi) = supp(\phi)$$

$$\pi \cdot \bot = \bot \qquad \pi \cdot \mathsf{P}(r) = \mathsf{P}(\pi \cdot r)$$
$$\pi \cdot (\phi \Rightarrow \psi) = (\pi \cdot \phi) \Rightarrow \pi \cdot \psi \qquad \pi \cdot \forall X.\phi = \forall X.\pi \cdot \phi$$

Notation 2.235. We may write $fa(\phi)$ for $supp(\phi)$.

2.4.1.2 Derivability

Definition 2.236. Φ and Ψ will range over sets of propositions. We may write ϕ, Φ and Φ, ϕ as shorthand for $\{\phi\} \cup \Phi$. We may write Φ, Ψ as shorthand for $\Phi \cup \Psi$.

Write $fU(\Phi) = \bigcup\{fU(\phi) \mid \phi \in \Phi\}$.

A **sequent** is a pair $\Phi \vdash \Psi$.

Definition 2.237 (Derivable sequents). The **derivable sequents** are defined in Fig. 2.5.

Remark 2.238. As standard, the intuition of $\Phi \vdash \Psi$ being derivable is "the conjunction (logical and) of the propositions in Φ entails the disjunction (logical or) of the propositions in Ψ". So for instance, intuitively the axiom rule (**Ax**) expresses that ϕ if and only if $\pi \cdot \phi$.

The permutation π in (**Ax**) is deliberate and represents equivariance (preservation of truth under permuting atoms) within permissive-nominal logic. Because of this permutation π, free atoms can behave like variables ranging over distinct atoms.

Thus in PNL we can express a theory of atoms-inequality as follows: Suppose a name sort Atm and a proposition-former $\mathsf{neq} : (\mathsf{Atm}, \mathsf{Atm})$, along with a single proposition $\mathsf{neq}(a, b)$ for two distinct atoms in Atm—and, if we wish, another

proposition $\mathsf{neq}(a,a) \Rightarrow \bot$. The permutation π in $(\mathbf{Ax})$ ensures that a and b represent *any* two distinct atoms.

Remark 2.239. The condition $fa(r) \subseteq supp(X)$ in $(\forall\mathbf{L})$ might suggest that $\forall X.\phi$ means "$\phi[X{:=}r]$ for every r such that $fa(r) \subseteq supp(X)$". This is so, but the π in $(\mathbf{Ax})$ means that what $supp(X)$ in $\forall X.\phi$ really restricts is *capture*, as we now discuss.

- Suppose a name sort Atm and suppose $X : \mathsf{Atm}$ and $\mathsf{P} : \mathsf{Atm}$. Suppose $b \in pmss(X)$. By considering the swapping $(b\ a)$ and $(\mathbf{Ax})$, and $(\forall\mathbf{L})$, $\forall X.\mathsf{P}(X) \vdash \mathsf{P}(b)$ for *all* a, even if $a \notin supp(X)$, as follows:

$$\dfrac{\dfrac{}{\mathsf{P}(b) \vdash \mathsf{P}(a)}\ (\mathbf{Ax}) \quad \pi = (b\ a)}{\forall X.\mathsf{P}(X) \vdash \mathsf{P}(a)}\ (\forall\mathbf{L}) \quad [X{:=}b]$$

  So we can derive $\mathsf{P}(a)$ from $\forall X.\mathsf{P}(X)$, even if a is not permitted in X.
- This may not work if we have to 'shift' infinitely many atoms; e.g. there is no finite π such that $fa(\pi{\cdot}(X,a)) \subseteq supp(X)$ where $a \notin supp(X)$. If $\mathbb{P}$ has *shift*-permutations (Definition 2.79), then we can use them.

  Consider any sort α and suppose $X : \alpha$ and $supp(X) = S$. Suppose $\mathsf{Q} : \alpha$. Consider any other $Y : \alpha$ with $supp(Y) = S \cup \{a\}$ where $a \notin S$. We will show that given *shift*-permutations, $\forall X.\mathsf{Q}(X) \vdash \mathsf{Q}(Y)$ is derivable.

  Suppose $S \cup \{a\} = \pi{\cdot}S$. We derive as follows:

$$\dfrac{\dfrac{}{\mathsf{Q}(\pi{\cdot}Y) \vdash \mathsf{Q}(Y)}\ (\mathbf{Ax})}{\forall X.\mathsf{Q}(X) \vdash \mathsf{Q}(Y)}\ (\forall\mathbf{L}) \quad [X{:=}\pi{\cdot}Y]$$

- Nevertheless, $\forall X.\phi$ does not mean "$\phi[X{:=}r]$ for every r". This is because permutations are bijective. For example, suppose $X : \mathsf{Atm}$, $a \notin supp(X)$, and $\mathsf{P} : ([\mathsf{Atm}]\mathsf{Atm})$. Then $\forall X.\mathsf{P}([a]X) \vdash P([a]r)$ for all r such that $a \notin fa(r)$, and also $\forall X.\mathsf{P}([b]X) \vdash P([b]r)$ for all r and all b such that $b \notin fa(r)$. However,

$$\forall X.\mathsf{P}([a]X) \nvdash P([a]a), \quad \text{and for all } b,\ \forall X.\mathsf{P}([a]X) \nvdash P([b]b).$$

  The fact that $a \notin supp(X)$ forbids capture by an instantiation, in a suitable sense.

2.4.1.3 Interpretation and Soundness

Definition 2.240. Suppose $\mathcal{S} = (\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{F},\mathcal{P},ar)$ is a signature.
A **(PNL) interpretation** $\mathscr{I}$ for $\mathcal{S}$ consists of the following data:

- An interpretation for the sort-signature $(\mathcal{A},\mathcal{B})$ (Definition 2.176).
- For every $\mathsf{f} \in \mathcal{F}$ with $ar(\mathsf{f}) = (\alpha')\alpha$ an equivariant function $\mathsf{f}^{\mathscr{I}}$ from $[\![\alpha']\!]^{\mathscr{I}}$ to $[\![\alpha]\!]^{\mathscr{I}}$ (Definition 2.19).

- For every $\mathsf{P} \in \mathcal{P}$ with $ar(\mathsf{P}) = \alpha$ an equivariant function $\mathsf{P}^{\mathscr{I}}$ from $[\![\alpha]\!]^{\mathscr{I}}$ to $\{0,1\}$ (Definition 2.23).

Definition 2.241. Suppose that $\mathscr{I}$ is an interpretation. Define an **interpretation of propositions** by:

$$\begin{array}{rl}
[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma} & = \mathsf{P}^{\mathscr{I}}([\![r]\!]^{\mathscr{I}}_{\varsigma}) \\
[\![\bot]\!]^{\mathscr{I}}_{\varsigma} & = 0 \\
[\![\phi \Rightarrow \psi]\!]^{\mathscr{I}}_{\varsigma} & = max\{1 - [\![\phi]\!]^{\mathscr{I}}_{\varsigma}, [\![\psi]\!]^{\mathscr{I}}_{\varsigma}\} \\
[\![\forall X.\phi]\!]^{\mathscr{I}}_{\varsigma} & = min\{[\![\phi]\!]^{\mathscr{I}}_{\varsigma[X:=x]} \mid x \in [\![sort(X)]\!]^{\mathscr{I}}, supp(x) \subseteq supp(X)\}
\end{array}$$

Lemma 2.242. $[\![\phi]\!]^{\mathscr{I}}_{\varsigma} = [\![\pi \cdot \phi]\!]^{\mathscr{I}}_{\varsigma}$ *always.*

Proof. By induction on ϕ. We consider two cases:

- The case $\forall X.\phi$. Suppose $[\![\forall X.\phi]\!]^{\mathscr{I}}_{\varsigma} = 1$. This means that $[\![\phi]\!]^{\mathscr{I}}_{\varsigma[X:=x]} = 1$ for all $x \in [\![\alpha]\!]^{\mathscr{I}}$ such that $supp(x) \subseteq supp(X)$. By inductive hypothesis $[\![\pi \cdot \phi]\!]^{\mathscr{I}}_{\varsigma[X:=x]} = 1$ for all $x \in [\![\alpha]\!]^{\mathscr{I}}$ such that $supp(x) \subseteq supp(X)$. Therefore $[\![\forall X.\pi \cdot \phi]\!]^{\mathscr{I}}_{\varsigma} = 1$. The result follows, since $\pi \cdot (\forall X.\phi) = \forall X.\pi \cdot \phi$.
- The case $\mathsf{P}(r)$. We have $[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma} = \mathsf{P}^{\mathscr{I}}([\![r]\!]^{\mathscr{I}}_{\varsigma})$. As $\mathsf{P}^{\mathscr{I}}$ is equivariant, we get $[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma} = \mathsf{P}^{\mathscr{I}}(\pi \cdot [\![r]\!]^{\mathscr{I}}_{\varsigma})$. By Lemma 2.181 $\pi \cdot [\![r]\!]^{\mathscr{I}}_{\varsigma} = [\![\pi \cdot r]\!]^{\mathscr{I}}_{\varsigma}$. Thus $[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma} = \mathsf{P}^{\mathscr{I}}([\![\pi \cdot r]\!]^{\mathscr{I}}_{\varsigma}) = [\![\pi \cdot \mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma}$.

■

Lemma 2.243. $[\![\phi]\!]^{\mathscr{I}}_{\varsigma[X:=[\![t]\!]^{\mathscr{I}}_{\varsigma}]} = [\![\phi[X:=t]]\!]^{\mathscr{I}}_{\varsigma}$.

Proof. By a routine induction on the definition of $[\![\phi]\!]^{\mathscr{I}}_{\varsigma}$ in Definition 2.241. We consider one case:

- The case of $[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma[X:=t]}$. We reason as follows:

$$\begin{array}{lll}
[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma[X:=[\![t]\!]^{\mathscr{I}}_{\varsigma}]} & = \mathsf{P}^{\mathscr{I}}([\![r]\!]^{\mathscr{I}}_{\varsigma[X:=[\![t]\!]^{\mathscr{I}}_{\varsigma}]}) & \text{Definition 2.241} \\
& = \mathsf{P}^{\mathscr{I}}([\![r[X:=t]]\!]^{\mathscr{I}}_{\varsigma}) & \text{Lemma 2.187} \\
& = [\![\mathsf{P}(r)[X:=t]]\!]^{\mathscr{I}}_{\varsigma} & \text{Definition 2.241.}
\end{array}$$

■

Validity and soundness

Definition 2.244 (Validity). Call the proposition ϕ **valid** in $\mathscr{I}$ when $[\![\phi]\!]^{\mathscr{I}}_{\varsigma} = 1$ for all ς.

Call the sequent $\phi_1, \ldots, \phi_n \vdash \psi_1, \ldots, \psi_p$ **valid** in $\mathscr{I}$ when $(\phi_1 \wedge \cdots \wedge \phi_n) \Rightarrow (\psi_1 \vee \cdots \vee \psi_p)$ is valid.

Theorem 2.245 (Soundness). *If* $\Phi \vdash \Psi$ *is derivable, then it is valid in all interpretations.*

Proof. By induction on derivations (Fig. 2.5). The case of (**Ax**) uses Lemma 2.242. The case of ($\forall$**L**) uses Lemma 2.243. The case of ($\forall$**R**) uses Lemma 2.184. Other rules are routine by unpacking definitions. ■

2.4.1.4 Completeness

In this section we prove Theorem 2.261: a converse to Theorem 2.245, that if ϕ is valid in all interpretations, then ϕ it is derivable.

For this section fix the following data:

- A signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{F}, \mathcal{P}, ar)$.
- A formula ϕ such that $\nvdash \phi$.

We will construct an interpretation $\mathscr{I}$ and a valuation ς (Definition 2.176) such that $[\![\phi]\!]^{\mathscr{I}}_{\varsigma} = 0$. This suffices to prove the result.

Maximally consistent set of propositions

Definition 2.246. Make a fixed but arbitrary order on propositions $\xi_1, \xi_2, \xi_3, \ldots$

Define $\Phi_1 = \{\neg\phi\}$ (where ϕ was fixed above). For each $i \geq 1$ we define Φ_{i+1} as follows:

- If $\Phi_i \vdash \xi_i$ then write $\xi = \xi_i$.
- If $\Phi_i \vdash \neg\xi_i$ then write $\xi = \neg\xi_i$.
- If $\Phi_i \nvdash \xi_i$ and $\Phi_i \nvdash \neg\xi_i$ then write $\xi = \xi_i$.

There are now two cases:

- If ξ has the form $\neg\forall X.\xi'$ then we define $\Phi_{i+1} = \Phi_i \cup \{\xi, \neg\xi'[X{:=}Z]\}$ where Z is some fixed but arbitrary choice of unknown that is not free in any proposition in Φ_i and is such that $supp(Z) = supp(X)$ and $sort(Z) = sort(X)$.
- Otherwise, we define $\Phi_{i+1} = \Phi_i \cup \{\xi\}$.

Finally, we define Φ_ω by $\Phi_\omega = \bigcup_i \Phi_i$.

Lemma 2.247. *For every i, $\Phi_i \nvdash \bot$.*

Proof. By induction on i:

- By definition $\Phi_1 = \{\neg\phi\}$. As $\nvdash \phi$, we have $\neg\phi \nvdash \bot$
- Suppose $\Phi_i \nvdash \bot$.
  Either $\Phi_{i+1} = \Phi_i \cup \{\neg\xi\}$ or $\Phi_{i+1} = \Phi_i \cup \{\neg\xi, \neg\xi'[X{:=}Z]\}$—we consider the first, simpler case; the second case is similar. Suppose $\Phi_i, \xi \vdash \bot$. It follows that $\Phi_i \vdash \neg\xi$. So we are not in the third case of Definition 2.246 and we are either in the first or the second. So $\Phi_i \vdash \xi$ and thus $\Phi_i \vdash \bot$; a contradiction.

■

Lemma 2.248. $\Phi_\omega \nvdash \bot$.

Proof. Assume $\Phi_\omega \vdash \bot$. So there exists a finite subset Γ of Φ_ω such that $\Gamma \vdash \bot$. As Γ is finite it is included in some Φ_i, and $\Phi_i \vdash \bot$, contradicting Proposition 2.247. ■

Remark 2.249. It is well-known that in nominal sets, least upper bounds can sometimes not exist if there are 'too many' atoms; so sometimes we have to be careful to not make too many arbitrary choices.[18]

The reader familiar with nominal techniques will be alert to the possibility that Φ_ω might fail to have a supporting permission set, and that this could cause problems. In fact, in this particular case this is a non-issue: (**Ax**) from Fig. 2.5 ensures that Φ_ω is not only supported, but in fact equivariant.

Lemma 2.250. *For every ξ, at least one of $\xi \in \Phi_\omega$ and $\neg\xi \in \Phi_\omega$ holds.*

Proof. We check of Definition 2.246 that for every i, either $\xi_i \in \Phi_{i+1}$ or $\neg\xi_i \in \Phi_{i+1}$. The result follows. ■

Lemma 2.251. *For every ξ, if $\neg\forall X.\xi \in \Phi_\omega$ then there exists a Z such that $\neg\xi[X{:=}Z] \in \Phi_\omega$.*

Proof. There exists an i such that $\xi_i = \neg\forall X.\xi$. Since $\Phi_\omega \vdash \xi_i$ and $\Phi_\omega \nvdash \bot$, we have that $\Phi_\omega \nvdash \neg\xi_i$, and so $\Phi_i \nvdash \neg\xi_i$. Thus $\Phi_{i+1} = \Phi_i \cup \{\neg\forall X.\xi,\ \neg\xi[X{:=}Z]\}$. The result follows. ■

Lemma 2.252. *If $\Phi_\omega \vdash \phi$ then $\phi \in \Phi_\omega$.*

Proof. As, by Lemma 2.248, $\Phi_\omega \nvdash \bot$, if $\Phi_\omega \vdash \phi$ then $\neg\phi \notin \Phi_\omega$. Thus by Lemma 2.250, $\phi \in \Phi_\omega$. ■

Corollary 2.253.

1. *$(\phi \Rightarrow \psi) \in \Phi_\omega$ if and only if ($\phi \notin \Phi_\omega$ or $\psi \in \Phi_\omega$).*
2. *$\forall X.\phi \in \Phi_\omega$ if and only if*
   (for every r such that $r : sort(X)$ and $fa(r) \subseteq supp(X)$, $\phi[X{:=}r] \in \Phi_\omega$).

Proof.

1. Suppose $(\phi \Rightarrow \psi) \in \Phi_\omega$ and $\phi \in \Phi_\omega$. Then $\Phi_\omega \vdash \psi$ and so by Lemma 2.252 $\psi \in \Phi_\omega$.
   Suppose $\phi \notin \Phi_\omega$. By Lemma 2.250 $\neg\phi \in \Phi_\omega$. So $\Phi_\omega \vdash \neg\phi$ and therefore $\Phi_\omega \vdash \phi \Rightarrow \psi$. By Lemma 2.252 $(\phi \Rightarrow \psi) \in \Phi_\omega$.
   Suppose $\psi \in \Phi_\omega$. Then $\Phi_\omega \vdash \psi$ and so $\Phi_\omega \vdash \phi \Rightarrow \psi$. By Lemma 2.252 $(\phi \Rightarrow \psi) \in \Phi_\omega$.
2. Suppose $\forall X.\phi \in \Phi_\omega$. By Lemma 2.252, if $r : sort(X)$ and $fa(r) \subseteq supp(X)$ then $\phi[X{:=}r] \in \Phi_\omega$.

[18] For instance, in permissive-nominal sets it is possible represent a well-order of each permission set, but not to represent a well-ordering of the set of all atoms (which is a limit of permission sets). This is also the feature which Fraenkel and Mostowksi used to prove the independence of the axiom of choice from the other axioms of set theory.

Conversely, suppose $\phi[X{:=}r] \in \Phi_\omega$ for every r such that $r : sort(X)$ and $fa(r) \subseteq supp(X)$. We proceed by contradiction: suppose $\forall X.\phi \notin \Phi_\omega$. By Lemma 2.250 $\neg\forall X.\phi \in \Phi_\omega$ and by Lemma 2.251, there exists a Z such that $\neg\phi[X{:=}Z] \in \Phi_\omega$. So $\Phi_\omega \vdash \neg\phi[X{:=}Z]$, and so $\Phi_\omega \vdash \phi[X{:=}Z]$, and so $\Phi_\omega \vdash \bot$, contradicting Lemma 2.248.

■

The term interpretation

Definition 2.254. Define the **term interpretation** $\mathscr{I}$ by:

- $[\![\alpha]\!]^{\mathscr{I}} = \{r \mid r : \alpha\}$.
- $\mathsf{f}^{\mathscr{I}}$ maps r to $\mathsf{f}(r)$.
- $\mathsf{P}^{\mathscr{I}}$ maps $r_1, \ldots, r_n$ to 1 if $\mathsf{P}(r_1, \ldots, r_n) \in \Phi_\omega$ and to 0 otherwise.

Define ς by $\varsigma(X) = X$ for all $X \in \mathcal{X}$ and endow $[\![\alpha]\!]^{\mathscr{I}}$ with a permutation action given by the action on terms.

Remark 2.255. In Definition 2.192 we built an interpretation to prove completeness of nominal algebra (Corollary 2.200). There, we built our interpretation out of terms quotiented by derivable equality; here we just use terms. Why the difference?

In nominal algebra the judgement-form of the logic *is* equality—so it makes sense to build an interpretation such that equality maps to denotational identity.

Lemma 2.256.

1. *If* $ar(\mathsf{f}) = (\alpha)\tau$ *then* $\mathsf{f}^{\mathscr{I}}$ *is well-defined, equivariant, and maps* $[\![\alpha]\!]^{\mathscr{I}}$ *to* $[\![\tau]\!]^{\mathscr{I}}$.
2. *If* $ar(\mathsf{P}) = \alpha$ *then* $\mathsf{P}^{\mathscr{I}}$ *is well-defined, equivariant, and maps* $[\![\alpha]\!]^{\mathscr{I}}$ *to* $\{0,1\}$.

Proposition 2.257. *$\mathscr{I}$ is an interpretation of the signature $\mathcal{S} = (\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{P}, ar)$ which we fixed at the beginning of this section. In addition, ς is a valuation to $\mathscr{I}$.*

Proof. By Lemma 2.256 for each $\mathsf{f} : (\alpha')\alpha \in \mathcal{F}$, $\mathsf{f}^{\mathscr{I}}$ is an equivariant map from $[\![\alpha']\!]^{\mathscr{I}}$ to $[\![\alpha]\!]^{\mathscr{I}}$ and for each $\mathsf{P} : \alpha \in \mathcal{P}$, $\mathsf{P}^{\mathscr{I}}$ is an equivariant function from $[\![\alpha]\!]^{\mathscr{I}}$ to $\{0,1\}$.

By construction $\varsigma(X) \in [\![sort(X)]\!]^{\mathscr{I}}$ always. Equivariance is easy. ■

Lemma 2.258. $[\![r]\!]^{\mathscr{I}}_{\varsigma} = r$.

Lemma 2.259. $[\![\xi]\!]^{\mathscr{I}}_{\varsigma} = 1$ *if and only if* $\xi \in \Phi_\omega$.

Proof. By induction on the definition of $[\![\xi]\!]^{\mathscr{I}}_{\varsigma}$ (Definition 2.241):

- The case of $[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma}$. We reason as follows:

$$
\begin{array}{lll}
[\![\mathsf{P}(r)]\!]^{\mathscr{I}}_{\varsigma}=1 & \Leftrightarrow \mathsf{P}^{\mathscr{I}}([\![r]\!]^{\mathscr{I}}_{\varsigma}) = 1 & \text{Definition 2.241} \\
& \Leftrightarrow \mathsf{P}^{\mathscr{I}}(r) = 1 & \text{Lemma 2.258} \\
& \Leftrightarrow \mathsf{P}(r) \in \Phi_\omega & \text{Definition 2.254}
\end{array}
$$

- The case of $[\![\bot]\!]^{\mathscr{I}}_{\varsigma}$. By definition $[\![\bot]\!]^{\mathscr{I}}_{\varsigma} = 0$. By part 1 of Corollary 2.253, $\bot \notin \Phi_\omega$.
- The case of $[\![\phi \Rightarrow \psi]\!]^{\mathscr{I}}_{\varsigma}$. We reason as follows:

$$\begin{array}{lll} [\![\phi \Rightarrow \psi]\!]^{\mathscr{I}}_{\varsigma}=1 \Leftrightarrow & [\![\phi]\!]^{\mathscr{I}}_{\varsigma}=0 \text{ or } [\![\psi]\!]^{\mathscr{I}}_{\varsigma}=1 & \text{Definition 2.241} \\ \Leftrightarrow & \phi \notin \Phi_\omega \text{ or } \psi \in \Phi_\omega & \text{ind. hyp.} \\ \Leftrightarrow & \phi \Rightarrow \psi \in \Phi_\omega & \text{Cor. 2.253, part 2} \end{array}$$

- The case of $[\![\forall X.\phi]\!]^{\mathscr{I}}_{\varsigma}$, where $\alpha = sort(X)$ and $S = supp(X)$.

$$\begin{array}{lll} [\![\forall X.\phi]\!]^{\mathscr{I}}_{\varsigma}=1 & \Leftrightarrow & \forall t \in [\![\alpha]\!]^{\mathscr{I}}.\, supp(t) \subseteq S \Rightarrow [\![\phi]\!]^{\mathscr{I}}_{\varsigma[X:=t]}=1 \quad \text{Definition 2.241} \\ & \Leftrightarrow & \forall t \in [\![\alpha]\!]^{\mathscr{I}}.\, supp(t) \subseteq S \Rightarrow [\![\phi[X:=t]]\!]^{\mathscr{I}}_{\varsigma} = 1 \quad \text{Lems. 2.184, 2.258} \\ & \Leftrightarrow & [\![\phi[X:=t]]\!]^{\mathscr{I}}_{\varsigma} = 1 \text{ every } t:\alpha \text{ s.t. } fa(t) \subseteq S \quad supp(t) = fa(t) \\ & \Leftrightarrow & \phi[X:=t] \in \Phi_\omega \text{ every } t:\alpha \text{ s.t. } fa(t) \subseteq S \quad \text{ind. hyp.} \\ & \Leftrightarrow & \forall X.\phi \in \Phi_\omega \quad \text{Cor. 2.253, part 4} \end{array}$$

■

Lemma 2.260. *If $\not\vdash \phi$, then there exists an interpretation $\mathscr{I}$ and a valuation ς such that $[\![\phi]\!]^{\mathscr{I}}_{\varsigma} = 0$.*

Proof. As $\neg\phi \in \Phi_0 \subseteq \Phi_\omega$ and $\Phi_\omega \not\vdash \bot$, we have $\phi \notin \Phi_\omega$. By Lemma 2.259, we get $[\![\phi]\!]^{\mathscr{I}}_{\varsigma} = 0$. ■

As a corollary we get Theorem 2.261:

Theorem 2.261 (Completeness). *If ϕ is valid in all interpretations, then ϕ is derivable.*

2.4.2 Case Study: Arithmetic in Permissive-Nominal Logic

Because term-formers in PNL can bind, we can axiomatise first-order logic. Thus assume a sort o whose terms reflect formulas of first-order logic. Then PNL quantification $\forall Z$ where $Z : o$ has the quality of an *axiom schema*, and we can use those terms to axiomatise arithmetic (a theory which in first-order logic famously involves an axiom schema).

So, we should be able to use PNL to give a finite, first-order axiomatisation of arithmetic. Writing down some plausible-looking axioms is one thing—proving they do what we expect them to do, is another. In this section, as a case study of an application of PNL, we do just that.

We assume one atomic sort ν and two base sorts ι and o.
We assume term-formers and proposition-formers as follows:

$$\begin{array}{llll} \dot{0} : \iota & \dot{\mathsf{succ}} : (\iota)\iota & \dot{+} : (\iota,\iota)\iota & \dot{*} : (\iota,\iota)\iota \\ \dot{\bot} : o & \dot{\Rightarrow} : (o,o)o & \dot{\forall} : ([\nu]o)o & \dot{\approx} : (\iota,\iota)o \\ \mathsf{var} : (\nu)\iota & \mathsf{sub}_\iota : ([\nu]\iota,\iota)\iota & \mathsf{sub}_o : ([\nu]o,\iota)o & \\ \approx_\iota : (\iota,\iota) & \approx_o : (o,o) & \varepsilon : (o) & \end{array}$$

Fig. 2.6 Signature $\dot{\mathcal{L}}$ suitable for a PNL specification of arithmetic

$$\begin{array}{lll} (\approx\mathbf{2}) & \forall X',X,Y',Y.(X'\approx X \wedge Y'\approx Y) \Rightarrow \big(& X' \dot{+} Y' \approx X \dot{+} Y \wedge \\ & & X' \dot{*} Y' \approx X \dot{*} Y \wedge \\ & & X' \dot{\Rightarrow} Y' \approx X \dot{\Rightarrow} Y \wedge \\ & & X' \dot{\approx} Y' \approx X \dot{\approx} Y \big) \\ (\approx\mathbf{1}) & \forall X',X.\ X'\approx X \ \Rightarrow & \dot{\mathsf{succ}}(X') \approx \dot{\mathsf{succ}}(X) \\ (\approx\mathbf{0}) & \forall X. & X \approx X \\ (\approx\dot{\forall}) & \forall Z',Z.\ Z'\approx Z \ \Rightarrow & \dot{\forall}([a]Z') \approx \dot{\forall}([a]Z) \\ (\approx\mathsf{sub}) & \forall X',X,Y',Y.(X'\approx X \wedge Y'\approx Y) \Rightarrow \big(& \mathsf{sub}_\iota([a]X',Y') \approx \mathsf{sub}_\iota([a]X,Y) \wedge \\ & & \mathsf{sub}_o([a]X',Y') \approx \mathsf{sub}_o([a]X,Y) \big) \\ (\approx o) & \forall Z',Z.\ Z'\approx Z \ \Rightarrow & (\varepsilon(Z') \Leftrightarrow \varepsilon(Z)) \\ (\approx\iota) & \forall X',X.\ X'\approx X \ \Rightarrow & \varepsilon(X' \dot{\approx} X) \end{array}$$

We fill in sorts as appropriate. Thus, $\dot{\bot} \approx_o \dot{\bot}$ whereas $0 \approx_\iota 0$, and so on. The permission sets of all unknowns are equal to $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Fig. 2.7 EQU: axioms for equality as a PNL theory

We proceed as follows, starting with the following PNL definitions:

- Figure 2.6 gives $\dot{\mathcal{L}}$ a signature for a shallow embedding of terms and formulas of first-order logic as PNL terms of sort ι and o respectively.
- Figure 2.7 gives equality axioms, as a transitive reflexive symmetric congruence for the term-formers in $\dot{\mathcal{L}}$.
- Figure 2.8 axiomatises substitution. We can have some confidence in this axiomatisation because it was already considered for nominal algebra in Gabbay and Mathijssen (2008a) and proven correct.
- Figure 2.9 gives axioms for first-order logic.
- Finally, Fig. 2.10 gives axioms for arithmetic. As discussed above, the induction axiom schema is captured using a universal quantification (the $\forall Z$ in (**PInd**)).

Sect. 2.4.2.4 briefly recalls the syntax and derivability relation of 'real' first-order logic. Then Sect. 2.4.2.5 maps this into the PNL theory we just constructed. Section 2.4.2.6 briefly recalls Peano arithmetic in the 'real' first-order logic.

$$
\begin{array}{lrl}
(\mathbf{sub}\mathsf{var}) & \forall X.\, \mathsf{var}(a)[a{\mapsto}X] & \approx X \\
(\mathbf{sub}\#) & \forall X,Z.\, Z[a{\mapsto}X] & \approx Z \\
(\mathbf{sub}\dot{\mathsf{succ}}) & \forall X',X.\, \dot{\mathsf{succ}}(X')[a{\mapsto}X] & \approx \dot{\mathsf{succ}}(X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{+}) & \forall X'',X',X.\, (X'' \mathbin{\dot{+}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{+}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{*}) & \forall X'',X',X.\, (X'' \mathbin{\dot{*}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{*}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{\Rightarrow}) & \forall X'',X',X.\, (X'' \mathbin{\dot{\Rightarrow}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{\Rightarrow}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{\approx}) & \forall X'',X',X.\, (X'' \mathbin{\dot{\approx}} X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X] \mathbin{\dot{\approx}} X'[a{\mapsto}X]) \\
(\mathbf{sub}\dot{\forall}) & \forall X,Z.\, (\dot{\forall}([b]Z))[a{\mapsto}X] & \approx \dot{\forall}([b](Z[a{\mapsto}X])) \\
(\mathbf{subid}) & \forall X.\, X[a{\mapsto}\mathsf{var}(a)] & \approx X
\end{array}
$$

$a \in \mathbb{A}^<$ and $b \notin \mathbb{A}^<$. The permission set of X'', X', and X is equal to $\mathbb{A}^<$. The permission set of Z is equal to $(b\ a){\cdot}\mathbb{A}^<$.

Fig. 2.8 SUB: axioms for substitution as a PNL theory

$$
\begin{array}{lrl}
(\dot{\Rightarrow}) & \forall Z',Z.\, \varepsilon(Z' \mathbin{\dot{\Rightarrow}} Z) & \Leftrightarrow (\varepsilon(Z') \Rightarrow \varepsilon(Z)) \\
(\dot{\forall}) & \forall Z.\, \big(\varepsilon(\dot{\forall}([a]Z)) & \Leftrightarrow \forall X.\varepsilon(Z[a{\mapsto}X])\big) \\
(\dot{\bot}) & \varepsilon(\dot{\bot}) & \Rightarrow \bot
\end{array}
$$

Here Z' and Z have sort o, permission set $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Fig. 2.9 FOL: axioms for first-order formulas as a PNL theory

$$
\begin{array}{lrl}
(\mathbf{PS0}) & \forall X. & \dot{\mathsf{succ}}(X) \approx \dot{0} \Rightarrow \bot \\
(\mathbf{PSS}) & \forall X',X. & \dot{\mathsf{succ}}(X') \approx \dot{\mathsf{succ}}(X) \Rightarrow X' \approx X \\
(\mathbf{P}{+}\mathbf{0}) & \forall X. & X \mathbin{\dot{+}} \dot{0} \approx X \\
(\mathbf{P}{+}\mathbf{succ}) & \forall X',X. & X' \mathbin{\dot{+}} \dot{\mathsf{succ}}(X) \approx \dot{\mathsf{succ}}(X') \mathbin{\dot{+}} X \\
(\mathbf{P}{*}\mathbf{0}) & \forall X. & X \mathbin{\dot{*}} \dot{0} \approx \dot{0} \\
(\mathbf{P}{*}\mathbf{succ}) & \forall X',X. & X' \mathbin{\dot{*}} \dot{\mathsf{succ}}(X) \approx (X' \mathbin{\dot{*}} X) \mathbin{\dot{+}} X \\
(\mathbf{PInd}) & \forall Z. \varepsilon(Z[a{\mapsto}\dot{0}]) \Rightarrow & \\
 & \quad \big(\forall X. (\varepsilon(Z[a{\mapsto}X]) \Rightarrow \varepsilon(Z[a{\mapsto}\dot{\mathsf{succ}}(X)]))\big) \Rightarrow & \\
 & \quad\quad \forall X. \varepsilon(Z[a{\mapsto}X]) &
\end{array}
$$

The permission set of X, X', and Z is $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Fig. 2.10 ARITH: axioms for arithmetic as a PNL theory

Finally, in Sect. 2.4.2.7 by arguments on models we show our main result of this section: Theorem 2.286. A formula is derivable in 'real' Peano arithmetic if and only if its translation in PNL is derivable in the PNL theory for arithmetic.

The permissive-nominal terms, PNL, permission-sets, and permissive-nominal sets semantics, all work together, and at the end of it all we really *can* embed a non-trivial theory with binding in PNL, and know it is correct.

2.4.2.1 The Signature $\dot{\mathcal{L}}$ and the Axioms

Definition 2.262. A signature $\dot{\mathcal{L}}$ suitable for writing out a PNL theory of first-order logic is given in Fig. 2.6.

Notation 2.263. We introduce the following syntactic sugar:

- We may write $\mathsf{sub}_o([a]r,t)$ as $r[a{\mapsto}t]$.
- We may write $\mathsf{sub}_\iota([a]r,t)$ as $r[a{\mapsto}t]$.
- We may write both $\approx_\iota$ and $\approx_o$ just as $\approx$.

Examples of this in use, follow immediately below.

2.4.2.2 The Axioms: Equality, Substitution, First-Order Logic, and Arithmetic

Equality

Axioms for equality $\approx\colon(\iota,\iota)$ and equality $\approx\colon(o,o)$ are given in Fig. 2.7.

Substitution

Axioms for substitution sub_ι and sub_o are given in Fig. 2.8.

We arguably abuse notation in Fig. 2.8 when we use unknowns of sort ι and o as appropriate not necessarily giving them distinct names (e.g. in (**sub**$*$) X has sort ι, whereas in (**sub**$\dot{\Rightarrow}$) we use another unknown also written X with sort o).

First-order logic

Axioms for (a shallow reflection of) first-order formulas as terms in PNL (the $\dot{\bot}$, $\dot{\Rightarrow}$, and $\dot{\forall}$) are given in Fig. 2.9.

Arithmetic

Given EQU, SUB, and FOL, it is not hard to write axioms for arithmetic in PNL. This is in Fig. 2.10. Later on in Theorem 2.286 we prove that this *is* an axiomatisation of arithmetic in PNL.

2.4.2.3 Comments on the Axioms

Remark 2.264. SUB is a PNL rendering of the nominal algebra theory naSUB from Example 2.170; the universal quantifiers which are implicit in the nominal algebraic judgement-form are made explicit here. This is essentially the same axiomatisation

as studied in Gabbay and Mathijssen (2006a, 2008a). Soundness and completeness are proved, so providing some formal sense in which the axioms of SUB are 'right'.

In Gabbay and Mathijssen (2008c) first-order logic is equationally axiomatised using nominal algebra (so the axioms involve only equality). Because PNL is already a first-order logic, we can use $\bot$, $\Rightarrow$, and $\forall$ directly to capture the behaviour of $\dot{\bot}$, $\dot{\Rightarrow}$, and $\dot{\forall}$. So note that FOL here is *not* the axiomatisation of Gabbay and Mathijssen (2008c); there we had to work a little harder because the ambient logic, nominal algebra, was purely equational.

Remark 2.265. Instead of the axioms for equality EQU, we could directly extend PNL by adding derivation rules Fig. 2.5 as follows:

$$\frac{\Phi, r \approx s, \phi[X{:=}r], \phi[X{:=}s] \vdash \Psi \qquad (fa(r) \cup fa(s) \subseteq supp(X))}{\Phi, r \approx s, \phi[X{:=}r] \vdash \Psi} (\approx\mathbf{S})$$

$$\frac{\Phi, r \approx r \vdash \Psi}{\Phi \vdash \Psi} (\approx\mathbf{R})$$

Remark 2.266. Every unknown has a sort, and a permission set.

Different choices of permission set may yield logically equivalent results. For example, in (**sub**lam) it is not vital that $supp(Z)$ is *exactly* $(b\ a)\cdot\mathbb{A}^<$. The important point is that $a \notin supp(Z)$.

Similarly, in (**sub**app) it is not vital that $supp(X'') = supp(X')$; when we use the axiom we can instantiate X'' and X' to r'' and r' such that $fa(r'') \neq fa(r')$, and conversely if we take $supp(X'') \neq supp(X')$ then we can still instantiate X'' and X' to r'' and r' such that $fa(r'') = fa(r') \subseteq supp(X'') \cap supp(X')$.

2.4.2.4 First-Order Logic $\mathcal{L}$

We will use the atoms $\mathbb{A}_\nu$ from $\dot{\mathcal{L}}$ in Sect. 2.4.2 as variables of our first-order logic (this is not necessary, but it is convenient). So for this section, $a, b, c, \ldots$ will range over distinct atoms in $\mathbb{A}_\nu$.

Definition 2.267. Define **terms** and **formulas** of $\mathcal{L}$ by:

$$t ::= a \mid 0 \mid succ(t) \mid t + t \mid t * t$$
$$\xi ::= t \approx t \mid \bot \mid \xi \Rightarrow \xi \mid \forall a.\xi$$

Substitution $t'[a{:=}t]$ and $\xi[a{:=}t]$ is as usual for first-order logic. We write sequents $\Xi \vdash \chi$ where Ξ and χ are sets of formulas. Derivability is as usual for first-order logic.

Definition 2.268. Define a mapping $(\text{-})^\bullet$ from terms and formulas of $\mathcal{L}$ to terms of $\dot{\mathcal{L}}$ (Sect. 2.4.2.1) by:

$$\begin{array}{ll} (a)^\bullet = a & (0)^\bullet = \dot{0} \\ (succ(t))^\bullet = \dot{succ}((t)^\bullet) & (t'+t)^\bullet = (t')^\bullet \dot{+} (t)^\bullet \\ (t' * t)^\bullet = (t')^\bullet \dot{*} (t)^\bullet & \\ (t' \approx t)^\bullet = (t')^\bullet \dot{\approx} (t)^\bullet & (\bot)^\bullet = \dot{\bot} \\ (\xi' \Rightarrow \xi)^\bullet = (\xi')^\bullet \dot{\Rightarrow} (\xi)^\bullet & (\forall a.\xi)^\bullet = \dot{\forall}[a](\xi)^\bullet \end{array}$$

Definition 2.269. Extend $(\text{-})^\bullet$ to first-order logic sequents $\Xi \vdash \chi$ as follows:

$$(\Xi \vdash \chi)^\bullet = \varepsilon(\dot{\forall}[a_1] \ldots \dot{\forall}[a_n]((\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l))^\bullet)$$

Here, $\Xi = \{\xi_1, \ldots, \xi_k\}$, $\chi = \{\chi_1, \ldots, \chi_l\}$, and the free variables of Ξ and χ are $\{a_1, \ldots, a_n\}$ (in some order).

Notation 2.270. Write S for $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$.

Lemma 2.271. $\mathsf{S} \vdash (t'[a{:=}t])^\bullet \approx (t')^\bullet[a{\mapsto}(t)^\bullet]$ *and* $\mathsf{S} \vdash (\xi[a{:=}t])^\bullet \approx (\xi)^\bullet[a{\mapsto}(t)^\bullet]$.

Proof. By routine inductions on t and ξ. ■

Theorem 2.272 (Correctness). *If* $\Xi \vdash \chi$ *is derivable in first-order logic then* $\mathsf{S} \vdash (\Xi \vdash \chi)^\bullet$ *is derivable in PNL.*

Proof. By a long but routine inspection we can check that EQU, SUB, and FOL allow us to model the behaviour of 'real' first-order logic. We use Lemma 2.271. ■

2.4.2.5 Interpretation of First-Order Logic

We recall the usual definition of interpretations in first-order logic:

Definition 2.273. A nominal **(first-order logic) interpretation** $\mathscr{M}$ is a **carrier set** M, and elements:

$$0^{\mathscr{M}} \in M, \quad succ^{\mathscr{M}} \in M \to M, \quad +^{\mathscr{M}} \in (M \times M) \to M, \text{ and } \quad *^{\mathscr{M}} \in (M \times M) \to M.$$

It is convenient to fix some $\mathscr{M}$ from here until Theorem 2.286.

Definition 2.274. Define $Valu_{\mathbb{A}_\nu}(M)$ by:

$$Valu_{\mathbb{A}_\nu}(M) = \{\varepsilon \in \mathbb{A}_\nu \to M \mid \exists A \subseteq \mathbb{A}_\nu . A \text{ finite } \wedge\ \forall a, b \notin A . \varepsilon(a) = \varepsilon(b)\}$$

Call elements of $Valu_{\mathbb{A}_\nu}(M)$ **$\mathbb{A}_\nu$-valuations** (to M). ε will range over $\mathbb{A}_\nu$-valuations.

If $x \in M$ write $\varepsilon[a{:=}x]$ for the valuation mapping b to $\varepsilon(b)$ and mapping a to x:

$$\varepsilon[a{:=}x](a) = x$$
$$\varepsilon[a{:=}x](b) = \varepsilon(b)$$

Give $\varepsilon \in Valu_{\mathbb{A}_\nu}(M)$ and $X \subseteq Valu_{\mathbb{A}_\nu}(M)$ a **pointwise** permutation action:

$$(\pi{\cdot}\varepsilon)(a) = \varepsilon(\pi^{-1}(a)).$$
$$\pi{\cdot}X = \{\pi{\cdot}\varepsilon \mid \varepsilon \in X\}.$$

U, V will range over *finitely-supported* subsets of $Valu_{\mathbb{A}_\nu}(M)$—so there exists some finite $A \subseteq \mathbb{A}_\nu$ such that for all π, if $\pi(a) = a$ for all $a \in A$ then $\pi{\cdot}U = U$.

Remark 2.275. $Valu_{\mathbb{A}_\nu}(M)$ would normally just be called 'the set of valuations'. We are more specific since we separately also have valuations on unknowns X (Definition 2.178).

PNL atoms are serving as variable symbols of $\mathcal{L}$. To conveniently apply nominal techniques, it is useful to restrict to valuations that are finite in the sense given in Definition 2.274. In any case, any term or formula will only contain finitely many atoms.

Definition 2.276. We extend the interpretation to first-order logic syntax as follows:

$$\begin{aligned}
[\![a]\!]^{\mathscr{M}}_{\varepsilon} &= \varepsilon(a)\\
[\![0]\!]^{\mathscr{M}}_{\varepsilon} &= 0^{\mathscr{M}}\\
[\![succ(t)]\!]^{\mathscr{M}}_{\varepsilon} &= succ^{\mathscr{M}}([\![t]\!]^{\mathscr{M}}_{\varepsilon})\\
[\![t'+t]\!]^{\mathscr{M}}_{\varepsilon} &= +^{\mathscr{M}}([\![t']\!]^{\mathscr{M}}_{\varepsilon}, [\![t]\!]^{\mathscr{M}}_{\varepsilon})\\
[\![t'*t]\!]^{\mathscr{M}}_{\varepsilon} &= *^{\mathscr{M}}([\![t']\!]^{\mathscr{M}}_{\varepsilon}, [\![t]\!]^{\mathscr{M}}_{\varepsilon})\\
[\![\bot]\!]^{\mathscr{M}}_{\varepsilon} &= 0\\
[\![\xi' \Rightarrow \xi]\!]^{\mathscr{M}}_{\varepsilon} &= max\{1-[\![\xi']\!]^{\mathscr{M}}_{\varepsilon}, [\![\xi]\!]^{\mathscr{M}}_{\varepsilon}\}\\
[\![\forall a.\xi]\!]^{\mathscr{M}}_{\varepsilon} &= min\{[\![\xi]\!]^{\mathscr{M}}_{\varepsilon[a:=x]} \mid x \in M\}\\
[\![t' \approx t]\!]^{\mathscr{M}}_{\varepsilon} &= 1 \text{ if } [\![t']\!]^{\mathscr{M}}_{\varepsilon} = [\![t]\!]^{\mathscr{M}}_{\varepsilon} \text{ and } 0 \text{ otherwise}
\end{aligned}$$

Definition 2.277. Call the formula ξ **valid** in $\mathscr{M}$ when $[\![\xi]\!]^{\mathscr{M}}_{\varepsilon} = 1$ for all ε.

Call $\xi_1, \ldots, \xi_k \vdash \chi_1, \ldots, \chi_l$ **valid** in $\mathscr{M}$ when $(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)$ is valid.

2.4.2.6 A Theory of Arithmetic in $\mathcal{L}$

Definition 2.278. Define a first-order theory of **arithmetic** by the axioms in Fig. 2.11.
An interpretation $\mathscr{M}$ is a **model** of arithmetic when $[\![\xi]\!]^{\mathscr{M}} = 1$ for ξ each of (**ps0**), (**pss**), (**p**+**0**), (**p**+**succ**), (**p**∗**0**), (**p**∗**succ**), and every instance of (**pind**).

$$
\begin{array}{ll}
(\mathbf{ps0}) & \forall a. \quad succ(a) \approx 0 \Rightarrow \bot \\
(\mathbf{pss}) & \forall a', a. \quad succ(a) \approx succ(a') \Rightarrow a \approx a' \\
(\mathbf{p{+}0}) & \forall a. \quad a{+}0 \approx a \\
(\mathbf{p{+}succ}) & \forall a', a.\ a'{+}succ(a) \approx succ(a'){+}a \\
(\mathbf{p{*}0}) & \forall a. \quad a{*}0 \approx 0 \\
(\mathbf{p{*}succ}) & \forall a', a.\ \ a'{*}succ(a) \approx (a'{*}a){+}a \\
(\mathbf{pind}) & \xi[a{:=}0] \Rightarrow \big(\forall a.(\xi \Rightarrow \xi[a{:=}succ(a)])\big) \Rightarrow \forall a.\xi \\
 & \text{(every } \xi\text{, every } a\text{)}
\end{array}
$$

Fig. 2.11 arithmetic: axioms for arithmetic in first-order logic

Remark 2.279. (**pind**) the induction axiom-scheme is of course of particular interest. We therefore unpack what its validity

$$[\![\xi[a{:=}0] \Rightarrow \forall a.(\xi \Rightarrow \xi[a{:=}succ(a)]) \Rightarrow \forall a.\xi]\!]^{\mathscr{M}} = 1 \qquad \text{(every } \xi\text{, every } a\text{)}$$

means, in a little more detail. For every a and ξ:

- If $[\![\xi[a{:=}0]]\!]^{\mathscr{M}}_{\varepsilon} = 1$, and
- if for every $x \in M$, $[\![\xi]\!]^{\mathscr{M}}_{\varepsilon[a:=x]} = 1$ implies that $[\![\xi[a{:=}succ(a)]]\!]^{\mathscr{M}}_{\varepsilon[a:=x]} = 1$,
- then for every $x \in M$, $[\![\xi]\!]^{\mathscr{M}}_{\varepsilon[a:=x]} = 1$.

In (**pind**) we take 'every a', and in (**PInd**) we do not. This is because in (**PInd**), a is α-convertible,

2.4.2.7 Building an Interpretation for $\dot{\mathcal{L}}$ from One for $\mathcal{L}$

Recall the PNL signature $\dot{\mathcal{L}}$ from Sect. 2.4.2.1. Suppose $\mathscr{M}$ is a model of arithmetic. We use it to build an interpretation $\mathscr{N}$ of $\dot{\mathcal{L}}$.

Definition 2.280. Extend $\mathcal{L}$ to $\mathcal{L}{+}M$ where we add all elements of M as constants, and extend the interpretation to interpret these constants as themselves in M. (So if $x \in M$ then x is a constant symbol in $\mathcal{L}{+}M$ and $[\![x]\!]^{\mathscr{M}}_{\varepsilon} = x$.)

Define an $\mathbb{A}_{\nu}$-valuation $\varepsilon_0 \in Valu_{\mathbb{A}_{\nu}}(M)$ by

$$\varepsilon_0(a) = 0^{\mathscr{M}} \quad \text{always.}$$

If t is a term, we write $[\![t]\!]^{\mathscr{M}}$ for the function $\lambda\varepsilon.[\![t]\!]^{\mathscr{M}}_{\varepsilon}$. If ξ is a formula, we write $[\![\xi]\!]^{\mathscr{M}}$ for the function $\lambda\varepsilon.[\![\xi]\!]^{\mathscr{M}}_{\varepsilon}$.

We now define an interpretation $\mathscr{N}$ for $\dot{\mathcal{L}}$. We give a denotation to the base sorts ι and o of $\dot{\mathcal{L}}$, as follows:

$$
\begin{array}{l}
\iota^{\mathscr{N}} = \{[\![t]\!]^{\mathscr{M}} \mid t \text{ a term of } \mathcal{L}{+}M\} \\
o^{\mathscr{N}} = \{[\![\xi]\!]^{\mathscr{M}} \mid \xi \text{ a formula of } \mathcal{L}{+}M\}
\end{array}
$$

We give a denotation to the term-formers and proposition-formers of $\dot{\mathcal{L}}$, as follows:

$$
\begin{array}{rlrl}
\mathsf{var}^{\mathcal{N}}\, a\, \varepsilon &= \varepsilon(a) & \mathsf{sub}_o^{\mathcal{N}}\,([a]u,v)\,\varepsilon &= u(\varepsilon[a{:=}v\varepsilon]) \\
\dot{0}^{\mathcal{N}}\,\varepsilon &= 0^{\mathcal{M}} & \dot{\Rightarrow}^{\mathcal{N}}\,(U,V)\,\varepsilon &= max\{1-U(\varepsilon),V(\varepsilon)\} \\
\dot{\mathsf{succ}}^{\mathcal{N}}\,u\,\varepsilon &= succ^{\mathcal{M}}(u\varepsilon) & \dot{\forall}^{\mathcal{N}}\,[a]U\,\varepsilon &= min\{U(\varepsilon[a{:=}x]) \mid x \in M\} \\
\dot{+}^{\mathcal{N}}\,(u,v)\,\varepsilon &= +^{\mathcal{M}}(u\varepsilon,v\varepsilon) & \dot{\approx}^{\mathcal{N}}\,(u,v)\,\varepsilon &= \approx^{\mathcal{M}}(u\varepsilon,v\varepsilon) \\
\dot{*}^{\mathcal{N}}\,(u,v)\,\varepsilon &= *^{\mathcal{M}}(u\varepsilon,v\varepsilon) & \approx_\iota^{\mathcal{N}}\,(u,v) &= 1 \text{ if } u{=}v \text{ and 0 otherwise} \\
\mathsf{sub}_\iota^{\mathcal{N}}\,([a]u,v)\,\varepsilon &= u(\varepsilon[a{:=}v\varepsilon]) & \approx_o^{\mathcal{N}}\,(U,V) &= 1 \text{ if } U{=}V \text{ and 0 otherwise} \\
\dot{\perp}^{\mathcal{N}}\,\varepsilon &= 0 & \varepsilon^{\mathcal{N}}\,U &= U(\varepsilon_0)
\end{array}
$$

Here, u and v range over $\iota^{\mathcal{N}}$ and U and V range over $o^{\mathcal{N}}$.

Lemma 2.281. *1.* $[\![t'[a{:=}t]]\!]^{\mathcal{M}}_\varepsilon = [\![t']\!]^{\mathcal{M}}_{\varepsilon[a:=[\![t]\!]^{\mathcal{M}}_\varepsilon]}$.

2. $[\![\xi[a{:=}t]]\!]^{\mathcal{M}}_\varepsilon = 1$ *if and only if* $[\![\xi]\!]^{\mathcal{M}}_{\varepsilon[a:=[\![t]\!]^{\mathcal{M}}_\varepsilon]} = 1$.

Lemma 2.282. *The following equalities all hold:*

$$
\begin{array}{rlrl}
\mathsf{var}^{\mathcal{N}}(a) &= [\![a]\!]^{\mathcal{M}} & \mathsf{sub}_\iota^{\mathcal{N}}([a][\![t']\!]^{\mathcal{M}},[\![t]\!]^{\mathcal{M}}) &= [\![t'[a{:=}t]]\!]^{\mathcal{M}} \\
\dot{0}^{\mathcal{N}} &= [\![0]\!]^{\mathcal{M}} & \mathsf{sub}_o^{\mathcal{N}}([a][\![\xi]\!]^{\mathcal{M}},[\![s]\!]^{\mathcal{M}}) &= [\![\xi[a{:=}s]]\!]^{\mathcal{M}} \\
\dot{\mathsf{succ}}^{\mathcal{N}}([\![t]\!]^{\mathcal{M}}) &= [\![succ(t)]\!]^{\mathcal{M}} & \dot{\perp}^{\mathcal{N}} &= [\![\perp]\!]^{\mathcal{M}} \\
\dot{+}^{\mathcal{N}}([\![t']\!]^{\mathcal{M}},[\![t]\!]^{\mathcal{M}}) &= [\![t'+t]\!]^{\mathcal{M}} & \dot{\Rightarrow}^{\mathcal{N}}([\![\xi']\!]^{\mathcal{M}},[\![\xi]\!]^{\mathcal{M}}) &= [\![\xi'\Rightarrow\xi]\!]^{\mathcal{M}} \\
\dot{*}^{\mathcal{N}}([\![t']\!]^{\mathcal{M}},[\![t]\!]^{\mathcal{M}}) &= [\![t'*t]\!]^{\mathcal{M}} & \dot{\forall}^{\mathcal{N}}([a][\![\xi]\!]^{\mathcal{M}}) &= [\![\forall a.\xi]\!]^{\mathcal{M}} \\
 & & \dot{\approx}^{\mathcal{N}}([\![r]\!]^{\mathcal{M}},[\![s]\!]^{\mathcal{M}}) &= [\![r\approx s]\!]^{\mathcal{M}}
\end{array}
$$

Proof. We compare Definitions 2.280 and 2.276. Most cases are immediate; we consider only the slightly less trivial ones:

$$
\begin{array}{rll}
\mathsf{var}^{\mathcal{N}}(a) &= (\lambda a.\lambda\varepsilon.\varepsilon(a))a & \text{Definition 2.280} \\
&= (\lambda a.[\![a]\!]^{\mathcal{M}})a & \text{Definition 2.276} \\
&= [\![a]\!]^{\mathcal{M}} & \text{fact} \\
\\
\mathsf{sub}_\iota^{\mathcal{N}}([a][\![t']\!]^{\mathcal{M}},[\![t]\!]^{\mathcal{M}}) &= \lambda\varepsilon.[\![t']\!]^{\mathcal{M}}(\varepsilon[a{:=}[\![t]\!]^{\mathcal{M}}\varepsilon]) & \text{Definition 2.280} \\
&= \lambda\varepsilon.[\![t'[a{:=}t]]\!]^{\mathcal{M}} & \text{Lemma 2.281}
\end{array}
$$

Other cases are no harder. ■

Lemma 2.283. *$\mathcal{N}$ (Definition 2.280) is a PNL interpretation.*

Proof. We must check that:

- *$\iota^{\mathcal{N}}$ and $o^{\mathcal{N}}$ are permissive-nominal sets.*
  By routine calculations. (In fact, $\iota^{\mathcal{N}}$ and $o^{\mathcal{N}}$ are *nominal* sets; that is, their elements all have finite support.)

- *The functions defined in Definition 2.280 map elements of $\iota^{\mathscr{N}}$, $o^{\mathscr{N}}$, $[\mathbb{A}]\iota^{\mathscr{N}}$, and $[\mathbb{A}]o^{\mathscr{N}}$ correctly to the appropriate sets.*
  By Lemma 2.282.
- *$\varepsilon^{\mathscr{N}}$ is equivariant from $o^{\mathscr{N}}$ to $\{0,1\}$.*
  By routine calculations using the fact that $(a\ b)\cdot\varepsilon_0 = \varepsilon_0$.

■

Lemma 2.284. *If $(\Xi \vdash \chi)^\bullet$ is valid in $\mathscr{N}$, then $\Xi \vdash \chi$ is valid in $\mathscr{M}$.*

Proof. We calculate that if $(\Xi \vdash \chi)^\bullet$ is valid in $\mathscr{N}$, then

$$[\![(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)]\!]^{\mathscr{M}}_{\varepsilon_0} = 1$$

But the proposition written out above is closed, so for all valuations ε, $[\![(\xi_1 \wedge \ldots \wedge \xi_k) \Rightarrow (\chi_1 \vee \ldots \vee \chi_l)]\!]^{\mathscr{M}}_{\varepsilon} = 1$. ■

Recall from Notation 2.270 that we write S for $\mathsf{EQU} \cup \mathsf{SUB} \cup \mathsf{FOL}$.

Proposition 2.285. *The axioms of $\mathsf{S} \cup \mathsf{ARITH}$ are valid in $\mathscr{N}$.*

Proof. By a routine verification. We consider the axiom $(\dot{\forall})$ from Fig. 2.9. We unpack definitions and see that we must prove that for every ξ in $\mathcal{L}{+}M$,

- $\forall x{\in}M.\varepsilon_0[a{:=}x] \in (\xi)^\bullet$ if and only if
- $\varepsilon_0[a{:=}(t)^\bullet] \in (\xi)^\bullet$ for every t a term of $\mathcal{L}{+}M$.

This follows, because $\mathcal{L}{+}M$ has a constant symbol for every $x \in M$. Validity of the other axioms is no harder. ■

Theorem 2.286. $\mathsf{arithmetic}, \Xi \vdash \chi$ *in first-order logic if and only if* $\mathsf{S} \cup \mathsf{ARITH} \vdash (\Xi \vdash \chi)^\bullet$ *in PNL.*

Proof. We prove two implications. The top-to-bottom implication follows using Theorem 2.272.

For the bottom-to-top implication, we reason as follows: Suppose $\mathsf{S} \cup \mathsf{ARITH} \vdash (\Xi \vdash \chi)^\bullet$ in PNL. Choose an arbitrary interpretation $\mathscr{M}$ of first-order logic that is a model of arithmetic, with carrier set M. By Soundness (Theorem 2.245) and Proposition 2.285, $(\Xi \vdash \chi)^\bullet$ is valid in $\mathscr{N}$. By Lemma 2.284 $\Xi \vdash \chi$ is valid in $\mathscr{M}$. $\mathscr{M}$ was arbitrary, so by completeness of first-order logic (Shoenfield 1967, §4.2) it follows that $\Xi \vdash \chi$ is derivable. ■

2.4.3 Further Properties of PNL

2.4.3.1 More PNL Theories

We briefly mention on how to express some familiar 'nominal' constructs in PNL.

Inductive types

Permissive-nominal logic can express the principles of nominal abstract syntax developed in Gabbay and Pitts (2001).

Suppose a base sort ι, a name sort ν, and term-formers

$$\mathsf{var} : (\nu)\iota, \quad \mathsf{app} : (\iota,\iota)\iota, \quad \text{and} \quad \mathsf{lam} : ([\nu]\iota)\iota.$$

Fix an unknown $U : \iota$ and for brevity write $\phi[U{:=}r]$ as $\phi(r)$ for every ϕ. Suppose an axiom-scheme, for every ϕ:

$$\begin{array}{l}\phi(\mathsf{var}(a)) \Rightarrow \\ \forall X.(\phi(X) \Rightarrow \phi(\mathsf{lam}([a]X))) \Rightarrow \\ \forall X,Y.(\phi(X) \Rightarrow \phi(Y) \Rightarrow \mathsf{app}(X,Y)) \Rightarrow \\ \qquad\qquad \forall X.(\phi(X))\end{array}$$

Here X and Y have sort ι and we make a fixed but arbitrary choice of atom $a \in \mathit{supp}(X)$.

We can also express this finitely, if we axiomatise a sort for predicates (as we did for arithmetic). Here is the axiom-scheme above made finite by using the theories EQU, SUB, and FOL from Sect. 2.4.2:

$$\begin{array}{l}\forall Z.\varepsilon(Z[a{\mapsto}\mathsf{var}(a)]) \Rightarrow \\ \quad \forall X.(\varepsilon(Z[a{\mapsto}X]) \Rightarrow \varepsilon(Z[a{\mapsto}\mathsf{lam}([a]X))) \Rightarrow \\ \quad \forall X,Y.(\varepsilon(Z[a{\mapsto}X]) \Rightarrow \varepsilon(Z[a{\mapsto}Y]) \Rightarrow \varepsilon(Z[a{\mapsto}\mathsf{app}(X,Y)])) \Rightarrow \\ \qquad\qquad \forall X.\varepsilon(Z[a{\mapsto}X])\end{array}$$

The И quantifier

Nominal sets support the И-quantifier Gabbay and Pitts (2001). PNL also includes the И-quantifier; the way in which it does this is quite interesting, as we shall see in a moment.

$И$ has some distinctive properties which are reflected in nominal logic (NL) and the logic of FM sets (FM):

$$\frac{\forall x.(\mathsf{P}(x) \Rightarrow И a.\mathsf{Q}(a,x))}{\forall x.И a.(\mathsf{P}(x) \Rightarrow \mathsf{Q}(a,x))} \qquad \frac{\forall x.И a.И b.(b\,a)\cdot x \approx x}{И a.И b.\forall x.(a\#x \Rightarrow b\#x \Rightarrow (b\,a)\cdot x \approx x)}$$

Here and below we write a double horizontal line for 'is provably equivalent to'. $И$ appears absent from Permissive-Nominal Logic (PNL). It is 'hiding' in the permission sets. Corresponding propositions are, where $a,b \notin supp(X)$

$$\frac{\forall X.(\mathsf{P}(X) \Rightarrow \mathsf{Q}(a,X))}{\forall X.(\mathsf{P}(X) \Rightarrow \mathsf{Q}(a,X))} \qquad \frac{\forall X.(b\,a)\cdot X \approx X}{\forall X.(b\,a)\cdot X \approx X}$$

We see from these examples that two things are happening: first, freshness conditions are hard-coded into the syntax by permission sets—and second, so is the $И$-quantifier.

It is interesting to consider another example. In NL/FM:

$$\frac{И a.\mathsf{P}(a) \wedge И a.\mathsf{Q}(b)}{И a.И b.(\mathsf{P}(a) \wedge \mathsf{Q}(b))} \qquad \frac{И a.\mathsf{P}(a) \wedge И a.\mathsf{Q}(b)}{И a.(\mathsf{P}(a) \wedge \mathsf{Q}(a))}$$

Correspondingly in PNL we have:

$$\frac{\mathsf{P}(a) \wedge \mathsf{Q}(b)}{\mathsf{P}(a) \wedge \mathsf{Q}(b)} \qquad \frac{\mathsf{P}(a) \wedge \mathsf{Q}(b)}{\mathsf{P}(a) \wedge \mathsf{Q}(a)}$$

It is easy to use the rule (**Ax**) from Fig. 2.5 to construct a derivation proving that $\mathsf{P}(a) \wedge \mathsf{Q}(b)$ and $\mathsf{P}(a) \wedge \mathsf{Q}(a)$ are indeed logically equivalent in Permissive-Nominal Logic.

The π in (**Ax**) expresses that truth is preserved by permutative renaming, or in symbols: $\vdash \phi \Leftrightarrow \pi\cdot\phi$ always.

A permission set S can be viewed in two ways: as giving permission to instantiate using free atoms in S—but also as a form of $И$ for the atoms not in S.

Semantic freshness

To express in permissive-nominal algebra that a is fresh for the denotation of s it suffices to assert $(b\ a)\cdot s = s$ where $b \notin supp(s)$. Thus the theory of a semantic freshness predicate Fresh has one axiom $\mathsf{Fresh}(a,X) \Leftrightarrow (b\ a)\cdot X = X$ where $a \in supp(X)$ and $b \notin supp(X)$ (and we fill in sorts as appropriate). In PNL with equality, the axiom is $\forall X.\ \mathsf{Fresh}(a,X) \Leftrightarrow (b\ a)\cdot X = X$.

Atoms-abstraction

Atoms-abstraction can also be expressed as a theory in permissive-nominal algebra. For a base sort τ and name sort ν assume a base sort $[\nu]\tau$ and a term-former

$\mathsf{abs} : (\nu, \tau)([\nu]\tau)$, along with a single axiom $\mathsf{abs}(a, X) = \mathsf{abs}(b, (b\ a)\cdot X)$ where $a \in supp(X)$ and $b \notin supp(X)$. In PNL with equality, the axiom is $\forall X.\mathsf{abs}(a, X) = \mathsf{abs}(b, (b\ a)\cdot X)$.

2.4.3.2 Admissibility of Cut

We indicate how (**Cut**) is admissible in the presence of the other rules in Fig. 2.5.

Definition 2.287. Suppose $fa(r) \subseteq supp(X)$ and $r : sort(X)$. Define $\Phi[X{:=}r]$ by

$$\Phi[X{:=}r] = \{\phi[X{:=}r] \mid \phi \in \Phi\}.$$

Lemma 2.288 is proved by routine arguments like those in Dowek et al. (2010), Urban et al. (2004):

Lemma 2.288. *Suppose $Y \notin fV(t)$. Then*

$$r[Y{:=}u][X{:=}t] = r[X{:=}t][Y{:=}u[X{:=}t]].$$

Lemma 2.289. *Suppose $fa(r) \subseteq supp(X)$ and $r : sort(X)$. Then*

$$\Phi \vdash \Psi \quad \textit{implies} \quad \Phi[X{:=}r] \vdash \Psi[X{:=}r].$$

Proof. By a routine induction on derivations. The case of (**Ax**) uses Lemmas 2.71 and 2.288. The case of ($\forall$**L**) uses Lemma 2.288. ■

Lemma 2.290.

1. *If there exists a derivation Δ of $\Phi \vdash \psi, \Psi$ then there exists a derivation of $\Phi \vdash \pi\cdot\psi, \Psi$.*
2. *If there exists a derivation Δ of $\Phi, \phi \vdash \Psi$ then there exists a derivation of $\Phi, \pi\cdot\phi \vdash \Psi$.*

Proof. By a simultaneous induction on Δ. The case of ($\forall$**L**) uses Lemma 2.71. (We need the *simultaneous* induction for ($\Rightarrow$**L**) and ($\Rightarrow$**R**), since parts of the proposition move between left and right.) ■

Notation 2.291. An instance of (**Cut**) rests on two sub-derivations. It is convenient to call them the **left branch** and **right branch** as illustrated:

$$\frac{\begin{array}{cc}\vdots\ \textit{Left branch} & \vdots\ \textit{Right branch} \\ \Phi,\ \phi \vdash \Psi & \Phi \vdash \phi, \Psi\end{array}}{\Phi \vdash \Psi}\ (\mathbf{Cut})$$

Theorem 2.292 (Cut-elimination). *If* $\Phi \vdash \Psi$ *is derivable with a derivation that uses* (**Cut**)*, then it is derivable with a derivation that does not use* (**Cut**).

Proof. The proof is as for first-order logic. The only differences are a π in (**Ax**) and a side-condition $fa(r) \subseteq supp(X)$ in ($\forall$**L**). These affect terms and have no effect on the structure of derivations; for the purposes of this proof they are irrelevant.

We commute instances of (**Cut**) upwards, as usual, following the method of (Dummett 1977, pages 139–145) or Gabbay (2011a). At each step, the following measure based on the depth of subderivations and the size of the cut formula, decreases:

- The size of the cut formula, and
- the longest path up the derivation the cut, that the formula persists,

lexicographically ordered.

- The commutation cases between rules for $\Rightarrow$ and $\forall$ are as standard for first-order logic.
- The essential case for $\Rightarrow$ is as standard.
- For the essential case for $\forall$, suppose the subderivation has the following form:

$$\dfrac{\dfrac{\Phi,\ \phi[X{:=}r] \vdash \Psi}{\Phi,\ \forall X.\phi \vdash \Psi}\ (\forall\mathbf{L}) \qquad \dfrac{\begin{array}{c}\vdots\ \Delta\\ \Phi \vdash \phi,\ \Psi\end{array}}{\Phi \vdash \forall X.\phi,\ \Psi}\ (\forall\mathbf{R})}{\Phi \vdash \Psi}\ (\mathbf{Cut})$$

  By Lemma 2.289 there is a derivation $\Delta[X{:=}r]$ of $\Phi \vdash \phi[X{:=}r],\ \Psi$. We eliminate the essential case as follows:

$$\dfrac{\Phi,\ \phi[X{:=}r] \vdash \Psi \qquad \begin{array}{c}\vdots\ \Delta[X{:=}r]\\ \Phi \vdash \phi[X{:=}r],\ \Psi\end{array}}{\Phi \vdash \Psi}\ (\mathbf{Cut})$$

- Suppose the subderivation has the following form:

$$\dfrac{\dfrac{}{\Phi,\ \phi \vdash \pi{\cdot}\phi, \Psi}\ (\mathbf{Ax}) \qquad \begin{array}{c}\vdots\ \Delta\\ \Phi,\ \pi{\cdot}\phi \vdash \Psi\end{array}}{\Phi,\ \phi \vdash \Psi}\ (\mathbf{Cut})$$

  We use Lemma 2.290 to obtain a derivation Δ' of $\Phi,\ \phi \vdash \Psi$ (the transformations involved in the proof of Lemma 2.290 do not increase the inductive measure).

■

2.4.3.3 Exhausting the Available Atoms

We conclude with a brief discussion on a subtle point in the PNL design. Suppose a name sort ν, a base sort τ, and a proposition former $\# : (\nu, \tau)$. Suppose an atom a and an unknown $X : \tau$ with $supp(X) = \mathbb{A}^<$. Suppose an unknown $Y : \nu$ with $supp(Y) = \mathbb{A}^<$. Consider an interpretation in which $\#(a,X)$ is interpreted as $a \notin supp(\varsigma(X))$ and τ is interpreted as $\mathbb{L}$ (Definition 2.26).

That is, # is interpreted as freshness and τ is interpreted as well-orderings of permission-sets.

In the PNL of this paper, the interpretation of the proposition $\phi = \forall X.\exists Y.\#(Y,X)$ is false: we take $\varsigma(X)$ to well-order $\mathbb{A}^<$ and there is no $a \in supp(Y)$ such that $a \notin supp(\varsigma(X))$.

Suppose we decide that we want a version of PNL in which ϕ is true. In this case, we can consider denotations such that every element has support of the form $\pi \cdot \mathbb{A}^{\ll}$ where $\mathbb{A}^{\ll}$ is infinite and $\mathbb{A}^{\ll} \subseteq \mathbb{A}^<$ and $\mathbb{A}^< \setminus \mathbb{A}^{\ll}$ is also infinite. In this way, an unknown X cannot 'exhaust' $\mathbb{A}^<$.

The lesson we draw from this small example is that nominal semantics offer a host of interesting and inspiring design options. In this paper, we have cut one path through this design space which is expressive enough to get the results we want. Other paths are possible.

2.4.4 Conclusions

This paper reflects a research arc by the author in collaboration with others, roughly from 2005 to 2012. Thanks to improvements in presentation and the use of permissive-nominal techniques, definitions and proofs are simpler than in previous literature, and new properties emerge.

We have constructed permissive-nominal sets. We gave a nominal syntax for them and explored their computational properties in nominal unification and rewriting. We considered nominal algebra and proved soundness, completeness, and HSP over permissive-nominal sets. We gave nominal terms a $\forall$-quantifier over unknowns and used this to build a first-order logic *permissive-nominal logic*. Finally, in an extended case study we gave finite axiomatisations of first-order logic and arithmetic and proved correctness.

Mathematical foundations influence language, and (famously) language influences thought. Nominal sets are a foundation with a model of names which is different from what has been considered before, so the question is: what new languages, and new thoughts, can emerge? This chapter attempts to address that question by illustrating the broad sweeps of what a 'nominal' meta-mathematics might look like.

We are not and cannot be encyclopaedic or exhaustive. For other work we should mention αProlog, which allows Horn clauses Cheney and Urban (2008) (this preceded PNL, and could be viewed as a subset of it). The author in collaboration has proved correctness for several non-trivial theories in nominal syntaxes, including equational treatments of substitution Gabbay and Mathijssen (2006a, 2008a), λ-calculus Gabbay and Mathijssen (2008b, 2010), and first-order logic Gabbay and Mathijssen (2006c, 2008c), as well as the finite first-order nominal axiomatisation of arithmetic Dowek and Gabbay (2010, 2012a) which we considered in Sect. 2.4.2. There are translations from nominal terms to λ-terms by Levy and Villaret and by Dowek, the author, and Mulligan Levy and Villaret (2008), Dowek et al. (2010), including a translation of algebraic reasoning (so, not just unification) Gabbay and Mulligan (2009); and there is a translation of permissive-nominal logic to higher-order logic in Dowek and Gabbay (2012b) which illustrates the differences and similarities of the two logics, and exploits some unusual model-theoretic ideas.

We also mention a translation of nominal terms to many-sorted first-order syntax by Kurz and Petrişan (2010), and a categorical treatment of nominal Lawvere theories in Clouston (2009). It may also prove useful to consider nominal languages over nominal structures other than sets, for instance over nominal domains Turner (2009).

See also the 'atlas of nominal languages' in Appendix A.

This research is developing a topic which this author believes could become an immense field; the informal meta-level having been relatively unformalised until now for want of a denotation with names, which is what nominal sets provide.

It is important to realise that this story is not just about nominal sets, nor is it just about semantics; there is also the issue of finding appropriate syntaxes for our semantics. The logic of FM sets, nominal logic, and the Nominal Isabelle package Gabbay and Pitts (2001), Pitts (2003), Urban (2008) are all first-order axiomatisations of nominal sets.[19] In all these cases, the syntax is that of 'ordinary' first- or higher-logic.[20] These are denotations for syntax-with-binding.

Nominal terms and permissive-nominal terms, and the syntaxes based on them such as nominal rewriting, algebra, and permissive-nominal logic, do not follow automatically from nominal sets. They are syntaxes for meta-mathematics of independent interest. Thus, this chapter has surveyed the author's attempts, via methods which are both syntactic and also semantic, to outline what meta-mathematics could

[19]Essentially, Gabbay and Pitts (2001) is the first third of the author's thesis; Pitts (2003) is the same but minus the cumulative sets hierarchy; and Urban (2008) is an extensive implementation in higher-order logic, with a library of powerful macros. One reason this is non-trivial has to do with automatically deriving the *equivariance* properties described e.g. in (Gabbay, 2011b, Section 4.2).

[20]Sometimes, authors write 'nominal logic' for that logic obtained by adding for each atom a constant symbol to the syntax of first-order logic, and adding infinitely many axioms reflecting nominal sets (equalities of swapping atoms, fresh atoms, and so on). This is nominal sets wearing a 'syntactic disguise': consider by analogy a theory of arithmetic with a constant symbol for each number and an axiom for every arithmetic equality.

look like if it were based on nominal foundations. The fact that—for instance—we were able to finitely axiomatise arithmetic in the nominal first-order that is PNL in Sect. 2.4.2.2, is one demonstration that this meta-mathematics is a new and different place from what the reader may be used to.

In a sense this paper is a sequel to the survey of Gabbay (2011b) (written in 2008 and submitted in early 2009). But whereas Gabbay (2011b) concentrated on applications of nominal sets to syntax with binding, this paper considers nominal sets as a basis for meta-mathematics. Hints of this appeared in nominal rewriting Fernández et al. (2004), Fernández and Gabbay (2007), which allowed arbitrary oriented equality theories over nominal terms. Perhaps unwisely, we shall succumb to a wordplay: Gabbay (2011b), Gabbay and Pitts (2001) explore *denotation of specification with binding*; whereas here we explore *specification of denotation with binding*.

Thus, in this document we have explored the consequences of taking FM-sets style names seriously in meta-mathematics. But even that does not exhaust the potential applications of nominal techniques. Mathematics and computer science are evolving in ways which increase the importance of names, and nominal techniques have arisen from this; we can expect that evolution to continue.

This motivates us to revisit certain foundational design decisions; whether to admit atoms—to sound more mathematical, we say *urelemente* and to sound less mathematical, we say *names*—and what properties these should have. Linguists might well call these *referents*, and have been studying them for a long time.

Whatever we call them, they exist and we use them all the time. So we will conclude with two slogans:

- *Names are data.*
- *Names with additional properties are ubiquitous.*

This chapter has studied formal languages with which to specify some of the possible additional properties of names, such as 'having a substitution action' or 'being universally quantifiable'. But more generally, by this combination of a new point of view and a rigorous mathematics, nominal techniques have the potential to simplify, factor out common properties, and help control some of a modern mathematics of logic and computation.

Names are not just a technical issue, to be ignored or circumvented with 'tricks'. Names are a philosophical, foundational, linguistic, and computational issue. The mathematics of names is the mathematics of mathematics.

Dov Gabbay wrote in his preface to the second edition that

> the researcher … is having more and more in common with the traditional philosopher who has been analysing such questions for centuries (unrestricted by the capabilities of any hardware). … I believe the day is not far away in the future when the computer scientist will wake up one morning with the realisation that he is actually a kind of formal philosopher!

We would add "and philosophers, linguists—and some artists too—may wake up one morning with the realisation that they are actually a kind of abstract computer scientist". Amen.

Appendix

A An Atlas of Nominal Languages

The reader coming to the nominal literature could be forgiven for finding it perplexing. What are 'Fraenkel-mostowski sets', 'nominal sets', 'nominal terms', 'nominal logic', 'nominal rewriting and algebra', 'αProlog', 'nominal equational logic', 'permissive-nominal algebra', 'permissive-nominal logic' (with/without *shift*-permutations)? In this Appendix we will give a brief annotated bibliography covering, loosely, the relevant publications. This list is not meant to be exhaustive.

Traditionally, nominal sets are understood as a tool for the mathematical analysis of syntax, as described for instance in the author's previous survey/research paper Gabbay (2011b), or in slides of an excellent course of lectures by Pitts (2011). This author takes a view of nominal sets not just as a foundation for syntax with binding, but as a foundation for mathematics itself—names and binding, after all, appear everywhere. The atlas below surveys relevant publications.

For each item in the list below, we reference where the idea was introduced to the 'nominal' literature, and any other relevant conference and journal papers.

A.1 FM Set Theory (Gabbay and Pitts 1999, 2001)

Fraenkel-Mostowski set theory (FM) and nominal sets (called 'equivariant FM sets' in that paper) are the foundational semantics for nominal techniques.

Fraenkel-Mostowski sets were already known and had been used for other purposes; see (Gabbay, 2011b, Remark 2.22) for more detailed historical comments. Nominal sets were familiar as e.g. the Schanuel topos. So both semantics were known.

What was new to Gabbay and Pitts (2001) was the observation by the author and Pitts of the notions of support, atoms-abstraction, the self-dual behaviour of the $\mathcal{N}$ quantifier, and the application to what is now called *nominal abstract syntax*.[21]

[21] At the same time, Fiore Plotkin and Turi developed an approach to abstract syntax which was really exactly the same thing Fiore et al. (1999). The key difference turned out to be that nominal sets admit a relatively elementary sets-based interpretation of the presheaves. As argued in Gabbay and Hofmann (2008) there are 'fewer presheaves' in the nominal semantics, we feel that an elementary presentation of the mathematics—where this is possible—is a powerful advantage not just for the reader but also for the practicing theorist.

Fiore has continued this line of research in collaboration and produced logics which in some sense which has never been made formal, parallel the development here. For an example of this see Fiore and Hur (2010).

A.2 Nominal Logic (Pitts 2001, 2003)

The constructions of Gabbay and Pitts (2001) are repeated, but in a first-order axiomatisation of nominal sets rather than one of the FM cumulative hierarchy. Pitts also coined the catchy label 'nominal'.

Sometimes authors identify the nominal logic of Pitts (2003) with nominal techniques in general. This is limiting, and it gets the mathematical development the wrong way round. Nominal logic is a Hilbert-style axiomatisation in first-order logic. These axioms have meaning because of the underlying nominal sets models, and not the other way around; nor does the axiomatisation *per se* contribute to new syntax or proof-theory with which to study names.

In order to make progress, we needed new syntax that more explicitly represents atoms and their properties.

Thus for instance the *nominal logic programming* developed by Cheney and Urban (2008) (also referenced below) is called logic-programming in nominal logic, but we also see from Figures 6, 7, and 8 of Cheney and Urban (2008) that the syntax and axioms used are a variant of nominal terms.

A.3 Proof-Theories for the И-quantifier (Gabbay 2007a; Gabbay and Cheney 2004; Cheney 2005b)

Some attempts have been made to give the distinctive И-quantifier of nominal techniques, a proof-theory. In arguably increasing order of elegance these are Gabbay (2007a) (this was received by the journal in 2003 but took 4 years to get printed), Gabbay and Cheney (2004) (written with Cheney to develop on Gabbay (2007a)), and Cheney (2005b).

The permissive-nominal logic (PNL) of this survey is another item on that list, and perhaps it is one of the nicest; certainly the PNL treatment of И is very different from what has come before, see Sect. 2.4.3.1.

Complete semantics for this family of logics are in Gabbay (2007a), Cheney (2006), and in Dowek and Gabbay (2012a). See also Sect. 2.4.1.4 of this survey.

A.4 Nominal Terms (Urban et al. 2003, 2004)

This new syntax introduced the distinctive freshness side-conditions and the nominal terms syntax, with its separation of atoms a and unknowns X into two syntactic classes. Urban et al. (2004) is where the syntactic ideas of this survey were born, if not the specific 'permissive' implementation, which came later (*permissive-nominal terms* below).

There is now quite a substantial body of work devoted to computing efficiently on nominal terms; notably Calvès (2010), Levy and Villaret (2010). There is also a body of work devoted to translating between nominal terms and *higher-order patterns* Miller et al. (1989). We are far from exhaustive, but good places to start reading are Cheney (2005a), Levy and Villaret (2008, 2012), Gabbay and Mulligan (2009), and Dowek et al. (2010).

A.5 Nominal Rewriting (Fernández et al. 2004; Fernández and Gabbay 2007) and αProlog (Cheney and Urban 2003, 2008)

These were the first logical languages using nominal terms as a general-purpose assertion language; nominal rewriting was designed explicitly to allow us to assert (directed) equalities between terms such as β or η-equivalence. αProlog was intended by its designers for reasoning on nominal abstract syntax, and explicitly presented as such—but in retrospect it can also be viewed as a general-purpose 'nominal' reasoning system in the same family as nominal rewriting and later work.[22]

A.6 Nominal Algebra (Gabbay 2005; Gabbay and Mathijssen 2006a, 2007, 2009)

Nominal algebra is simply the undirected version of nominal rewriting.[23] What makes nominal algebra interesting above and beyond nominal rewriting is the different theorems we prove about equality instead of rewriting; for instance the HSPA theorem of Gabbay (2009) (much simplified here in Sect. 2.3.5), and various correctness results for axiomatisations of e.g. substitution, λ-calculus, and first-order logic Gabbay and Mathijssen (2006a,c, 2008a,b,c, 2010).

The paper Gabbay and Mathijssen (2006a) is where the *permutative convention* of Definition 2.2 was introduced, used by the author consistently since then. This comes from the author's work formalising nominal reasoning in Isabelle in Gabbay (2001) and spares us from having to explicitly enumerate all inequalities between

[22] James Cheney, private communication.

[23] Actually, this is a simplification. There is a significant difference, which is described in Fernández and Gabbay (2010): nominal rewriting does not have an explicit rule to generate fresh atoms, whereas nominal algebra does. To the level of detail we wish to go into here, this does not matter. The permissive-nominal syntax of this survey makes the issue obsolete because fresh atoms are a structural fact of the permission sets.

atoms. Thus, if pressed to be entirely formal, 'a and b' refers to two meta-variables ranging over *distinct* atoms.

Kurz and Petrişan proved an HSP theorem for nominal algebra by treating nominal algebra as a kind of many-sorted first-order logic Kurz and Petrişan (2010)—the sorts are finite sets of atoms and come from the categorical view of nominal sets as presheaves. The effect of nominal theories can thus be attained in many-sorted first-order syntax. That syntax is just standard first-order syntax is potentially a big advantage, for instance if one wants to transfer results directly from universal algebra. This offers alternative and effective methods of semantic proof; e.g. Kurz and Petrişan (2010) significantly simplifies the proofs of Gabbay (2009). We pay for this convenience with infinities; e.g. even the simplest theory is infinite since equalities are replicated at every sort. Of course, the theory may still be finitely presentable. Section 2.3.5 of the current paper contains another, further simplified, HSP proof.

A.7 Nominal Equational Logic (Clouston and Pitts 2007; Clouston 2011)

Call the judgement 'a is fresh for the syntax s' **syntactic freshness** and 'a is fresh for the denotation of s' **semantic freshness**. *Nominal Equational Logic* (**NEL**) closely resembles NA, but whereas both have a semantic equality judgement ($s = t$), NEL adds a semantic freshness judgement.

In Clouston and Pitts (2007) Clouston and Pitts claimed that NEL was significantly more complete than NA because of this, but they had missed that semantic freshness is expressible using equality and syntactic freshness (see for instance (Gabbay and Mathijssen 2007, Theorem 5.5) and (Gabbay and Mathijssen 2009, Lemma 4.51)).[24]

Note that two distinct logics have been called NEL: one in Clouston and Pitts (2007), and one in Clouston (2011) which restricts semantic freshness to the left of the turnstile; compare Figure 5 of Clouston and Pitts (2007) with Figure 1 of Clouston (2011). Both have syntactic freshness: see the side-condition $a\#(\bar{a},t,t')$ in the ATM-INTRO and ATM-ELIM rules of Figure 5, and similar side-conditions in Figure 1. Thus, when Clouston writes in Clouston (2011) that "*[syntactic]* freshness in NA is sound, but not complete, for freshness in the underlying nominal sets interpretation *[semantic freshness]*", echoing similar comments in Clouston and Pitts (2007), this omits mention that NEL also has a syntactic freshness.

It is in any case a red herring. If we can choose fresh atoms and compare elements for semantic equality, then semantic freshness makes the logic 'do equality twice' and just adds complexity Gabbay (2012b)—*without* equality, the story can

[24] Syntactic freshness appears in this paper as $a \notin fa(r)$. We considered semantic freshness in Sect. 2.4.3. See also Proposition 2.201.

be different; this was encountered in the first attempt at a nominal functional programming language, in which we included freshness information in the types Gabbay (2001).

A.8 Permissive-Nominal Terms (Gabbay and Mulligan 2009; Dowek et al. 2009, 2010)

These simplify and improve classical nominal terms in two ways: we give explicitly the (countably infinite) atoms that may be free/are guaranteed to be fresh in every unknown, and since freshness information is stored directly we eliminate the need for freshness contexts. Thus good properties emerge: permissive-nominal terms can be constructed as nominal abstract syntax, we can directly choose a name fresh for a term (which is not possible in nominal terms without expanding the freshness context), and properties and proofs can then be expressed for terms alone, rather than for terms-in-freshness-context.

For instance, in classical nominal terms a solution of a nominal unification problem is a pair of a substitution and a freshness context; a nominal rewrite rule is a left and a right-hand side term and a freshness context; the proof-theory of nominal algebra requires an explicit freshness rule to generate fresh atoms, and so on. In fact, manipulating nominal terms almost always requires us to manipulate an external structure representing freshness constraints.

In contrast permissive terms are 'self-sufficient', like ordinary syntax. Proofs and algorithms have more of the look and feel of ordinary syntax. We have seen how, in the body of this survey. A detailed treatment of permissive-nominal syntax, including a simple translation from the nominal terms of Urban et al. (2004) into permissive-nominal terms, is also in Dowek et al. (2010).

A.9 Permissive-Nominal Algebra (Gabbay and Mulligan 2009, and Sect. 2.3.4)

The permissive-nominal algebra of Sect. 2.3.4 uses permissive-nominal terms and has a significantly different proof-theory.

The notable differences are, aside from being permissive-nominal, the inclusion (if we want them) of infinitely-supported constant symbols and of infinitely-supported permutations. So previous work is a special case of the general framework of this survey, but what we do here goes strictly beyond what was possible in previous work, also in some significant mathematical properties such as satisfying an HSP instead of an HSPA result; see the discussion opening Sect. 2.3.5.

A.10 Permissive-Nominal Logic (Dowek and Gabbay 2010, 2012a and Sect. 2.4.1)

As we discuss in this survey, permissive-nominal logic (**PNL**) adds universal quantification over unknowns X. This is non-evident for nominal terms because of their freshness contexts; in nominal terms X behaves like an element with cofinite support so we lose α-equivalence whereas in permissive-nominal terms X has coinfinite support and we can always α-rename bound atoms. We get a proof-theory which is pleasingly close to that of first-order logic, a sound and complete semantics, and we can axiomatise and prove correct a non-trivial and mathematically relevant theory, such as arithmetic.

Name	Intended model	Refs	Notes
FM set theory	Cumulative hierarchy / Cat. of FM sets	Gabbay and Pitts (1999, 2001)	Previously used to prove independence of AC
Nominal / FM sets	Themselves	Gabbay and Pitts (1999, 2001)	Nominal sets called 'equivariant FM sets' in these papers
Nom. logic	Schanuel topos / Cat. of nom. sets	Pitts (2001, 2003)	States axioms of nominal sets, used word 'nominal'
Nom. terms	Nom. sets	Urban et al. (2003, 2004)	Introduced a, X, $a\#X$, $[a]X$, $\pi{\cdot}X$
Nom. rewriting	Nom. terms	Fernández et al. (2004), Fernández and Gabbay (2007)	First framework for asserting general theories on nominal terms
αProlog	Nom. terms	Cheney and Urban (2003, 2008)	Intended as logic programming language for abstract syntax, but can be viewed more generally
Nom. algebra	Nom. sets	Gabbay (2005), Gabbay and Mathijssen (2006a, 2007, 2009)	Axiomatisation & models for binders like $[a{\mapsto}t]$, λa, $\forall a$
Nom. equational logic	Nom. sets / Nom. Lawvere Theories	Clouston and Pitts (2007), Clouston (2009)	Also provides semantics for nominal algebra
Permissive-nom. terms	Nom. sets or permissive-nom. sets	Dowek et al. (2009), Gabbay and Mulligan (2009); Dowek et al. (2010)	Eliminate freshness contexts; add *shift*-permutation; standardise α-equivalence
Permissive-nom. algebra	Permissive-nom. sets or nom. sets	Gabbay and Mulligan (2009), this survey	More expressive, esp. in presence of *shift*-permutation
Permissive-nom. logic	Permissive-nom. sets or nom. sets	Dowek and Gabbay (2010, 2012a), this survey	A first-order logic for nom. terms

Fig. A.1 Cheat-sheet of nominal languages

References

Abadi, M., L. Cardelli, P.-L. Curien, and J.-J. Lévy. 1991. Explicit substitutions. *Journal of Functional Programming* 1(4):375–416.

Baader, F., and T. Nipkow. 1998. *Term rewriting and all that*. Great Britain: Cambridge University Press.

Birkhoff, G. 1935. On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society* 31:433–454.

Burris, S.N., and H.P. Sankappanavar. 1981. A course in universal algebra. In *Graduate texts in mathematics*. Springer: New York.

Calvès, C. 2010. *Complexity and implementation of nominal algorithms*. Ph.D. thesis, King's College London.

Cheney, J. 2004. The complexity of equivariant unification. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, vol. 3142 of *Lecture notes in computer science*, 332–344. Berlin/New York: Springer.

Cheney, J. 2005a. Relating nominal and higher-order pattern unification. In *Proceedings of the 19th international workshop on Unification (UNIF 2005)*, 104–119. LORIA research report A05-R-022.

Cheney, J. 2005b. A simpler proof theory for nominal logic. In *FoSSaCS*, vol. 3441 of *Lecture notes in computer science*, 379–394. Berlin/New York: Springer.

Cheney, J. 2006. Completeness and Herbrand theorems for nominal logic. *Journal of Symbolic Logic* 71:299–320.

Cheney, J. 2010, October. Equivariant unification. *Journal of Automated Reasoning* 45(3):267–300.

Cheney, J., and C. Urban. 2003. System description: Alpha-Prolog, a fresh approach to logic programming modulo alpha-equivalence. In *UNIF'03*, 15–19. Universidad Politécnica de Valencia.

Cheney, J., and C. Urban. 2008. Nominal logic programming. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 30(5):1–47.

Clouston, R. 2007. Closed terms (unpublished notes). http://users.cecs.anu.edu.au/~rclouston/closedterms.pdf.

Clouston, R. 2009. *Equational logic for names and binding*. Ph.D. thesis, University of Cambridge, UK.

Clouston, R. 2011. Nominal Lawvere theories. In *Proceedings of the 18th international Workshop on Logic, Language, and Information (WoLLIC)*, vol. 6642 of *Lecture notes in computer science*. Berlin/Heidelberg/New York: Springer.

Clouston, R.A., and A.M. Pitts. 2007. Nominal equational logic. In *Computation, meaning and logic: Articles dedicated to Gordon Plotkin*, vol. 172 of *Electronic notes in theoretical computer science*, 223–257. Amsterdam: Elsevier.

Dershowitz, N., and J.-P. Jouannaud. 1989. Rewrite systems. In *Handbook of theoretical computer science: Formal methods and semantics*, vol. B, ed. J. van Leeuwen. Amsterdam/New York/Cambridge: Elsevier and MIT Press.

Dowek, G. 2001. Higher-order unification and matching. In *Handbook of automated reasoning*, 1009–1062. Amsterdam: Elsevier.

Dowek, G. and M.J. Gabbay. 2010. Permissive nominal logic. In *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and Practice of Declarative Programming (PPDP 2010)*, 165–176. New York: ACM Press.

Dowek, G., and M.J. Gabbay. 2012a. Permissive nominal logic (journal version). http://dl.acm.org/citation.cfm?doid=2287718.2287720. *Transactions on Computational Logic* 13(3).

Dowek, G., and M.J. Gabbay. 2012b. PNL to HOL: from the logic of nominal sets to the logic of higher-order functions. *Theoretical Computer Science* 451:38–69.

Dowek, G., M.J. Gabbay, and D.P. Mulligan. 2009. Permissive nominal terms and their unification. In *Proceedings of the 24th Italian Conference on Computational Logic (CILC'09)*.

Dowek, G., M.J. Gabbay, and D.P. Mulligan. 2010. Permissive nominal terms and their unification: An infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL* 18(6):769–822.

Dummett, M. 1977. *Elements of intuitionism*, 1st ed. Oxford: Clarendon Press.

Felleisen, M., and R. Hieb. 1992. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science* 103(2):235–271.

Fernández, M., and M.J. Gabbay. 2007. Nominal rewriting (journal version). *Information and Computation* 205(6):917–965.

Fernández, M., and M.J. Gabbay. 2010. Closed nominal rewriting and efficiently computable nominal algebra equality. http://arxiv.org/abs/1009.2791v1. In *Electronic proceedings in theoretical computer science*, vol. 34, 37–51.

Fernández, M., M.J. Gabbay, and I. Mackie. 2004. Nominal rewriting systems. In *Proceedings of the 6th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming (PPDP 2004)*, 108–119. New York: ACM Press.

Fiore, M., and C.-K. Hur. 2010. Second-order equational logic. In *Proceedings of the 19th EACSL annual conference on Computer Science Logic (CSL 2010)*, *Lecture notes in computer science*. Berlin: Springer.

Fiore, M.P., G.D. Plotkin, and D. Turi. 1999. Abstract syntax and variable binding. In *Proceedings of the 14th IEEE symposium on Logic in Computer Science (LICS 1999)*, 193–202. Los Alamitos: IEEE Computer Society Press.

Gabbay, M.J. 2001. *A theory of inductive definitions with alpha-equivalence*. Ph.D. thesis, University of Cambridge, UK.

Gabbay, M.J. 2005. Axiomatisation of first-order logic (talk). In *Second workshop on Computational Aspects of Nominal sets (CANS'05)*. London: King's College.

Gabbay, M.J. 2007a. Fresh logic. *Journal of Applied Logic* 5(2):356–387.

Gabbay, M.J. 2007b. A general mathematics of names. *Information and Computation* 205(7):982–1011.

Gabbay, M.J. 2009. Nominal algebra and the HSP theorem. *Journal of Logic and Computation* 19(2):341–367.

Gabbay, M. 2011a. A proof-theoretic treatment of lambda-reduction with cut-elimination: Lambda calculus as a logic programming language. *Journal of Symbolic Logic* 76(2):673–699.

Gabbay, M.J. 2011b. Foundations of nominal techniques: Logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic* 17:(2):161–229.

Gabbay, M.J. 2011c. Two-level nominal sets and semantic nominal terms: An extension of nominal set theory for handling meta-variables. *Mathematical Structures in Computer Science* 21:997–1033.

Gabbay, M.J. 2012a. Meta-variables as infinite lists in nominal terms unification and rewriting. *Logic Journal of the IGPL* 20:967–1000.

Gabbay, M.J. 2012b. Unity in nominal equational reasoning: The algebra of equality on nominal sets. *Journal of Applied Logic* 10:199–217.

Gabbay, M.J., and J. Cheney. 2004. A sequent calculus for nominal logic. In *Proceedings of the 19th IEEE symposium on Logic in Computer Science (LICS 2004)*, 139–148. Los Alamitos: IEEE Computer Society.

Gabbay, M.J., and M. Hofmann. 2008. Nominal renaming sets. In *Proceedings of the 15th international conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008)*, 158–173. Berlin: Springer.

Gabbay, M.J., and A. Mathijssen. 2006a. Capture-avoiding substitution as a nominal algebra. In *ICTAC 2006: Theoretical aspects of computing*, vol. 4281 of *Lecture notes in computer science*, 198–212. Berlin: Springer.

Gabbay, M.J., and A. Mathijssen. 2006b. Nominal algebra. In *18th Nordic workshop on programming theory*.

Gabbay, M.J., and A. Mathijssen. 2006c. One-and-a-halfth-order logic. In *Proceedings of the 8th ACM-SIGPLAN international symposium on Principles and Practice of Declarative Programming (PPDP 2006)*, 189–200. New York: ACM.

Gabbay, M.J., and A. Mathijssen. 2007. A formal calculus for informal equality with binding. In *WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation*, vol. 4576 of *Lecture notes in computer science*, 162–176. Berlin/New York: Springer.

Gabbay, M.J., and A. Mathijssen. 2008a. Capture-avoiding substitution as a nominal algebra. *Formal Aspects of Computing* 20(4-5):451–479.

Gabbay, M.J., and A. Mathijssen. 2008b. The lambda-calculus is nominal algebraic. In *Reasoning in simple type theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, Studies in logic and the foundations of mathematics, ed. C. Benzmüller, C. Brown, J. Siekmann, and R. Statman. London: IFCoLog.

Gabbay, M.J., and A. Mathijssen. 2008c. One-and-a-halfth-order logic. *Journal of Logic and Computation* 18(4):521–562.

Gabbay, M.J., and A. Mathijssen. 2009. Nominal universal algebra: Equational logic with names and binding. *Journal of Logic and Computation* 19(6):1455–1508.

Gabbay, M.J., and D.P. Mulligan. 2009. Universal algebra over lambda-terms and nominal terms: The connection in logic between nominal techniques and higher-order variables. In *Proceedings of the 4th international workshop on Logical Frameworks and Meta-Languages (LFMTP 2009)*, 64–73. New York: ACM.

Gabbay, M.J., and A. Mathijssen. 2010. A nominal axiomatisation of the lambda-calculus. *Journal of Logic and Computation* 20(2):501–531.

Gabbay, M.J., and A.M. Pitts. 1999. A new approach to abstract syntax involving binders. In *Proceedings of the 14th annual symposium on Logic in Computer Science (LICS 1999)*, 214–224. Los Alamitos: IEEE Computer Society Press.

Gabbay, M.J., and A.M. Pitts. 2001. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing* 13:(3–5):341–363.

Gentzen, G. 1935. Untersuchungen über das logische Schließen [Investigations into logical deduction]. *Mathematische Zeitschrift* 39:176–210,405–431. Translated in Szabo (1969), pp. 68–131.

Kurz, A., and D. Petrişan. 2010. On universal algebra over nominal sets. *Mathematical Structures in Computer Science* 20:285–318.

Levy, J., and M. Villaret. 2008. Nominal unification from a higher-order perspective. In *Rewriting Techniques and Applications, proceedings of RTA 2008*, vol. 5117 of *Lecture notes in computer science*. Berlin/New York: Springer.

Levy, J., and M. Villaret. 2010. An efficient nominal unification algorithm. In *Proceedings of the 21st international conference on Rewriting Techniques and Applications (RTA 2010)*, vol. 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 209–226. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Levy, J., and M. Villaret. 2012. Nominal unification from a higher-order perspective. *Transactions on Computational logic (TOCL)* 13(2):10.

Mac Lane, S., and I. Moerdijk. 1992. *Sheaves in geometry and logic: A first introduction to topos theory*. Universitext. New York: Springer.

Mathijssen, A. 2007. *Logical calculi for reasoning with binding*. Ph.D. thesis, Technische Universiteit Eindhoven.

Melliès, P.-A., 1995. Typed lambda-calculi with explicit substitutions may not terminate. In *Proceedings of the 2nd international conference on Typed Lambda Calculi and Applications, (TLCA 1995)*, vol. 902 of *Lecture notes in computer science*, ed. M. Dezani-Ciancaglini, and G.D. Plotkin, 328–334. Berlin/New York: Springer.

Miller, D., G. Nadathur, F. Pfenning, and A. Scedrov. 1989. Uniform proofs as a foundation for logic programming. Technical report, Durham, NC.

Pitts, A. 2011. Nominal sets and their applications. In *Midlands Graduate School (MGS 2011)*. Available online at cl.cam.ac.uk/~amp12/talks/MGS2011_nominal_sets_slides.pdf.

Pitts, A.M. 2001. Nominal logic: A first order theory of names and binding. In *Proceedings of the 4th international symposium on Theoretical Aspects of Computer Software (TACS 2001)*, vol. 2215 of *Lecture notes in computer science*, ed. N. Kobayashi, and B.C. Pierce, 219–242. Berlin/New York: Springer.

Pitts, A.M. 2003. Nominal logic, a first order theory of names and binding. *Information and Computation* 186(2):165–193.

Prawitz, D. 1965. *Natural deduction: A prooof-theoretical study*. Stockholm: Almquist and Wiksell. Reprinted by Dover, 2006.

Shoenfield, J. 1967. *Mathematical logic*. Reading, MA: Addison-Wesley.

Smullyan, R. 1968. *First-order logic*. Berlin/New York: Springer. Reprinted by Dover, 1995.

Turner, D.C. 2009. *Nominal Domain theory for concurrency*. Ph.D. thesis, University of Cambridge.

Tzevelekos, N. 2007. Full abstraction for nominal general references. In *Proceedings of the 22nd IEEE symposium on Logic in Computer Science (LICS 2007)*, 399–410. Los Alamitos: IEEE Computer Society Press.

Urban, C. 2008. Nominal reasoning techniques in Isabelle/HOL. *Journal of Automatic Reasoning* 40(4):327–356.

Urban, C., A.M. Pitts, and M.J. Gabbay. 2003. Nominal unification. In *Proceedings of the 17th international workshop on Computer Science Logic (CSL 2003)*, vol. 2803 of *Lecture notes in computer science*, 513–527. Berlin/New York: Springer.

Urban, C., A.M. Pitts, and M.J. Gabbay. 2004. Nominal unification. *Theoretical Computer Science* 323(1–3):473–497.

Chapter 3
Introduction to Labelled Deductive Systems

Dov M. Gabbay

3.1 Labelled Deductive Systems in Context

In the past 40 years logic has undergone a serious evolutionary development. The meteoric rise of the applied areas of computer science and artificial intelligence put pressure on traditional logic to evolve. There was the urgent need to develop new logics in order to provide better models of human behaviour and actions. Such models are used to help design products which aid/replace the human in his daily activity. As a result, a rich variety of new logics have been developed and there was the need for a new unifying methodology for the chaotic landscape of the new logics.[1]

Such a methodology is *Labelled Deductive Systems* (*LDS*), introduced in the 1990 and unified many discrete logical systems and later evolved to network logics, 2009–2011, unifying logic and network reasoning.

The purpose of this chapter is to first introduce Labelled Deductive Systems and show that many logical systems, new and old, monotonic and non-monotonic all fall within this new framework. The most recent answer to the question of what is a logical system, which integrates both logic and network reasoning can be found in our very recent papers (Gabbay 2011, 2012).

We begin with the traditional view of what is a logical system.

Traditionally, to present a logic **L**, we need to first present the set of well-formed formulas of that logic. This is the *language* of the logic. We specify the sets of atomic formulas, connectives, quantifiers and the set of well-formed formulas.

[1] See the editorial for this volume.

D.M. Gabbay (✉)
Bar Ilan University, Israel; King's College London, UK; University of Luxembourg, Luxembourg; University of Manchester, UK

D.M. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic: Volume 17*, Handbook of Philosophical Logic 17, DOI 10.1007/978-94-007-6600-6_3,

Secondly, we mathematically define the notion of consequence, that is, for sets of formulas Δ and formulas Q, we define the consequence relation $\Delta \vdash_{\mathbf{L}} Q$, which is read "$Q$ follows from Δ in the logic $\mathbf{L}$".

The consequence relation is required to satisfy the following intuitive properties: (Δ, Δ' abbreviates $\Delta \cup \Delta'$).

Reflexivity

$$\Delta \vdash Q \text{ if } Q \in \Delta$$

Monotonicity

$$\frac{\Delta \vdash Q}{\Delta, \Delta' \vdash Q}$$

Transitivity

$$\frac{\Delta \vdash A; \Delta, A \vdash Q}{\Delta \vdash Q}$$

If you think of Δ as a database and Q as a query, then reflexivity means that the answer "yes" is given for any Q which is already listed in the database Δ. Monotonicity reflects the accumulation of data, and transitivity is nothing but lemma generation, namely, if $\Delta \vdash A$, then A can be used as a lemma to derive B from Δ.

These three properties have appeared to constitute a set of minimal and most natural requirements for a logical system, given that the main applications of logic were in mathematics and philosophy.

The above notions were essentially put forward by Tarski in (1936) and is referred to as Tarski consequence. Scott (1974), inspired by constructions in Gabbay (1991a), generalised the notion to allow Q to be a set of formulas Γ. The basic relation is then of the form $\Delta \vdash \Gamma$, satisfying:[2]

Reflexivity

$$\Delta \vdash \Gamma \text{ if } \Delta \cap \Gamma \neq \varnothing$$

Monotonicity

$$\frac{\Delta \vdash \Gamma}{\Delta, \Delta' \vdash \Gamma}$$

Cut

$$\frac{\Delta, A \vdash \Gamma; \Delta' \vdash A, \Gamma'}{\Delta, \Delta' \vdash \Gamma, \Gamma'}$$

[2]The similarity with Gentzen sequents is obvious. A sequent $\Delta \vdash \Gamma$ is a relation between Δ and Γ. Such a relation can either be defined axiomatically (as a consequence relation) or be generated via closure conditions like $A \vdash A$ (initial) and other generating rules. The generating rules correspond to Gentzen rules. In many logics we have $\Delta \vdash \Gamma$ iff $\varnothing \vdash \bigwedge \Delta \to \bigvee \Gamma$, which gives an intuitive meaning to $\vdash$.

Scott further showed that for any Tarski consequence relation $\vdash$ there exist two Scott consequence relations (a maximal one and a minimal one) that agree with it, namely, that $\Delta \vdash A$ (Tarski) iff $\Delta \vdash \{A\}$ (Scott) (see Gabbay 1981).

The above notions are monotonic. However, the increasing use of logic in computer science and artificial intelligence has given rise to logical systems which are not monotonic, i.e., to systems in which the axiom of monotonicity is not satisfied. There are many such systems, satisfying a variety of conditions and presented in a variety of ways. Furthermore, some are characterized in a proof theoretical and some in a model theoretical manner. All these different presentations give rise to some notion of consequence $\Delta \vdash Q$, but they only seem to all agree on reflexivity.[3] The essential difference between these logics (commonly called *non-monotonic logics*) and the more traditional logics (now referred to as *monotonic logics*) is the fact that $\Delta \vdash A$ holds in the monotonic case because of some $\Delta_A \subseteq \Delta$, while in the non-monotonic case the entire set Δ is somehow used to derive A. Thus if Δ is increased to Δ', there is no change in the monotonic case, while there may be a change in the non-monotonic case.

The above describes the situation current in the early 1980's. We have had a multitude of systems generally accepted as "logics" without a unifying underlying theory and many had semantics without proof theory or vice-versa, though almost all of them were based on some sound intuitions of one form or another. Clearly there was the need for a general unifying framework. An early attempt at classifying non-monotonic systems was Gabbay (1985). It was put forward that basic axioms for a Tarski type consequence relation should be *reflexivity, transitivity* and *restricted monotonicity*, namely:

Restricted Monotonicity (Cumulativity)

$$\frac{\Delta \vdash A; \Delta \vdash B}{\Delta, A \vdash B}$$

A variety of systems seem to satisfy this axiom. See a survey in Makinson (1994) and Gabbay (1996).

Although some sort of classification was obtained and semantical results were proved, the approach does not seem to be strong enough. Many systems do not satisfy restricted monotonicity. Other systems such as relevance logic, do not even satisfy reflexivity. Others have a richness of their own which is lost in a simple presentation as an axiomatic consequence relation. Obviously a different approach is needed, one which would be more sensitive to the variety of features of the systems in the field. Fortunately, developments in a neighbouring area, that of automated deduction, seem to be of help. New automated deduction methods were developed for non classical logics, and resolution was generalised and modified to be applicable to these logics. In general, because of the value of these logics in theoretical computer science and artificial intelligence, a greater awareness of the computational aspects

[3] With the exception of the highly successful input output logics which were put forward by Makinson–Torre which do not satisfy reflexivity (Makinson and van der Torre 2001).

of logical systems was developing and more attention was being devoted to proof-theoretical presentations. It became apparent to us that a key feature in the proof-theoretic study of these logics is that a slight natural variation in an automated or proof-theoretic system of one logic (say $\mathbf{L}_1$), can yield another logic (say $\mathbf{L}_2$).

Although $\mathbf{L}_1$ and $\mathbf{L}_2$ may be conceptually far apart (in their philosophical motivation, and mathematical definitions) when it comes to automated techniques and proof theoretical presentation, they turn out to be brother and sister. This kind of relationship is not isolated and seems to be widespread. Furthermore, non monotonic systems seem to be obtainable from monotonic ones through variations on some of their monotonic proof-theoretical formulation, thus giving us a handle on classifying non-monotonic systems.

This phenomena has prompted Gabbay (Gabbay 1992a; Gabbay and Olivetti 2000) to put forward the view that a logical system $\mathbf{L}$ is not just the traditional consequence relation $\vdash$ (monotonic or non monotonic) but a pair $(\vdash, \mathbf{S}_\vdash)$, where $\vdash$ is a mathematically defined consequence relation (i.e. the set of pairs (Δ, Q) such that $\Delta \vdash Q$) satisfying whatever minimal conditions on a consequence relation one happens to agree on, and $\mathbf{S}_\vdash$ is an algorithmic system for generating all those pairs. Thus according to this definition, classical logic $\vdash$ perceived as a set of tautologies together with a Gentzen system $\mathbf{S}_\vdash$, is not the same as classical logic together with the two-valued truth table decision procedure $\mathbf{T}_\vdash$ for it. In our conceptual framework,$(\vdash, \mathbf{S}_\vdash)$ is *not the same logic* as $(\vdash, \mathbf{T}_\vdash)$.

To illustrate and motivate our way of thinking, observe that it is very easy to move from $\mathbf{T}_\vdash$ for classical logic to a truth table system $\mathbf{T}^n_\vdash$ for Łukasiewicz n-valued logic. It is not so easy to move to an algorithmic system for intuitionistic logic. In comparison, for a Gentzen system presentation, exactly the opposite is true. Intuitionistic and classical logics are neighbours, while Łukasiewicz logics seem completely different.[4] In fact, some of the examples of this chapter show proof theoretic similarities between Łukasiewicz's infinite valued logic and Girard's Linear Logic, which in turn is proof theoretically similar to intuitionistic logic.

There are many more such examples among temporal logics, modal logics, defeasible logics and others. Obviously, there is a need for a more unifying framework. The question is then whether we can adopt a concept of a logic where the passage from one system to another is natural, and along predefined acceptable modes of variation? Can we put forward a framework where the computational aspects of a logic also play a role? Is it possible to find a common home for a variety of seemingly different techniques introduced for different purposes in seemingly different intellectual logical traditions?

To find an answer, let us ask ourselves what makes one logic different from another? How is a new logic presented and described and compared to another?

[4]This example was put forward in the mid-1980s. Since then work on Gentzen formulations of many valued logics has progressed, with contributions by many authors. See for example our book Metcalfe et al. (2008). I still think however, that the Gentzen connection between classical logic and intuitionistic logic is much more straight forward then the connection with fuzzy logics.

The answer is obvious. These considerations are usually dealt with on the meta-level. Most logics are based on modus ponens and the quantifier rules are formally the same anyway and the differences between them are meta-level considerations on the proof theory or semantics. If we can find a mode of presentation of logical systems where metalevel features and semantic features can reside side by side with object level features then we can hope for a general framework. We must be careful here. In the logical community the notions of object-level vs meta-level are not so clear. Most people think of *naming* and *proof predicates* in this connection. This is not what we mean by meta-level here. We need a more refined understanding of the concept. There is a similar need in computer science. In Gabbay (1996) we devote a chapter to these considerations. See also Gabbay (1992c).

We found that the best framework to put forward is that of a *Labelled Deductive System, LDS*. Our notion of what constitutes a logic will be that of a pair $(\vdash, \mathbf{S}_\vdash)$ where $\vdash$ is a set-theoretic (possibly non-monotonic) consequence relation on a language $\mathbf{L}$ and $\mathbf{S}_\vdash$ is an *LDS*, and where $\vdash$ is essentially required to satisfy no more than *Identity* (i.e. $\{A\} \vdash A$) and *Surgical Cut* (see below and Gabbay 1991b, 1993b). This is a refinement of our concept of a logical system mentioned above and first presented in Gabbay (1992a). We now not only say that a logical system is a pair $(\vdash, \mathbf{S}_\vdash)$, but we are adding that $\mathbf{S}_\vdash$ itself has a special presentation, that of an *LDS*.

An *LDS* system is a triple $(\mathbf{L}, \Gamma, \mathbf{M})$, where $\mathbf{L}$ is a logical language (connectives and wffs) and Γ is an algebra (with some operations) of labels and $\mathbf{M}$ is a discipline of labelling formulas of the logic (from the algebra of labels Γ), together with deduction rules and with agreed ways of propagating the labels via the application of the deduction rules. The way the rules are used is more or less uniform to all systems. In the general case we allow Γ, the algebra of labels, to be an *LDS* system itself! Furthermore, if our view of a logical system is that the declarative unit is a pair, a formula and a label, then we can also label the pair itself and get multiple labelling.

The perceptive reader may feel resistance to this idea at this stage. First be assured that you are not asked to give up your favourite logic or proof theory nor is there any hint of a claim that your activity is now obsolete. In mathematics a good concept can rarely be seen or studied from one point of view only and it is a sign of strength to have several views connecting different concepts. So the traditional logical views are as valid as ever and add strength to the new point of view. In fact, a closer examination of the material in my book (Gabbay, 1996) would reveal that manifestations of our *LDS* approach already exist in the literature in various forms (see Anderson and Belnap 1975 and Fitting 1983; Gabbay 1996 and the references there), however, they were locally regarded as convenient tools and there was not the realisation that there is a general framework to be studied and developed. None of us is working in a vacuum and we build on each others work. Further, the existence of a general framework in which any particular case can be represented does not necessarily mean that the best way to treat that particular case is within the general framework. Thus if some modal logics can be formulated in *LDS*, this does not mean that in practice we should replace existing ways of treating the logics by their *LDS* formulation. The latter may not be the most efficient for those particular logics. It is

sufficient to show how the *LDS* principles specialise and manifest themselves in the given known practical formulation of the logic.

The reader may further have doubts about the use of labels from the computational point of view. What do we mean by a unifying framework? Surely a Turing machine can simulate any logic, is that a unifying framework? The use of labels is powerful, as we know from computer science, are we using labels to play the role of a Turing machine? The answer to the question is twofold. First that we are not operating at the metalevel, but at the object level. Second, there are severe restrictions on the way we use *LDS*. Here is a preview:

1. The only rules of inference allowed are the traditional ones, modus ponens and some form of deduction theorem for implication, for example.
2. Allowable modes of label propagation are fixed for all logics. They can be adjusted in agreed ways to obtain variations but in general the format is the same. For example, it has the following form for implications:
   $(A \to B)$ gets label t iff $\forall x \in \Gamma_1$ [If A is labelled x then B can be proved with labels $t + x$], where Γ_1 is a set of labels characterising the implication in that particular logic. For example Γ_1 may be all atomic labels or related labels to t, or variations. The freedom that different logics have is in the choice of Γ_1 and the properties of "+". For example we can restrict the use of modus ponens by a wise propagation of labels.
3. The quantifier rules are the same for all logics.
4. Metalevel features are implemented via the labelling mechanism, which is object language.
5. The labels can be a reflection of the intended semantics for the logic (e.g. time stamps for temporal logic).

The reader who prefers to remain within the traditional point of view of:

assumptions (data) proving a conclusion

can view the labelled formulas as another form of data.

There are many occasions when it is most intuitive to present an item of data in the form $t : A$, where t is a label and A is a formula. The common underlying reason for the use of the label t is that t represents information which is needed to modify A or to supplement (the information in) A which is not of the same type or nature as (the information represented by) A itself. A is a logical formula representing information declaratively, and the additional information of t can certainly be added declaratively to A to form A', however, we may find it convenient to put forward the additional information through the label t as part of a pair $t : A$.

Take for example a source of information which is not reliable. A natural way of representing an item of information from that source is $t : A$, where A is a declarative presentation of the information itself and t is a number representing its reliability. Such expert systems exist (e.g. Mycin) with rules which manipulate both t and A as one unit, propagating the reliability values t_i through applications of modus ponens. We may also use a label naming the source of information and this would give us a qualitative idea of its reliability.

Another area where it is natural to use labels is in reasoning from data and rules. If we want to keep track, for reasons of maintaining consistency and/or integrity constraints, where and how a formula was deduced, we use a label t. In this case, the label t in $t : A$ can be the part of the data which was used to get A. Formally in this case t is a formula, the conjunction of the data used. We thus get pairs of the form $\Delta_i : A_i$, where A_i are formulas and Δ_i are the parts of the database from which A_i was derived.

A third example where it is natural to use labels is time stamping of data. Where data is constantly revised and updated, it is important to time stamp the data items. Thus the data items would look like $t_i : A_i$, where t_i are time stamps. A_i itself may be a temporal formula. Thus there are two times involved, the logical time s_i in $A_i(s_i)$ and the time stamping t_i of A_i. For reasons of clarity, we may wish to regard t_i as a label rather than incorporate it into the logic (by writing for example $A^*(t_i, s_i)$).

To summarise then, we replace the traditional notion of consequence between formulas of the form $A_1, \ldots, A_n \vdash B$ by the notion of consequence between labelled formulas

$$t_1 : A_1, t_2 : A_2, \ldots t_n : A_n \vdash s : B$$

Depending on the logical system involved, the intuitive meaning of the labels varies. In querying databases, we may be interested in labelling the assumptions so that when we get an answer to a query, we can record, via the label of the answer, from which part of the database the answer was obtained. Another area where labelling is used is temporal logic. We can time stamp assumptions as to when they are true and query, given those assumptions, whether a certain conclusion will be true at a certain time. Thus the consequence notion for labelled deduction is essentially the same as that of any logic:

> given assumptions we ask does a conclusion follow?

Whereas in the traditional logical system the consequence is defined using proof rules on the formulas, in the *LDS* methodology the consequence is defined by using rules on both formulas and their labels. Formally we have formal rules for manipulating labels and this allows for more scope in decomposing the various features of the consequence relation. The meta features can be reflected in the algebra or logic of the labels and the object features can be reflected in the rules of the formulas.

The notion of a database or of a "set of assumptions" also has to be changed. A database is a configuration of labelled formulas. The configuration depends on the labelling discipline. For example, it can be a linearly ordered set $\{a_1 : A_1, \ldots, a_n : A_n\}$, $a_1 < a_2 < \ldots < a_n$. The proof discipline for the logic will specify how the assumptions are to be used. We need to develop the notions of the Cut Rule and the Deduction Theorem in such an environment. This we do in the monograph Gabbay (1996).

The next two sections will give many examples of *LDS* disciplines featuring many known monotonic and non-monotonic logics. It is of value for now to summarise our view listing the key points involved:

- The unit of declarative data is a labelled formula of the form $t : A$, where A is a wff of a language $\mathbf{L}$ and t is a label. The labels come from an algebra (set) of labels.
- A database is a set of labelled formulas. The database has structure arising from the algebraic relationships of the labels.
- An *LDS* discipline is a system (algorithmic) for manipulating both formulas and their labels. Using this discipline the statement $\Delta \vdash \Gamma$ is well defined for the two databases Δ and Γ. Especially $\Delta \vdash t : A$ is well defined.
- $\vdash$ must satisfy the minimal conditions, namely

Identity

$$\{t : A\} \vdash t : A$$

Surgical Cut

$$\frac{\Delta \vdash t : A, \Gamma[t : A] \vdash s : B}{\Gamma[\Delta] \vdash s : B}$$

where $\Gamma[t : A]$ means that $t : A$ is contained/occurs somewhere in the structure Γ and $\Gamma[\Delta]$ means that Δ replaces A in the structure. The notion of how this substitution/replacement is to be done is also part of the *LDS* logic.

- A logical system is a pair $(\vdash, \mathbf{S}_{\vdash})$, where $\vdash$ is a consequence relation and $\mathbf{S}_{\vdash}$ is an *LDS* for it.

3.2 Examples from Monotonic Logics

To motivate our approach we study several known examples in this section.

We start with Example 3.1, a practical insurance example, which will illustrate the usefullness of labels.

Example 3.2 shows a standard deduction from Relevance Logic. The purpose of the example is to illustrate our point of view. There are many such examples in Anderson and Belnap (1975). Example 3.8 considers a derivation in modal logic. There we use labels to denote essentially possible worlds. The objective of the example is to show the formal similarities to the relevance logic case in Example 3.2. Example 3.9 can reap the benefits of the formal similarities of the first two examples and introduce, in the most natural way, a system of relevant modal logic. The objective of Example 3.9 is to show that the labels in Examples 3.2 and 3.8 can be read as determining the metalanguage features of the logic and can therefore be combined "declaratively" to form the new system of Example 3.9. Example 3.10

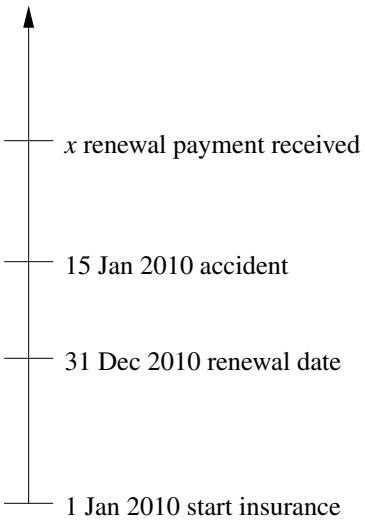


Fig. 3.1

considers strict implication. This example shows that for strict **S4** implication one can read the labels either as relevance labels or as possible world labels. Example 3.11 shows how labels can interact with quantifiers in modal logic. We continue with examples of relevance reasoning, many-valued logics, formulas as types, realisability and conclude with a formal definition of an algebraic *LDS* for $\rightarrow$ and $\neg$.

Example 3.1 (Insurance example). Consider any insurance policy covering car accidents. The policy can be simply formalised in logic as an implication

$$\text{car accident} \rightarrow \text{pay damages}.$$

The policy has a starting date, say $s = 1$ January 2010 and is valid for a calendar year, which, for the purpose of calculation we use 365 days. If an accident occurs at date t, we need to check that the policy was in force at the time. If yes, then payment is within 30 days. Formally we have

$$\begin{array}{ll} 1. & s: \text{ accident} \rightarrow \text{pay} \\ 2. & t: \text{ accident} \\ 3. & s \leq t \leq s+365 \\ \hline 4. & t+30: \text{ pay} \end{array}$$

Let's look further into this example. Many people renew their policies every year. The insurance companies allow customers one month to pay the renewal fee. So we may have the following situation, see Fig. 3.1.

When a claim is made, the insurance company will check whether the date of the renewal payment is before 1 February 2011. If yes, then the accident on 15 January

2011 is covered. Formally the modus ponens has the following form for renewal policies:

$$\begin{array}{ll} 1. & s: \text{ accident} \wedge \text{renewal} \rightarrow \text{pay} \\ 2. & x: \text{ policy renewal} \\ 3. & t: \text{ accident} \\ 4. & s+365+31 > x \text{ and} \\ & s+365+365 > t \\ \hline 4. & t+30: \text{ pay} \end{array}$$

The above has the following pattern

$$\begin{array}{ll} 1. & s: A \rightarrow B \\ 2. & t: A \\ 3. & \varphi(s,t) \text{ holds} \\ \hline 4. & \mathbf{f}(s,t): B \end{array}$$

$\varphi(s,t)$ is to check whether the labels match for the purpose of allowing the modus ponens to go through and $\mathbf{f}(s,t)$ is the new label of the result of executing the modus ponens.

Let us continue the story. Suppose we do have an accident at time t and the car is a total loss. The car owner goes to a dealer with a letter from the insurance company that they will pay so much. The owner wants to get a car immediately from the dealer and let the dealer settle with the insurance company. The dealer checks whether this insurance company is listed as one of the companies he has an arrangement with. If yes, the deal can go forward. So we have (**d** is the dealer, **n** is the insurance company):

$$\begin{array}{ll} 5. & \mathbf{d}: \text{credit OK} \rightarrow \text{deliver new car} \\ 6. & \mathbf{n}: \text{ credit} \\ 7. & \varphi(\mathbf{d},\mathbf{n}) \text{ holds} \\ \hline 8. & \mathbf{f}(\mathbf{d},\mathbf{n}): \text{ deliver new car} \end{array}$$

To code the entire story we need to add the name of the insurance company **n** to the insurance labels. We get:

1. $(\mathbf{n},s)$: Accident $\wedge$ renewal $\rightarrow$ pay
2. $(\mathbf{n},x)$: policy renewed
3. t : accident
4. **d**: credit OK $\rightarrow$ deliver new car
5. $(\mathbf{n},t+30)$: pay $\rightarrow (\mathbf{n},t+15)$: credit

The above is the data, and we want to derive

6. $(\mathbf{n},\mathbf{d},t+15)$: new car.

Here is the proof

7. Confirm that $s+365+31 > x$ and $s+365+365 > t$ holds.
8. $(\mathbf{n}, t+30)$: pay, from (1), (2), (3), (6), and (7)
9. $(\mathbf{n}, t+15)$: credit, from (8) and (5)
10. Confirm that **n** and **d** work together
11. $(\mathbf{n}, \mathbf{d}, t+15)$: new car, from (1), (4), and (10).

Example 3.2 (Relevance and Linear Logic). Consider a propositional language with implication "$\rightarrow$" only. The forward elimination rule is modus ponens. From the theorem proving view, modus ponens is an object language consideration. Thus a proof of $\vdash (B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))$ can proceed as follows:

Assume $a_1 : B \rightarrow A$ and show $(A \rightarrow B) \rightarrow (A \rightarrow B)$. Further assume $a_2 : A \rightarrow B$ and show $A \rightarrow B$. Further assume $a_3 : A$ and show B. We thus end up with the following problem:

Assumptions

1. $a_1 : B \rightarrow A$
2. $a_2 : A \rightarrow B$
3. $a_3 : A$

Derivation

4. $a_2 a_3 : B$ by modus ponens from lines (2) and (3).
5. $a_1 a_2 a_3 : A$ from (4) and (1).
6. $a_2 a_1 a_2 a_3 : B$ from (5) and (2).
7. $a_2 a_1 a_2 : A \rightarrow B$ from (3) and (6) and $\rightarrow$ Introduction rule.

We delete the label of (3) from the label of (6) and get the label of (7).

8. $a_2 a_1 : (A \rightarrow B) \rightarrow (A \rightarrow B)$ from (2) and (7) and $\rightarrow$ Introduction rule.

We delete the label of (2) from the label of (7) and get the label of (8).

9. $a_2 : (B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))$ from (1) and (8) and $\rightarrow$ Introduction rule.

We delete the label of (1) from the label of (8) and get the label of (9).

The meta aspect of this proof is the annotation of the assumptions and the keeping track of what was used in the deduction. A metaleval condition would determine the logic involved. For example item 2, which has label a_2 was used twice in the proof.

This explains why one copy of the label a_2 remained in line 9. A metalevel condition might say we do not accept such proofs.

A formal definition of the labelling discipline for this class of logics is given in Gabbay (1996). For this example it is sufficient to note the following three conventions:

1. Each assumption is labelled by a new atomic label.
   An ordering on the labels can be imposed, namely $a_1 < a_2 < a_3$. This is to reflect the fact that the assumptions arose from our attempt to prove $(B \to A) \to ((A \to B) \to (A \to B))$ and not for example from $(A \to B) \to ((B \to A) \to (A \to B))$ in which case the ordering would be $a_2 < a_1 < a_3$. The ordering can affect the proofs in certain logics.
2. If in the proof, A is labelled by the multiset α and $A \to B$ is labelled by β then B can be derived with a label $\alpha \cup \beta$ where "$\cup$" denotes multiset union.
3. If B was derived using A as evidenced by the fact that the label α of A is a submultiset of the label β of $B (\alpha \subseteq \beta)$ then we can derive $A \to B$ with the label $\beta - \alpha$ ("-" is multiset subtraction).

The derivation can be represented in a more graphical way.

To show $(B \to A) \to ((A \to B) \to (A \to B))$ see Fig. 3.2.

The above is the *metabox* way of representing the deduction. Note that in line 8, multiset subtraction was used and only one copy of the label a_2 was taken out. The other copy of a_2 remains and cannot be cancelled. Thus this formula is not a theorem of linear logic, because the outer box does not exit with label $\varnothing$. In relevance logic, the discipline uses sets and not multisets. Thus the label of line 8 in this case would be a_1 and that of line 9 would be $\varnothing$. The above deduction can be made even more explicit as follows:

$(B \to A) \to ((A \to B) \to (A \to B))$ follows with a label from Box a_1.

Box a_1

a_1 :	$B \to A$ assumption
$a_2 a_1$:	$(A \to B) \to (A \to B)$ from Box a_2

Box a_2

a_2 :	$A \to B$ assumption
$a_2 a_1 a_2$:	$A \to B$ from Box a_3

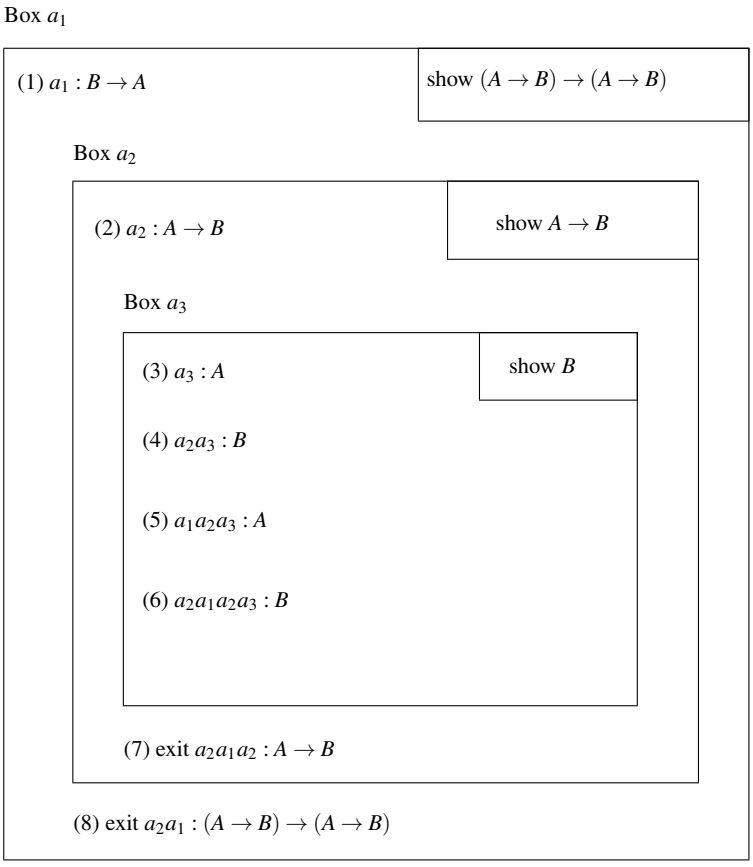


Fig. 3.2

Box a_3

$a_3 : A$	assumption
$a_2 : A \to B$	reiteration from box a_2
$a_2a_3 : B$	by modus ponens
$a_1 : B \to A$	reiteration from box a_1
$a_1a_2a_3 : A$	modus ponens from the two preceding lines
$a_2 : A \to B$	repetition of an earlier line
$a_2a_1a_2a_3 : B$	modus ponens from the two preceding lines

The following meta-rule was used:

- We have a systems of partially ordered metaboxes $a_1 < a_2 < a_3$. Any assumption in a box a can be reiterated in any box b provided $a < b$.

Remark 3.3. a. The above presentation of the boxes makes them look more like possible worlds. The labels are the worlds and formulas can be exported from one world to another according to some rules. Example 3.8 describes modal logic in just this way.

b. Note that different meta-conditions on labels and metaboxes correspond to different logics.

The following table gives intuitively some correspondence between metaconditions and logics.

Meta-condition:	**Logic**
Ignore the labels	Intuitionistic logic
Accept only the derivations which use all the assumptions	Relevance logic
Accept derivations which use all assumptions exactly once	Linear logic

The meta-conditions can be translated into object conditions in terms of axioms and rules. If we consider a Hilbert system with modus ponens and substitution then the additional axioms involved are given below:

Linear Logic
$A \to A$
$(A \to (B \to C)) \to (B \to (A \to C))$
$(C \to A) \to ((B \to C) \to (B \to A))$
$(C \to A) \to ((A \to B) \to (C \to B))$

Relevance Logic
Add the schema below to linear logic
$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

Intuitionistic Logic
Add the schema below to relevance logic:
$A \to (B \to A)$
The reader can note that the following axiom (Peirce Rule) yields classical logic. Further note that for example, we can define "Linear Classical Logic" by adding Peirce Rule to linear logic. A new logic is obtained.

Classical Logic
Add the schema below to intuitionistic logic:
$((A \to B) \to A) \to A$.

Remark 3.4 (Full implicational LDS*).* This remark will show how to formulate an algebraic labelled deductive system for a language with $\to$ only. All relevant parameters will be used. We need the following components.

1. The logic language is propositional. It has atoms $\{p,q,r\ldots\}$ and implication $\rightarrow$.
2. We need an algebra $(\mathcal{A},*)$, where $*$ is a binary operation on $\mathcal{A}$. For example, $*$ may be an associative semigroup operation. Another possibility is to take the algebra $\mathcal{A}$ as all finite sequence of some set off atomic labels and $*$ to be concatenation.
3. We need to assume further properties on $\mathcal{A}$. Consider an algebraic expression $g(x,a_1,\ldots,a_k)$ built up from the variable x and the elements $a_1,\ldots,a_k \in \mathcal{A}$. Consider the function

$$x \mapsto g(x,a_1,\ldots,a_k).$$

We can write this function using λ abstraction as

$$\lambda x g(x,a_1,\ldots,a_k).$$

We need to assume that the algebra $\mathcal{A}$ is such that there exists a function $\mathbf{exit}(\lambda x g(x,a_i))$ that satisfies the following for all b.

$$\mathbf{exit}(\lambda x g(x,a_1,\ldots,a_k)) * b = g(b,a_1,\ldots,a_k).$$

For example, if $\mathcal{A}$ is a Boolean algebra of sets and $*$ is union, then for any $g(x,a_1,\ldots,a_k)$ we have

$$g(x,a_1,\ldots,a_k) = x \cup a_1 \cup \ldots \cup a_k$$
$$\mathbf{exit}(\lambda x g(x,a_i)) = a_1 \cup \ldots \cup a_k$$

Hence

$$b \cup \mathbf{exit}(\lambda x g(x,a_i)) = b \cup a_1 \cup \ldots \cup a_k = g(b,a_1,\ldots,a_k).$$

In practice the function **exit** may not always be definable. So we can allow **exit** to be partial and say that we cannot exit, when **exit** is not defined.

We can now give the $\rightarrow$ Introduction and $\rightarrow$ elimination rules.

$\rightarrow E$ **rule**

$$\frac{\begin{array}{l}\alpha : A \\ \beta : A \rightarrow B \\ \varphi(\beta,\alpha)\end{array}}{\beta * \alpha : B}$$

$\rightarrow$ **Introduction rule**

Show $y : A \rightarrow B$ from box.

1. $x : A$ Assumption, x new atomic label variable
$\vdots$
$k : \gamma(x) : B$
get to $\gamma(x) : B$ using allowed proof rules $\rightarrow E, \rightarrow I$.
If $\mathbf{exit}(\lambda x \gamma(x))$is defined, then
exit with $y = \mathbf{exit}(\lambda x \gamma(x)) : A \rightarrow B$

Note that we can do the following continuation of the proof after we exit:

$$\frac{\begin{array}{l}\mathbf{exit}(\lambda x \gamma(x)) : A \rightarrow B \\ \alpha : A \\ \varphi(\beta, \mathbf{exit}(\lambda x \gamma(x)))\end{array}}{\mathbf{exit}(\lambda x \gamma(x)) * \alpha}$$

By our assumptions on **exit**, we get

$$\mathbf{exit}(\lambda x \gamma(x)) * \alpha = \gamma(\alpha).$$

Remark 3.5 (Comparison of LDS *with tableaux).* The perceptive reader might ask how does *LDS* compare with tableaux? Let us look at the database of Example 3.2.The databse Δ is:

$$\begin{array}{l} a_1 : B \rightarrow A \\ a_2 : A \rightarrow B \\ a_3 : A \end{array}$$

We want to derive B. Consider now E:

$$E = [(B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))].$$

Let us now do tableaux for the formula E. We write $\bot$ in front of a wff to say we want it false and $\top$ to say we want it true.

Let us do it by steps. We use the tableaux rule

$$\frac{\bot : X \rightarrow Y}{\top : X; \bot : Y}.$$

1. $\bot : [(B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))]$
2. $\top : (B \rightarrow A)$
   $\bot : (A \rightarrow B) \rightarrow (A \rightarrow B)$
3. $\top : B \rightarrow A$
   $\top : A \rightarrow B$
   $\bot : A \rightarrow B$

4. $\top : B \to A$
   $\top : A \to B$
   $\top : A$
   $\bot : B$

We see that the tableaux at item 4 is almost the same as the database, except that in the tableaux we have $\bot : B$ and in the database we have "we wish o prove B".This is a stylistic difference. More fundamental is the possibility of labeling and imposing metalevel restrictions on the tableaux, we get

$4'$. $\top : a_1 : B \to A$
   $\top : a_2 : A \to B$
   $\top : a_3 : A$
   $\bot : a_4 : B$

To appreciate the difference between Tableaux and *LDS*, let us ask: How can we read from $4'$ whether $A \to B$ is "used" twice? Such considerations make sense only in proof systems while tableaux is a labelled semantic system. They are different.

3.2.1 Labelled Modal Logics

An ordinary traditional Kripke model has the form $\mathbf{m} = (S, R, a, h)$, where S is the set of possible worlds, $R \subseteq S \times S$ is the binary accessibility relation and $a \in S$ is the actual world. h is the assignment of truth values to the atoms of the language; for each atomic $q, h(q) \subseteq S$.

Since we are going to adopt a new labelled view on satisfaction, let us write down the traditional definition in detail:

Note that (S, R, a) is mathematically just a directed graph with a distinguished point a on the graph. In the possible world approach we choose to view the points of the graph as worlds and the relation R as accessibility relation. In the labelled point of view we maintain the view of (S, R) as directed graph.

Definition 3.6 (Traditional satisfaction in a Kripke model). Let $\mathbf{m} = (S, R, a, h)$ be a Kripke model. We define the notion of

$$t \vDash_{\mathbf{m}} A$$

by induction on A:

- $t \vDash_{\mathbf{m}} q$ if $t \in h(q)$, for q atomic
- $t \vDash_{\mathbf{m}} A \wedge B$ iff $t \vDash_{\mathbf{m}} A$ and $t \vDash_{\mathbf{m}} B$
- $t \vDash_{\mathbf{m}} \neg A$ iff $t \nvDash_{\mathbf{m}} A$
- $t \vDash_{\mathbf{m}} \Diamond A$ iff for some s, tRs and $s \vDash_{\mathbf{m}} A$.
- $\mathbf{m} \vDash A$ iff $a \vDash_{\mathbf{m}} A$

The above clauses allow for non-normal modal logics, since satisfaction in the model is defined as satisfaction in the actual world.

The traditional point of view allows for correspondence between modal axioms and properties of R. For example the above semantics is complete for modal logic **K**. The axiom corresponding to R reflexive is $A \to \Diamond A$ (in the presence of the rule of necessitation) and the axiom corresponding to R transitive is $\Diamond\Diamond A \to \Diamond A$ (again in the presence of the rule of necessitation). It is relevant to mention that since validity is defined using the actual world, then the axiom corresponding to aRa (reflexivity only in the actual world and not in general for R) is $A \to \Diamond A$, but it must be added to a formulation of **K** without the rule of necessitation.

We now expand our point of view a bit more. A traditional modal theory Δ is a set of modal sentences, e.g. $\Delta = \{q, \Diamond q\}$. Δ is satisfied in a model $\mathbf{m} = (S, R, a, h)$, iff for all $A \in \Delta$ we have $a \vDash_{\mathbf{m}} A$. We write $\mathbf{m} \vDash \Delta$.

This point of view was generalised in Gabbay (1998, 1996) to that of a labelled theory. To form a labelled theory we add names for worlds say $\{\bar{a}, \bar{b}, \ldots\}$ to the syntax and allow for a labelled theory to use these names and have the form say, for example

$$\Delta = \{\bar{a} : A, \bar{b} : B, \bar{a}\bar{R}\bar{b}, \bar{a} \neq \bar{b}, \bar{a} \text{ actual}\}$$

where $\bar{R}$ names R in the syntax.

Here we are bringing some of the semantics into the syntax using labels. This is a big step, where we expand the formal language in the direction of some specific semantical interpretation of the language. In our case we want a model **m** with actual world $\bar{a} = a$ and another possible world $\bar{b} = b$ with $b \neq a$ such that aRb holds and $a \vDash A$ and $b \vDash B$. The view of Δ is that we are more demanding of the model and we want also that certain worlds relate to each other in certain ways and satisfy certain formulas as specified.[5]

Such theories Δ can be described as graphs. See Fig. 3.3. To show that this is a figure of a theory and not a model, we enclose it in a triangle. Models we enclose in big circles or elliptical closed curves.

The correlation $\bar{x} \mapsto x$ indicates that the syntactical name $\bar{x}$ in the theory Δ is mapped onto the element x in the model. Note that the model in Fig. 3.4 may contain more elements. In comparison, the theory Δ of Fig. 3.3 shows all the syntactical elements.

Consider now another ordinary traditional theory $\Theta = \{A \wedge \Diamond B\}$. In our notation, this is a one element graph $\{\bar{a} : A \wedge \Diamond B\}$. Figures 3.5 and 3.6 show two possible models for Θ. All we ask is that $a \vDash A \wedge \Diamond B$. So Fig. 3.5 does not show a model for Δ (of Fig. 3.3), while Fig. 3.6 could be a model for Δ if $a \neq b$.

Remark 3.7. Formally a theory Δ contains labelled formulas $t_i : A_i$, relations $\pm t_i \bar{R} t_j$ between labels and inequalities $t_i \neq t_j$ between labels and a possible indication which label is the actual world.

[5] When we put $\bar{x} : A$ in a theory Δ we mean that we demand that A holds in the world named by $\bar{x}$, i.e. $x \vDash A$.

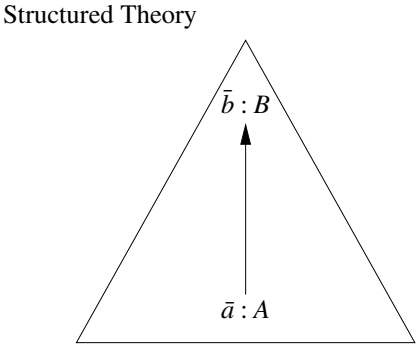


Figure $\bar{a} \neq \bar{b}$

Fig. 3.3 Dec08

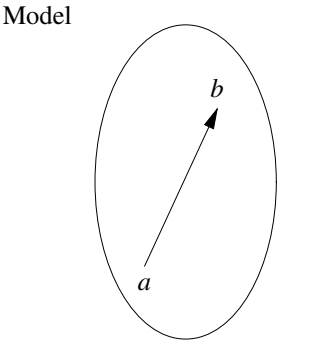


Figure: $a \neq b; a \vDash A; b \vDash B$

Fig. 3.4 Dec09

A model for Δ is a Kripke model $(S, R, a, h, \mathbf{g})$ where (S, R, a, h) is a model and $\mathbf{g}$ is a function on the labels t of Δ, such that $\mathbf{g}(t) \in S$ and the following holds:

1. If Δ says t is the actual world then $\mathbf{g}(t) = a$.
2. If Δ says $t_1 \bar{R} t_2$ then $\mathbf{g}(t_1) R\, \mathbf{g}(t_2)$.
3. If Δ says $t_1 \neq t_2$ then $\mathbf{g}(t_1) \neq \mathbf{g}(t_2)$.
4. If $t : A$ is in Δ then in the model (S, R, a, h) we have $\mathbf{g}(t) \vDash A$.

Figure 3.7 gives an example of a theory and its model. We have

$$\mathbf{g}(t_1) = a_1, \mathbf{g}(t_2) = a_2, \mathbf{g}(t_3) = a_3.$$

The theory says

$$t_1 \bar{R} t_2 \wedge t_2 \neq t_3 \wedge t_3 \bar{R} t_1$$

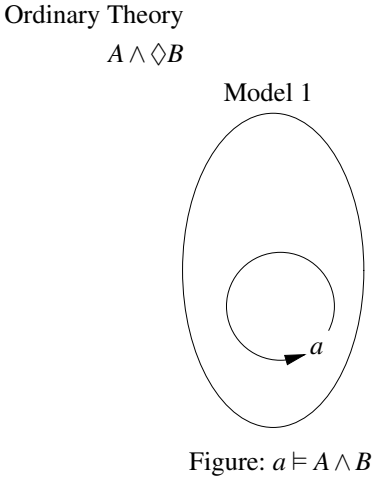


Fig. 3.5 Dec10

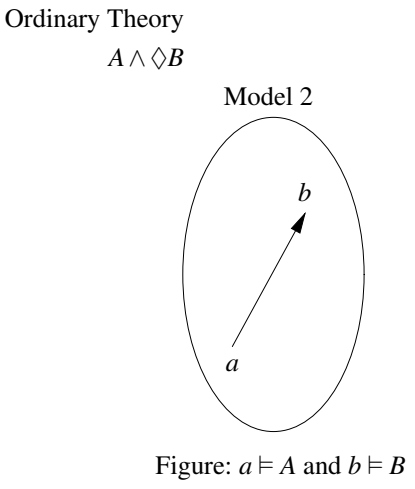
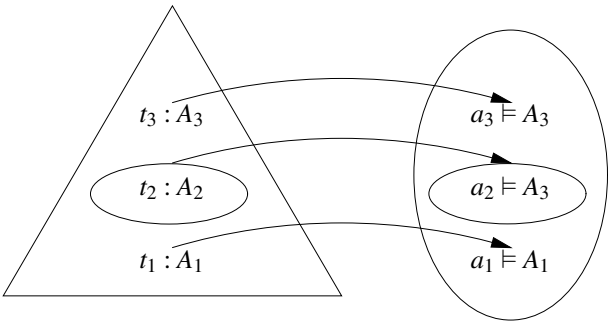


Fig. 3.6 Dec11



- $t_1 R t_2 \neq t_3 R t_1$ implies a_i related to each other, in the same way.

Fig. 3.7 Dec12

In the model, $\mathbf{g}(t_i)$ are R and $=$ related in the same way.

Thus it makes sense sometimes to give an 'almost' model to a theory and indicate how to deal with demands which are irrelevant to the evaluation.

Another possibility is the following theory:

$$\begin{aligned}\Delta = \{&(\bar{a} : A \text{ and } \bar{a}\bar{R}\bar{x}) \\ &\text{or } (\bar{a} : \neg A \text{ and } \neg\bar{a}\bar{R}\bar{x})\}. \\ = \{&\bar{a}\bar{R}\bar{x} \text{ iff } \bar{a} : A\}.\end{aligned}$$

We now have the notion of a structured labelled theory Δ for traditional Kripke semantics and we understand what it means for such a theory Δ to hold in a model $\mathbf{m} = (S, R, a, h)$. It means that there exists a function $\mathbf{g}$ from the labels of Δ into S, which does all that Δ says, as illustrated in Fig. 3.7, and Remark 3.7.

We now give some examples.

Example 3.8. This example shows the meta level–object level division in the case of modal logic. Modal logic has to do with possible worlds. We thus think of our basic database (or assumptions) as a finite set of information about possible worlds. This consists of two parts. The configuration part, the finite

configuration of possible worlds for the database, and the assumptions part which tells us what formulas hold in each world. The following is an example of a database:

	Assumptions	**Configuration**
(1)	$t : \Box\Box B$	$t < s$
(2)	$s : \Diamond(B \to C)$	

The conclusion to show (or query) is:

$$t : \Diamond\Diamond C.$$

The derivation is as follows:

3. From (2) create a new point r with $s < r$ and get $r : B \to C$.

We thus have

Assumptions	**Configuration**
(1), (2), (3)	$t < s < r$

4. From (1), since $t < s$ we get $s : \Box B$.
5. From (4) since $s < r$ we get $r : B$.
6. From (5) and (3) we get $r : C$.
7. From (6) since $s < r$ we get $s : \Diamond C$.
8. From (7) using $t < s$ we get $t : \Diamond\Diamond C$.

Discussion:
The object rules involved are:

$\Box E$ **Rule:**

$$\frac{t < s; t : \Box A}{s : A}$$

$\Diamond I$ **Rule:**

$$\frac{t < s, s : B}{t : \Diamond B}$$

$\Diamond E$ **Rule:**

$$\frac{t : \Diamond A}{\text{create a new point } s \text{ with } t < s \text{ and deduce } s : A}$$

Note that the above rules are not complete. We do not have rules for deriving, for example, $\Box A$. Also, the rules are all for intuitionistic modal logic.

The meta level consideration may be properties of $<$,
e.g. transitivity $t < s \wedge s < r \to t < r$ or
e.g. linearity: $t < s \vee t = s \vee s < t$ etc.

Example 3.9. The reader can already see the benefit of separating the metalevel (the handling of possible worlds i.e. labels) and the object-level (i.e. formulas) features. We can combine both the metalevel features of Examples 3.2 and 3.8 to create for example a modal relevance logic in a natural way. Each assumption has a relevance label as well as world label. Thus the proof of the previous example becomes the following:

	Assumptions	Configuration
(1)	$(a_1,t):\Box\Box B$	$t<s$
(2)	$(a_2,s):\Diamond(B\to C)$	

We proceed to create a new label r using $\Diamond E$ rule. The relevance label is carried over. We have $t<s<r$.

3. $(a_2,r):B\to C$

Using $\Box E$ rule with relevance label carried over, we have:

4. $(a_1,s):\Box B$
5. $(a_1,r):B$

Using modus ponens with relevance label updated

6. $(a_1,a_2,r):C$

Using $\Diamond I$ rule:

7. $(a_1,a_2,s):\Diamond C$
8. $(a_1,a_2,t):\Diamond\Diamond C$

(8) means that we got $t:\Diamond\Diamond C$ using both assumptions a_1 and a_2.

There are two serious problems in modal and temporal theorem proving. One is that of Skolem functions for $\exists x\Diamond A(x)$ and $\Diamond\exists xA(x)$ are not logically the same. If we skolemise we get $\Diamond A(c)$. Unfortunately it is not clear where c exists, in the current world $((\exists x=c)\Diamond A(x))$ or the possible world $(\Diamond(\exists x=c)A(x))$.

If we use labelled assumptions then, $t:\exists x\Diamond A(x)$ becomes $t:\Diamond A(c)$ and it is clear that c is introduced at t. In fact we shall write it as c^t.

On the other hand, the assumption $t:\Diamond\exists xA(x)$ will be used by the $\Diamond E$ rule to introduce a new point $s,t<s$ and conclude $s:\exists xA(x)$. We can further skolemise at s and get $s:A(c)$, with c introduced at s and write it as c^s. We thus need the mechanism of remembering or labelling constants as well, to indicate where they were first introduced, and we need rules to govern them. This is illustrated in Example 3.11.

Labelling systems for modal and temporal logics is studied in Gabbay (1991c, 1992b) and Gabbay (1996). See also references Basin et al. (2000), Gabbay (1993a), and Vigano (1999).

Example 3.10. The following example describes the logic of modal **S4** strict implication. In this logic the labels can be read either as relevance labels or as possible worlds. **S4** strict implication $A\to B$ can be understood as a temporal connective, as follows:

"$A\to B$ is true at world t iff for all future worlds s to t and for t itself we have that if A is true at s then B is true at s". Thus $A\to B$ reads "From now on, if A then B".[6]

Suppose we want to prove that $A\to B$ and $A\to(B\to C)$ imply $A\to C$. To show this we reason semantically and assume that at time t, the two assumptions are true. We want to show that $A\to C$ is also true at t. To prove that we take any future time s,

[6]Compare this with our insurance Example 3.1.

assume that A is true at s and show that C is also true at s. We thus have the following situation:

1. $t : A \to B$
2. $t : A \to (B \to C)$
3. show $t : A \to C$
   from box

3.1	Assume $s : A$ Show $s : C$ Since s is in the future of t, we get that at s, (1) and (2) are also true.
3.2	$s : A \to B$ from (1)
3.3	$s : A \to (B \to C)$ from (2) We now use modus ponens, because $X \to Y$ means "from now on, if X then Y"
3.4	$s : B$ from (3.1) and (3.2)
3.5	$s : B \to C$ from (3.2) and (3.3)
3.6	$s : C$ modus ponens from (3.4) and (3.5)

exit $t : A \to C$

Notice that any $t : D$ can be brought into (reiterated) the box as $s : D$, provided it has an implicational form, $D = D_1 \to D_2$. We can thus regard the labels above as simply naming assumptions (not as possible worlds) and the logic has the reiteration rule which says that only implications can be reiterated.

Let us add a further note to sharpen our understanding. Suppose $\to$ is read as a **K4** implication (i.e. transitivity without reflexivity). Then the above proof should fail. Indeed the corresponding restriction on modus ponens is that we do perform $X, X \to Y \vdash Y$ in a box, provided $X \to Y$ is a reiteration into the box and was not itself derived in that same box. This will block line (3.6).

Example 3.11. Another example has to do with the Barcan formula

This is a case of quantified modal logic. We need to organise how to deal with quantifiers in *LDS*The idea is that whenever we introduce a variable or a constant under a label we must label the variable/constant as well. Thus we have the rule:

$$\frac{t : \exists x A(x)}{t : A(c^t)} \qquad \frac{t : \forall x A(x)}{t : A(x^t)}$$

we also have $t : x^t$ and $t : c^t$ holding, where $t : y$ means that y *ressides* at t. A rule of the form

$$\frac{t : y}{s : y}$$

is called a visa rule, allowing for a term y residing at t also to reside at s. Thus we have the $\exists$ introduction rule as

$$\frac{t : A(y); t : y}{t : \exists y A(y)}.$$

and the universal generalisation rule:

$$\frac{t : A(x); t : x, x \text{ universal variable}}{t : \forall x A(x)}$$

To get the Barcan formula we need a visa rule

$$\frac{t : y; t < s}{s : y}$$

We can now prove this formula.

	Assumption	Configuration
(1)	$t : \forall x \Box A(x)$	$t < s$

We show

$$s : \forall x A(x)$$

We proceed intuitively

1. $t : \Box A(x)$ (stripping $\forall x$, remembering x is arbitrary), and $t : x$.
2. Since the configuration contains $s, t < s$ we get

$$s : A(x)$$

3. Since x is arbitrary we get by visa rule and $\Box$ rule:

$$s : \forall x A(x); s : x$$

The rule

$$\frac{t : \Box A(x), t < s}{s : A(x)}$$

is allowed because of the visa rule.

To have the above rule for arbitrary x is equivalent to adopting the Barcan formula axiom:

$$\forall x \Box A(x) \to \Box \forall x A(x)$$

To show $\Box \forall x A(x) \to \forall x \Box A(x)$, we need the visa rule:

$$\frac{t : y; s < t}{s : y}$$

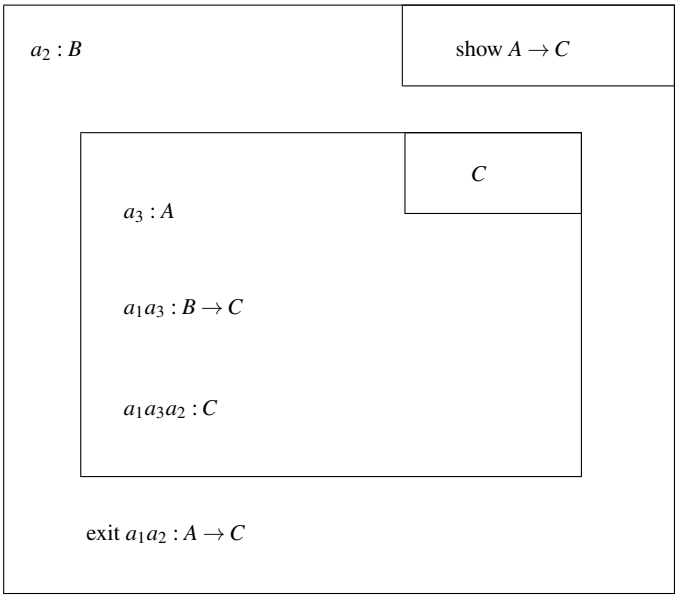


Fig. 3.8

The above are just a few examples for the scope we get using labels. The exact details and correspondences are worked out in our monograph (Gabbay 1996).

It is worth while to expand more on labelled modal logic, since modal logic is central to may applications in computer science, Language, Artificial Intelligence and Analytic Philosophy. So we have more in Appendix B.

3.2.2 Other Labelled Systems

The next batch of examples covers a variety of different labelled deductive systems.

Example 3.12 (Relevance Reasoning). The indices are $\alpha, \beta,$ and $\gamma = (\beta - \alpha)$. The reasoning structure is:
Assume $\alpha : A$
Show $\beta : B$
If $\beta \supseteq \alpha$ then exit with $(\beta - \alpha) : A \to B$.
To show $A \to (B \to C) \vdash B \to (A \to C)$

Assume

$$a_1 : A \to (B \to C)$$

we use the metabox to show $B \to (A \to C)$. See Fig. 3.8.

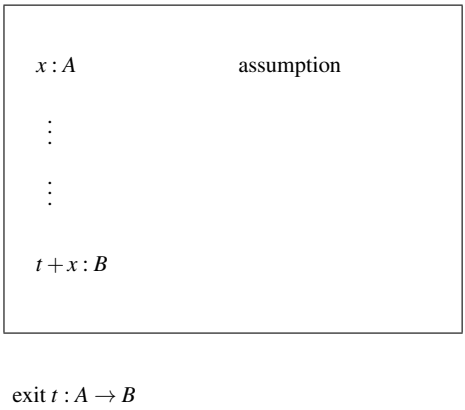


Fig. 3.9

Example 3.13 (Łukasiewicz many-valued logics). Consider Łukasiewicz infinite-valued logic, where the values are all real numbers or rationals in [0,1]. We designate 0 as **truth** and the truth table for implication is

$$x \to y = \max(0, y - x)$$

Here the language contains atoms and implication only, assignments h give values to atoms in [0,1], $h(q) \in [0,1]$ and h is extended to arbitrary formulas via the table for $\to$ above. Define the relation

$$A_1, \ldots, A_n \vdash B$$

to mean that for all $h, h(A_1) + \ldots + h(A_n) \geqq h(B)$, where + is numerical addition.

This logic can be regarded as a labelled deductive system, where the labels are values $t \in [0,1]$. $t : A$ means that $h(A) = t$, for a given background assignment h. The interesting part is that to show $t : A \to B$ (i.e. that $A \to B$ has value t) we assume $x : A$ (i.e. that A has value x) and then have to show that B has value $t + x$, i.e. show $t + x : B$.

This is according to the table of $\to$.

Thus Fig. 3.9 shows the deduction in box form.

This has the *same structure* as the case of relevance logic, where + was understood as concatenation.

A full study of many valued logics from the *LDS* point of view is given in Gabbay (1996).

Example 3.14 (Formulas as Types). Another instance of the natural use of labels is the Curry-Howard interpretation of formulas as types. This interpretation conforms

exactly to our framework. In fact, our framework gives the incentive to extend the formulas as types interpretation in a natural way to other logics, such as linear and relevance logics and surprisingly, also many valued logics, modal logics, and intermediate logics. A formula is considered as a type and its label is a *definable* λ-term of the same type. Given a system for defining λ-terms, the theorems of the logic are all those types which can be shown to be non-empty.

The basic propagation mechanism corresponding to modus ponens is:

$$\frac{t^A : A \qquad t^{A\to B} : A \to B}{t^{A\to B}(t^A) : B}$$

It is satisfied by *application*.

Thus if we read the + in $t^{A\to B} + t^A$ as application, we get the exact parallel to the general schema of propagation. Compare with relevance logic where + was concatenation, and with many valued logics where + was numerical addition!

To show $t : A \to B$ we assume $x : A$, with x arbitrary, i.e. start with a term x of type A, use the proof rules to get B. As we saw, applications of modus ponens generate more terms which contain x in them via application. If we accept that proofs generate functionals, then we get B with a label $y = t(x)$. Thus $t = \lambda x t(x)$. This again conforms with our general schema for $\to$.

In our paper (Gabbay and Queiroz 1992) and recently in our book (Quieroz et al. 2011) on the Curry-Howard interpretation we exploit this idea systematically. There are two mechanisms which allow us to restrict or expand our ability to define terms of any type. We can restrict λ-abstraction, (e.g. allow $\lambda x t(x)$ only if x actually occurs in t), this will give us logics weaker than intuitionistic logic, or we can increase our world of terms by requiring diagrams to be closed e.g., for any φ of classical logic such that

$$\vdash (A \to B) \to [\varphi(A) \to \varphi(B)]$$

in classical logic, we want the following diagram to be complete, i.e. for any term t there must exist a term t' (see Fig. 3.10).

Take for example the formula $A \to (B \to A)$ as type. We want to show a definable term of this type, we can try and use the standard proof (see Fig. 3.11), however, with the restriction on λ-abstraction which requires the abstracted variable to actually occur in the formula, we cannot exit the inner box. For details see Gabbay and Queiroz (1992).

Example 3.15 (Realisability Interpretation). The well known realisability interpretation for intuitionistic implication is another example of a functional interpretation for $\to$ which has the same universal *LDS* form. A notation for a recursive function $\{e\}$ realises an implication $A \to B$ iff for any n which realises A, $\{e\}(n)$ realises B. Thus

$$e : A \to B \text{ iff } \forall n[n : A \Rightarrow \{e\}(n) : B]$$

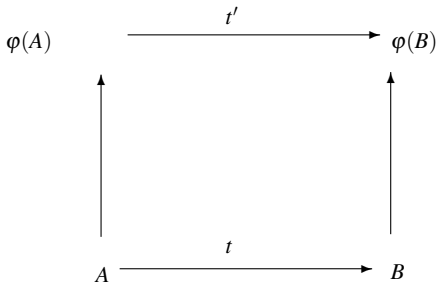


Fig. 3.10

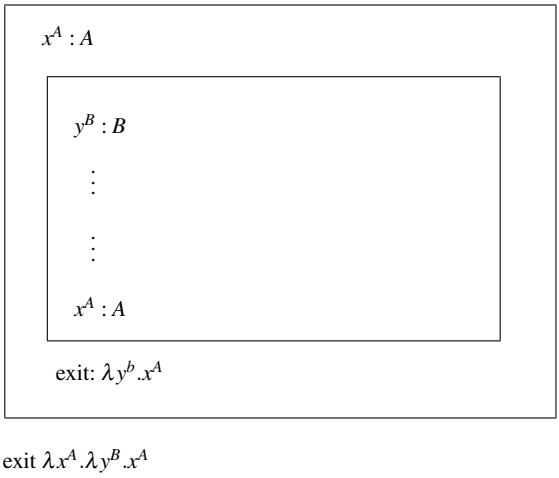


Fig. 3.11

It is an open problem to find an axiomatic description of the set of all wffs which are realisable.

Definition 3.16 (An algebraic *LDS* for implication and negation). Let **L** be a propositional language with $\rightarrow, \neg$ and atoms. Let $\mathcal{A}$ be an algebra of labels with relations $x < y$ for priority among labels, $\varphi(x,y)$ of compatibility among labels and functions, $\mathbf{f}(x,y)$ for propagating labels and $\uplus$ for aggregating labels.

Given two labelled formulas $t : A$ and $s : A \rightarrow B, \varphi(s,t)$ must hold in order to licence the modus ponens. If it does not hold, we cannot get B. If it does hold, we can get B but we must know what is the label of B. This is the job of the function $\mathbf{f}(s,t)$. The aggregation function tells us how different proofs of the same B with different labels can reinforce one another. Thus if we have $t : B$ and $s : B$ we can aggregate and get $t \uplus s : B$. See Example 3.20 for a very famous aggregation rule.

1. A *declarative unit* is a pair $t : A$, where A is a formula and t a term on the algebra of labels (built up from atomic labels and the functions f and $\uplus$).
2. A *database* is a set containing declarative units and formulae of the form $t_i < s_i$ and $\varphi(t_i, s_i)$ for some labels $t_1, \ldots, s_i, \ldots$.
3. The $\rightarrow$ elimination rule, *modus ponens*, has the form

$$\frac{t : A; s : A \rightarrow B; \varphi(s,t)}{\mathbf{f}(s,t) : B}$$

4. The $\Rightarrow$ introduction rule has the form
   - To introduce $t : A \rightarrow B$
     Assume $x : A$, for x arbitrary in the set $\{y \mid \varphi(t,y)\}$, and show $\mathbf{f}(t,x) : B$.
5. Negation rules have the form

$$\frac{t : B; s : \neg B}{r : C}$$

   We are not writing any specific rules because there are so many options for negation.
6. A family of *flattening rules* Flat of the form

$$\frac{t_1 : A, \ldots, t_k : A; s_1 : \neg A, \ldots, s_m : \neg A; y_i < y_j, i = 1,2,\ldots, j = 1,2,\ldots}{\gamma = \mathsf{Flat}(\{t_1, \ldots, t_k, s_1, \ldots, s_m\})}$$

   where γ is either 0 or 1 and is the result of applying the function Flat on the set containing t_i, s_j and where y_j, y_i range over $\{t_1, \ldots, t_k, s_1, \ldots, s_m\}$.[7]

   The meaning of γ is as follows. Since obviously we can prove both A and $\neg A$ with different labels, we need a flat decision on whether we take $A, (\gamma = 1)$ or $\neg A, (\gamma = 0)$.
7. *Aggregation rule*

$$\frac{t : A; s : A}{t \uplus s : A}$$

8. $\uplus$ is associative, commutative and $\mathbf{f}$ is distributive over $\uplus$.
9. A proof is a sequence of expressions which are of the form $t < s$, $\varphi(t,s)$ or $t : A$ such that each element of the sequence is either an assumption or is obtained from previous elements in the sequence by an elimination rule or is introduced by a subcomputation via the $\rightarrow$ introduction rule. Flattening rules are to be used last.

[7] Flat is a function defined on any set of labels and giving as value a new label. To understand this, recall another function on numbers which we may call **Sum**. It adds any set of numbers to give a new number: their sum!

3.3 Examples from Non-monotonic Logics

The examples in the previous section are from the area of monotonic reasoning. This section will give examples from non-monotonic reasoning. As we have already mentioned, we hope that the idea of *LDS* will unify these two areas.

Example 3.17 (Ordered Logic). An ordered logic database is a partially ordered set of local databases, each local database being a set of clauses. The following diagram (Fig. 3.12) describes an ordered logic database:

The local databases are labelled t_1, t_2, t_3, s_1, s_2 and $\varnothing$ and are partially ordered as in the figure.

To motivate such databases, consider an ordinary logic program $C_1 = \{p \leftarrow \neg q\}$. The computation of a logic program assumes that, since q is not a head of any clause, $\neg q$ is part of the data, (this is the *closed world assumption*). Suppose we relinquish this principle and adopt the principle of asking an *advisor* what to do with $\neg q$. The advisor might say that $\neg q$ succeeds or might say that $\neg q$ fails. The advisor might have his own program to consult. If his program is C_2, he might run the goal q (or $\neg q$), look at what he gets and then advise. To make the situation symmetrical and general we must allow for Horn programs to have rules with both q and $\neg q$ (i.e. literals) in heads and bodies and have any number of negotiating advisors. Thus we can have $C_2 = \{\neg q\}, C_1 = \{q \leftarrow \neg q\}$ and C_1 depends on C_2. Ordered logic develops and studies various aspects of such an advisor system which is modelled as a partially ordered set of theories. Such a logic is useful, e.g. for multi-expert systems where we want to represent the knowledge of several experts in a single system. Experts may then be ordered according to an "advisory" or a relative preference relation.

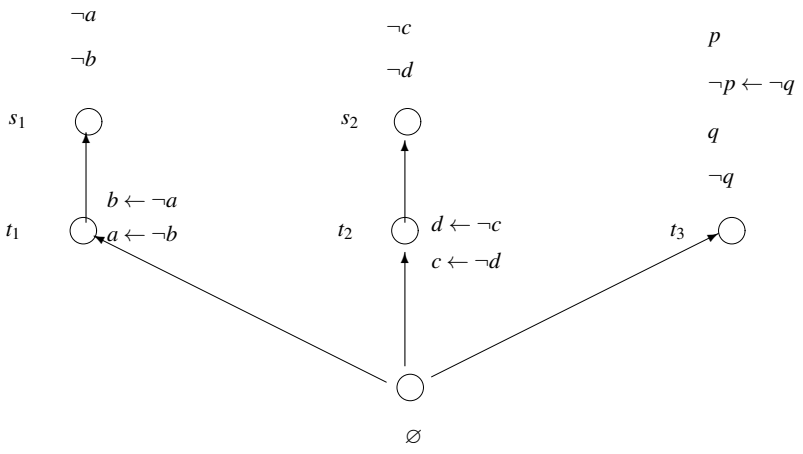


Fig. 3.12

A problem to consider is what happens when we have several advisors that are in conflict. For example, C_1 depends on C_2 and C_1 depends on C_3. The two advisers, C_2 and C_3, may be in conflict. One may advise $\neg q$, the other q. How to decide? There are several options:

1. We can accept q if all advisors say "yes" to q.
2. We can accept q if at least one advisor says "yes" to q.
3. We can apply some non-monotonic or probabilistic mechanism to decide.

If we choose options (1) or (2) we are essentially in modal logic. To have a node t and to have $?q$ refer to advisors $t_1,\ldots,t_n$ with $t < t_i, i = 1,\ldots,n$ is like considering $?\Box q$ at t in modal logic with $t_1,\ldots,t_n$ possible worlds in option 1 and like considering $\Diamond q$ at t in option (2). Option (3) is more general, and here an *LDS* approach is most useful. We see from this advisors examples an application area where the labels arise naturally and usefully. The area of ordered logic is surveyed in Vermeir and Laenens (1990).

Example 3.18 (Defeasible Logic). This important approach to non-monotonic reasoning was introduced by Nute (1994). The idea is that rules can prove either an atom q or its negation $\neg q$. If two rules are in conflict, one proving q and one proving $\neg q$, the deduction that is stronger is from a rule whose antecedent is logically more specific. Thus the database:

$$\begin{aligned} t_1 &: \quad \text{Bird}\,(x) \rightarrow \text{Fly}\,(x) \\ t_2 &: \quad \text{Big}\,(x) \wedge \text{Bird}\,(x) \rightarrow \neg\,\text{Fly}\,(x) \\ t_3 &: \quad \text{Big}\,(a) \\ t_4 &: \quad \text{Bird}\,(a) \end{aligned}$$

$$\begin{aligned} t_1 &< t_2 \\ t_3 & \\ t_4 & \end{aligned}$$

can prove:

$$\begin{aligned} t_2 t_3 t_4 &: \quad \neg\text{Fly}(a) \\ t_1 t_4 &: \quad \text{Fly}(a) \end{aligned}$$

The database will entail $\neg$ Fly (a) because the second rule is more specific.

As an *LDS* system the labelling of rules in a database Δ is very simple. We label a rule by its antecedent. The ordering of the labels is done by logical strength relative to some background theory Θ (which can be a subtheory of Δ of some form). Deduction pays attention to strength of labels.

Example 3.19 (Fallacies). The reader should note that our point of view and the use of labels is genuinely more general and is capable of yielding more. We describe an unexpected application of our view. There is a serious, well-motivated and well-organised community, the informal logic and argumentation community, studying the nature of human reasoning and argumentation in general and attempting to

foundationally explain the role of the fallacies in human arguments. Fallacies are argument structures which appear to be correct and convincing, but are actually wrong. Many of them can be effectively used in some situations, but not in others. Any account of real life human practical reasoning must give account of the fallacies. In Hamblin Words (Hamblin 1970), a fallacy is an argument that 'seems to be valid but is not so'.

The handling of the fallacies in the traditional literature is divergent between two extremes.

There are those who reject the fallacies as not having any logical value (see Lambert and Ulrich 1980) and there are those who try to see some logic in them. Among the latter are John Woods and Douglas Walton. They believe that the traditional fallacies can be explained within the framework of other logics, such as inductive logics, non-classical logics, logics of plausible reasoning, relevance logics and more. The Woods–Walton approach, see Woods and Walton (1989), Woods (1988), Walton (1990), is successful in many cases in showing and explaining how some fallacies are really not fallacies. However the Woods–Walton approach was in principle criticised by F. H. Van Emerson and R. Grootendorst (1992), who point out that this approach, although successful in many cases, creates new and serious problems. Van Emeren and Grootendorst, justly point out that every fallacy, in this approach needs, so to speak, its own logic. In van Eemeren and Grootendorst (1992, p. 103) they say

> 'For practical purposes this approach is not very realistic. In order to be able to carry out the analyses, a considerable amount of logical knowledge is required. There are also some theoretical disadvantages inherent in this approach. By relying on so many logical systems, one only gets fragmentary description of the various fallacies, and no overall picture of the domain of the fallacies as a whole. Ideally, one unified theory that is capable of dealing with all the different phenomena, is to be preferred.'

We agree with both Van Emeren–Grootendorst and with Woods–Walton. There is indeed a possible candidate for a unifying logic in which suitable theories for practical reasoning and the fallacies can be formulated.

It is the framework of Labelled Deductive Systems.

This example is a preliminary study at classifying and explaining some of the fallacies in *LDS*.

Here we quote Douglas Walton's words (Walton 1990, p. 353)

> '... until we have a clearer definition of theoretical reasoning, it is not possible to refute the argument that there is one underlying kind of reasoning that has two uses—practical problem solving and theoretical problem solving ...'

Well known among the fallacies is the fallacy Ad Hominem, the fallacy of attacking not the argument but the person presenting it. This kind of reasoning is sometimes acceptable and sometimes not. It is generally considered non-logical, although admittedly extensively used by the human practical reasoner. In our framework, this fallacy has a natural place.

Consider the notion of a database Δ. This is a structure of declarative units of the form $t : A$, where t is the label and A the formula. The label t annotates A. Suppose

the annotation indicates the priority of the formula A and that in an external ordering $<$ gives the relative strength of the priorities. Thus a priority database can be for example

$$\{t : A, s : B, t < s\}$$

t and s can be numbers of algebraic terms and $t < s$ indicates that B has a higher priority than A. This priority can be used in derivation. For example, in the presence of $A \to \neg C, B \to C$ of equal priority, C will be derived.

The data items A and B are formulas of the logic $\mathbf{L}_1$, which is applied to some application area. In many areas it is quite reasonable to have the labels themselves be formulas α, β of another language and logic $\mathbf{L}_2$, describing the origin and nature of the data items, A, B. Some reasoning in $\mathbf{L}_2$ may be available to determine the priority (if any) of α and β. A formula $\Psi(\alpha, \beta)$ and a base theory Θ (possibly dependent on Δ) of $\mathbf{L}_2$ may be used for this purpose, i.e. we have:

$$\alpha \leq \beta \text{ iff } \Theta \vdash_2 \Psi(\alpha, \beta).$$

The simplest condition (in case $\mathbf{L}_2$ has some form of implication) is

$$\alpha \leq \beta \text{ iff } \Theta \vdash_2 \beta \to \alpha.$$

Note that our labels are wffs α of $\mathbf{L}_2$ labelling wffs A of $\mathbf{L}_1$ and the base theory Θ determines the priorities of labels. We now explain the logical force of the fallacy by an example. Suppose we are faced with the following deduction.

$$\begin{array}{l} \alpha : A \to \neg C \\ \beta : B \to C \\ \gamma : A \\ \gamma : B \\ \Theta \vdash_2 \beta \to \alpha \end{array}$$

We must conclude C, because β has higher priority than α. To counter this argument, we may either prove $\neg C$ from additional data or we may attack the source of information, i.e. add Θ_0 to Θ or try and show that $\Theta \cup \Theta_0 \nvdash_2 \beta \to \alpha$?, (Note that $\mathbf{L}_2$ reasoning is also non-monotonic!). This move appears to us as attacking, not the argument, but its source. However, in the correct context (priority logic) it is a correct move. Other fallacies which are explainable in this framework are Ad Verecundiam, appeal to unsuitable authority, where the labelling is incorrect and fallacies of irrelevance. A systematic study of the fallacies in our context will (hopefully) be done elsewhere.

To make the above database more concrete consider the following scenario. A man is imprisoned for fraud for a long period of time. During that period, medical evidence emerges that the prisoner has terminal cancer. The question is whether to release him from jail. One legal argument supports an early release. The problem seems to be that the prisoner made some threats during the trial and a social and

psycological report cannot exclude the possibility that the prisoner might use his remaining free days for revenge. Our database now reads

$m : B$	medical file m supporting the statement that the prisoner has cancer
$p : A$	social workers report supporting the statement that the prisoner is seeking revenge
$\alpha : A \to \neg C$	legal precedents α supporting the rule that in case of possible revenge the prisoner should not be released
$\beta : B \to C$	legal reasoning β supporting that in case of cancer the prisoner should be released
$p < m$	medical files are stronger than 'psychological' files'

From the above data we can conclude

$$\beta * m : C$$

and

$$\alpha * p : \neg C$$

Since both β and m have higher priority, C will follow by the *flattening* process.

If we want to change the conclusion (to get $\neg C$), we must either attack the medical file m, discrediting the medical evidence or boost up the credibility of the psychological report.

Example 3.20 (Dempster–Shafer rule). The present example presents a very well known rule of aggregation, the Dempster–Shafer rule. Our exposition relies on Ng and Subrahmanian (1994).

The algebra $\mathcal{A}$ we are dealing with is the set of all subintervals of the unit interval [0,1]. The Dempster–Shafer addition on these intervals is defined by

$$[a,b] \oplus [c,d] = \left[\frac{a \cdot d + b \cdot c - a \cdot c}{1-k}\,,\quad \frac{b \cdot d}{1-k}\right]$$

where $k = a \cdot (1-d) + c \cdot (1-b)$, where '$\cdot$', '$+$', '$-$' are the usual arithmetical operations. The compatibility condition required on a, b, c, d is

$$\mathcal{F}([a,b],[c,d]) \equiv k \neq 1.$$

The operation $\oplus$ is commutative and associative. Let $\mathbf{e} = [0,1]$.

The following also holds:

- $[a,b] \oplus \mathbf{e} = [a,b]$
- For $[a,b] \neq [1,1]$ we have $[a,b] \oplus [0,0] = [0,0]$
- For $[a,b] \neq [0,0]$ we have $[a,b] \oplus [1,1] = [1,1]$

- $[a,b] \oplus [c,d] = \varnothing$ iff either $[a,b] = [0,0]$ and $[c,d] = [1,1]$ or $[a,b] = [1,1]$ and $[c,d] = [0,0]$.

In this algebra, we understand the declarative unit $[a,b] : A$ as saying that the probability of the event represented by A lies in the interval $[a,b]$. We have, of course

$$\frac{[a,b] : A \to B; [c,d] : A}{[a,b] \oplus [c,d] : B},$$

provided $\mathcal{F}([a,b],[c,d])$ holds.

It is also possible to move to a higher language and write clauses of the form

$$t : (t_1 : A_1) \to ((t_2 : A_2) \to (t_3 : A_3))$$

which is more like the way clauses are used in traditional Dempster–Shafer applications.

3.4 Conclusion and Further Reading

We started this chapter by mentioning the need, arising from applications, for a general unifying theory of "what is a logical system". The theory of *LDS* arose as a partial answer to this question. The perceptive reader would surely like to know in concluding this chapter the answer to the following two questions:

1. What is our current (2011) view of "what is a logical system"?
2. Can we give some sample diverse applications of *LDS* in some interesting "hot" areas?

The remainder of this section answers question 1 and the appendices answer question 2.

Logic is widely applied in computer science and artificial intelligence, in philosophy, linguistics, argumentation, psychology, decision theory, theology and Law. The reader can see the discussion in the editorial for this Handbook. The needs of the application areas in computing are different from those in mathematics and philosophy. In response to urgent needs of many application areas, intensive research has been directed in the area of non- classical and non-monotonic logic. New logics have been developed and studied. Certain logical features, which have not received extensive attention in the pure logic community, are repeatedly being called upon in computational applications. The following list identifies some of the changes

The traditional notions of logic rely on the following building blocks.

A1. a. Formulas are built up from atoms using connectives, variables and quantifiers.

b. Atoms have no internal structure.
c. Our view is that atoms can be complex objects such as networks of nodes with relations and structure among them and that the logical connectives correspond to operations on networks to get new networks. The notion of substitution of a formula A for an atom q in $B(q)$ to form $B(q/A)$ corresponds to the notion of fibring of networks. See references (Gabbay and Williamson 2004; Gabbay and d'Avila Garcez 2004; Gabbay 2009; Carnielli et al. 2007).

A2. Traditionally a logic is recognised by its theorems (of the form $\vdash B$) or perhaps by its consequence relation (of the form $A \vdash B$).

Our view is that an essential part of a logic may also include other mechanisms and the manner in which they interact.

A3. A semantic interpretation is traditionally desirable (provided in set theory) and a completeness theorem is expected. There is a clear distinction between syntax (theorems) and semantics.

We allow for the semantics to be brought into the syntax via labelling and look at the notion of interpretation and translation as 'semantics'. Furthermore, we allow the semantics to be reactive and change during the process of the semantic evaluation in response to the evaluation steps.

A4. To the extent that proof theory is provided it is considered as a syntactical/algorithmic means of presenting the consequence relation. So, for example, classical logic presented via tableaux is generally considered to be the same logic as when presented using resolution. We propose that we consider them as two different logics.

A5. The traditional data structure of theories of the logic are usually sets of formulas, maybe lists of formulas and maybe lists of lists. This is as far as they go. Certainly not a general structure of any kind, with an entire mathematical algebraic procedure for manipulating it.

Not only do we accept general structures for data but also algorithms for fetching more data are considered also as part of the data. The new aspect of allowing algorithms to be data items in the database is that the algorithm may give different results depending on when in the proof process it is invoked! See also A10.

A6. In traditional logics any data item (formula) can be put into the database. There is no elaborate procedure for deciding whether to allow a data item A to be input into a database Δ. A data item may be rejected if it is inconsistent with existing data. However, there is nothing like the considerations of admissibility of evidence into the database that we have, for example, in law.

A7. The entire presentation of the logic is supposed to be in the object level. Some meta-level considerations are studied, mainly in connection with self-reference. There is no elaborate meta-level hierarchy and a variety of levels available as an essential part of the logic and certainly no serious body of rules is considered for moving between levels.

A8. In traditional logic, time does not enter into the picture at all. The logic is static. It does not depend on actions, or evolve or change. There is no notion

of input and output. Logic is not a process but a fixed consequence relation to be studied by various means. Our concept of a logic involves action and change. The most simple model interprets $\Delta \vdash A$ as having a sequence of actions which modify Δ to Δ' and $\Delta' \vdash A$.

A9. Traditionally, a formula can either be a theorem of the logic or not. Even in many valued logics the body of theorems is defined using the values but the logic is viewed as defined by the body of its theorems. There is no systematic notion of several categories of strong/weak theorems and how to move from one category to another.

A10. Various mechanisms such as abduction, revision, etc, are considered meta-level to traditional logic and not part of the logic itself.

For example, items of data are not restricted to just formulas but can also be mechanisms which can be activated to get more data.

A11. There is no recognition for methodologies for allowing logics to interact with one another as part of the general notion of a logic. There is no realisation that it may be the case that a certain logic can be presented only as part of a larger family of interacting logics.

A12. There is no recognition that the process of substitution is an important proof theoretical step (similar to e.g. modus ponens) and that all variables should come annotated with a range of allowable and forbidden substitutions. Changing such a range is a legitimate and important proof step. This is important in modelling language.

A13. Two features in logic seem to be of crucial importance to the needs of computer science and stand in need of further study. These are:

1. The metalevel features of logical systems
2. The "logic" of Skolem functions and unification

The meta-language properties of logical systems are usually hidden in the object language. Either in the proof theory or via some higher-order or many-sorted devices. The logic of Skolem functions is non-existent.

A14. There is a need to integrate logic with network reasoning. The problem is discussed in Gabbay (2009) and Gabbay (2010). A solution can be found in Gabbay (2011, 2012) and Gabbay (2011c).

To conclude, the traditional presentation of classical and non-classical logics is not conducive to bringing out and developing the features needed for human oriented applications. The very concept of what is a logical system seems to be in need of revision and clarification. A closer examination of classical and non-classical logics reveals the possibility of introducing a new approach to logic; the discipline of *Labelled Deductive Systems* (*LDS*) which, I believe, will not only be ideal for computer science applications but will also serve, I hope, as a new unifying logical framework of value to logic itself. What seem to be isolated local features of some known logics turn out to be, in my view, manifestations of more general logical phenomena of interest to the future development of logic itself.

Semantics for *LDS* logics is presented in my book on Fibring Logics (Gabbay 1998).

LDS is part of a more general view of logic. This view is discussed elsewhere (Gabbay 1996, 1991b, 1993b), however in brief, we claim the following. The new concept of a logical system is that of a *network* of *LDS* systems which has mechanisms for *communication* (through the labels, which code meta information) and *evolution* or change.

Evaluation is a general concept which can embrace updating, abduction, consistency maintenance, action and planning. The above statement of position is vague but it does imply that we believe that notions like abduction and updating are logical notions of equal standing to those of provability. See Gabbay and Woods (2003/2005).

Appendices

Appendix A. Case Study Application of *LDS* to Law: Hearsay Case, *Myers v DPP*

To show you the power of *LDS* we quote, model and analyse a famous legal case from thee UK, concerning the Theory of Evidence. It is concerned with the admissibility of evidence. In logical terms it means the following.

Given a structured labelled database Δ, we want to input into it another item of data A. We need to say with what label the item goes into the database and where in the structure of the database it is to reside. We might even reject the item if it seems to be problematic. The Theory of Evidence in Law deals extensively with such items. When to accept them, how to use them, etc., etc. One such example is if the item A to be admitted comes from Hearsay.

This case appears in every legal textbook on Evidence. We begin by quoting from Allen (2001, p. 133).

A good statement of the hearsay rule was given originally in *Cross on Evidence*, Cross (1999).

> An assertion other than one made by a person while giving oral evidence in the proceedings is inadmissible as evidence of any fact asserted.

Allen continued on page 135:

> Hearsay law has been described as 'exceptionally complex and difficult to interpret' (RCCJ 1993). What we need is a method of approach to the subject which will enable us to understand why some cases were decided as they were and why others are open to criticism. Above all, we need a technique [our comment: i.e. logic] for thinking about hearsay, ...

We now examine a key case, which seems to be quoted in every textbook on Evidence (and hearsay). This is a case of *written statements*, which may fall under hearsay law.

We quote two descriptions of this case, one from Keane (2000) and one from Uglow (1997), and then we model the arguments as quoted in Uglow (1997).

We begin with Keane (2000, pp. 250–252)

> *(b) Written statements*
> The leading case on written hearsay is *Myers v DPP* ([1965] AC 1001). The appellant was convicted of offences relating to the theft of motor cars. He would buy a wrecked car, steal a car resembling it, disguise the stolen car so that it corresponded with the particulars of the wrecked car as noted in its log book, and then sell the stolen car with the log book of the wrecked one. The prosecution case involved proving that the disguised cars were stolen by reference to the cylinder-block numbers indelibly stamped on their engines. In the case of some cars, therefore, they sought to adduce evidence derived from records kept by a motor manufacturer. An officer in charge of these records was called to produce microfilms which were prepared from cards filled in by workmen on the assembly line and which contained the cylinder-block numbers of the cars manufactured. The Court of Criminal Appeal held that the trial judge had properly allowed the evidence to be admitted because of the circumstances in which the record was maintained and the inherent probability that it was correct rather than incorrect. The House of Lords held that the records constituted inadmissible hearsay evidence. The entries on the cards and contained in the microfilms were out-of-court assertions by unidentifiable workmen that certain cars bore certain cylinder-block numbers. The officer called could not prove that the records were correct and that the numbers they contained were in fact the numbers on the cars in question. Their Lordships, however, were divided as to whether the evidence should be admitted by the creation of a new exception to the hearsay rule.[8] Lords Pearce and Donovan were in favour of such a course, but the majority, comprising Lords Reid, Morris and Hodson, declined to do so, being of the opinion that it was for the legislature and not the judiciary to add to the classes of admissible hearsay.[9] It was argued before the House that the trial judge has a discretion to admit a record in a particular case if satisfied that it is trustworthy and that justice requires its admission. Lord Reid, while acknowledging that the hearsay rule was 'absurdly technical', held that 'no matter how cogent particular evidence may seem to be, unless it comes within a class which is admissible, it is excluded . . . '
>
> The actual decision in *Myers v DPP* was reversed by the Criminal Evidence Act 1965, which provided for the admissibility of certain hearsay statements contained in trade or business records. Although the 1965 Act was repealed by the Police and Criminal Evidence Act 1984, ss 23 and 24 of the Criminal Justice Act 1988 are wider in scope than the provisions of the 1965 Act and provide for the admissibility of first-hand hearsay statements in documents generally as well as hearsay statements contained in documents created or received by a person in the course of, inter alia, a trade or business. The principles enunciated in *Myers v DPP*, however, remain of importance in relation to hearsay statements falling outside the statutory exceptions. Over 25 years later, another majority of the House of Lords, in *R v Kearley*,[10] although of the opinion that there may be a case for a general relaxation of the hearsay rule, affirmed the majority view in *Myers v DPP* that the only satisfactory solution is legislation following on a wide survey of the whole field.

[8]The Lords were unanimous in dismissing the appeal on the grounds that the other evidence of guilt being overwhelming, there had been no substantial miscarriage of justice.

[9]The minority view, that it was within the provenance of the judiciary to restate the exceptions to the hearsay rule, was adopted by the Supreme Court of Canada in *Ares v Venner* [1970] SCR 608. See also per Lord Griffiths in *R v Kearley* [1992] 2 All ER 345, HL at 348.

[10][1992] 2 All ER 345, HL, per Lords Bridge, Ackner and Oliver at 360–361, 366 and 382–383 respectively.

> *Patel v Comptroller of Customs*[11] also illustrates the application of the hearsay rule to written statements. The appellant was convicted of making a false declaration in an import entry form concerning certain bags of seed. Evidence was admitted that the bags of seed bore the words 'Produce of Morocco'. The Privy Council held that the evidence was inadmissible hearsay and advised that the conviction be quashed. The decision may be usefully compared with that in *R v Lydon*.[12] The appellant, Sean Lydon, was convicted of robbery. His defence was one of alibi. About one mile from the scene of the robbery, on the verge of the road which the getaway car had followed, were found a gun and, nearby, two pieces of rolled paper on which someone had written 'Sean rules' and 'Sean rules 85'. Ink of similar appearance and composition to that on the paper was found on the gun barrel. The Court of Appeal held that evidence relating to the pieces of paper had been properly admitted as circumstantial evidence: if the jury were satisfied that the gun was used in the robbery and that the pieces of paper were linked to the gun, the references to Sean could be a fact which would fit in with the appellant having committed the offence. The references were not hearsay because they involved no assertion as to the truth of the contents of the pieces of paper: they were not tendered to show that Sean ruled anything.[13]

If we go 480 pages into Steven Uglow's book (Uglow 1997), we find his account of the same case.

> *written statements*: the classic case here is *Myers v DPP* ([1964] 2 All E.R. 877) where the defendant bought wrecked cars for their registration certificates. He would then steal a similar car and alter it to fit the details in the document. He would sell the disguised stolen car along with the genuine log book of the wrecked car. The prosecution sought to show that the cars and registration documents did not match up by reference to the engine block numbers and introduced microfilm evidence kept by the manufacturer, showing that this block number did not belong in a car of this registration date. The microfilm was prepared from cards which were themselves prepared by workers on the assembly line. Lord Reid in the House of Lords held that the microfilm was inadmissible since it contained the out-of-court assertions by unidentified workers.

The labelled structure of the above is as follows.

Let

- $t : C$ The numbers assigned to the cars by the manufacturers are $x_1, x_2, \ldots$
- $t' : C'$ The numbers in the cars' logbook are $y_1, y_2, \ldots$.

---

[11] [1966] AC 356, PC. See also *R v Sealby* [1965] 1 All ER 701 and *R v Brown* [1991] Crim LR835, CA (evidence of a name on an appliance inadmissible to establish its ownership); and cf *R v Rice* [1963] 1 QB 857, below.

[12] [1987] Crim LR 407, CA.

[13] See also *R v McIntosh* [1992] Crim LR 651, CA (calculations as to the purchase and sale prices of 12 oz of an unnamed commodity, not in M's handwriting but found concealed in the chimney of a house where he had been living, admissible as circumstantial evidence tending to connect him with drug-related offences); and cf *R v Horne* [1992] Crim LR 304, CA (documents of unknown authorship, referring to H, containing calculations possibly relating to the cost of importing drugs, and found in the flat of a co-accused to which H was supposed to deliver the drugs, inadmissible against H). *R v McIntosh* was applied in *Roberts v DP* [1994] Crim LR 926, DC: documents found at R's offices and home, including repair and gas bills and other accounts relating to certain premises, were admissible as circumstantial evidence linking R with those premises, on charges of assisting in the management of a brothel and running a massage parlour without a licence.

If $x_i \neq y_i$, then we get:

- $t + t' : C''$ = the numbers on the cars and numbers on the registration documents do not match

where

- t = description of how the microfilm supporting C was obtained and compiled.
- t' = the cars' logbooks.

The candidate item of data for admissibility is

- $t : C$.

The following passage is Lord Reid's argument that $t : C$ should be inadmissible, i.e. Lord Reid wants to argue that t should also contain the phrase "do not use me".

This is done in the logic of the labels. In other words, Lord Reid's argument has to do with the data inside t.

Here is Lord Reid's argument (technically it is part of t). It also quotes the arguments given in favour of admitting $t : C$.

> ***Myers v DPP*** **[1964] 2 All E.R. 877 at 886b–887h, *per* Lord Reid**
>
> It is not disputed before your Lordships that to admit these records is to admit hearsay. They only tend to prove that a particular car bore a particular number when it was assembled if the jury were entitled to infer that the entries were accurate, at least in the main; and the entries on the cards were assertions by the unidentifiable men who made them that they had entered numbers which they had seen on the cars. Counsel for the respondents were unable to adduce any reported case or any textbook as direct authority for their submission. Only four reasons for their submission were put forward. It was said that evidence of this kind is in practice admitted at least at the Central Criminal Court. Then it was argued that a judge has a discretion to admit such evidence. Then the reasons given in the Court of Criminal Appeal were relied on. And lastly it was said with truth that common sense rebels against the rejection of this evidence.
>
> At the trial counsel for the prosecution sought to support the existing practice of admitting such records, if produced by the persons in charge of them, by arguing that they were not adduced to prove the truth of the recorded particulars but only to prove that they were records kept in the normal course of business. Counsel for the accused then asked the very pertinent question — if they were not intended to prove the truth of the entries, what were they intended to prove? I ask what the jury would infer from them: obviously that they were probably true records. If they were not capable of supporting an inference that they were probably true records, then I do not see what probative value they could have, and their admission was bound to mislead the jury.
>
> The first reason given by the Court of Criminal Appeal for sustaining the admission of the records was that, although the records might not be evidence standing by themselves, they could be used to corroborate the evidence of other witnesses.[14] I regret to say that I have great difficulty in understanding that … Unless the jury were entitled to regard them, I

[14]This is our footnote. "corroborate evidence of other witnesses" means in our *LDS* language "help with the flattening process".

can see no reason why they should only become admissible evidence after some witnesses have identified the cars for different reasons ... [15]

At the end of their judgement, the Court of Criminal Appeal gave a different reason. 'In our view the admission of such evidence does not infringe the hearsay rule because its probative value does not depend upon the credit of an unidentified person but rather on the circumstances in which the record is maintained and the inherent probability that it will be correct rather than incorrect.' That, if I may say so, is undeniable as a matter of common sense. But can it be reconciled with the existing law? I need not discuss the question on general lines because I think that this ground is quite inconsistent with the established rule regarding public records. Public records are prima facie evidence of the fact which they contain but it is quite clear that a record is not a public record within the scope of that rule unless it is open to inspection by at least a section of the public. Unless we are to alter that rule how can we possibly say that a private record not open to public inspection can be prima facie evidence of the truth of its contents? I would agree that it is quite unreasonable to refuse to accept as prima facie evidence a record obviously well kept by public officers and proved never to have been discovered to contain a wrong entry though frequently consulted by officials, merely because it is not open to inspection. But that is settled law. This seems to me to be a good example of the wide repercussions which would follow if we accepted the judgement of the Court of Criminal Appeal. I must therefore regretfully decline to accept this reason as correct in law.

In argument, the Solicitor-General maintained that, although the general rule may be against the admission of private records to prove the truth of entries in them, the trial judge has a discretion to admit a record in a particular case if satisfied that it is trustworthy and that justice requires its admission. That appears to me to be contrary to the whole framework of the existing law. It is true that a judge has a discretion to exclude legally admissible evidence if justice so requires, but it is a very different thing to say that he has a discretion to admit legally inadmissible evidence. The whole development of the exceptions to the hearsay rule is based on the determination of certain classes of evidence as admissible or inadmissible and not on the apparent credibility of particular evidence tendered. No matter how cogent particular evidence may seem to be, unless it comes within a class which is admissible, it is excluded. Half a dozen witnesses may offer to prove that they heard two men of high character who cannot now be found discuss in detail the fact now in issue and agree on a credible account of it, but that evidence would not be admitted although it might be by far the best evidence available.

It was admitted in argument before your Lordships that not every private record would be admissible. If challenged it would be necessary to prove in some way that it had proved to be reliable, before the judge would allow it to be put before the jury. And I think that some such limitation must be implicit in the last reason given by the Court of Criminal Appeal. I see no objection to a judge having a discretion of this kind though it might be awkward in a civil case; but it appears to me to be an innovation on the existing law which decides inadmissibility by categories and not by apparent trustworthiness ...

Structure of Lord Reid's Argument

$\Delta_1 : N =$ number on car A is a, (when assembled), and Δ_1 is the support of this claim.

$\Delta_1 =$ description of procedures of entering numbers during assembly.

[15] Our footnote: i.e. $u_1 : X$ is admissible only if some other $u_2 : X$ is already admissible. See objection $s_{3,2}$ below. *LDS* allows formally for putting item $u_1 : X$ in the database in such a way that it can be used only in the flattening process to support other items but not in deduction.

We also have a common sense metalevel persistence principle: numbers on cars persist (don't fade away or change).

$$N \rightarrow \textbf{Always}\, N.$$

Thus, according to Lord Reid, t is equal to:

$$t = \{\Delta_1 : N, N \rightarrow \textbf{Always}\, N\}.$$

He wants to block the use of t by attacking the admissibility of Δ_1.

Four reasons were quoted for the admissibility of Δ_1 and three reasons for non-admissibility:

r_1: Evidence of this kind is admitted in Central Criminal Court.
r_2: Judge has discretion to admit such evidence.
r_3: This is a list of reasons given in Court of Criminal Appeal, namely:

- $r_{3,1}$: The records were produced to show that the records were kept in the normal course of business (but not to prove the truth of the recorded particulars).
- $r_{3,2}$: Although the record may not be evidence by themselves, they may be used to corroborate other evidence.
- $r_{3,3}$: We do not have dependency on the credit of an unidentified person but rather on a probably reliable process of record maintenance, and can therefore admit them.

r_4: Common sense rebels against rejection of such evidence.
s_0: No reported case or any textbook as direct authority for admission.

It seems at this point that r_1–r_4 are stronger than s_0.[16] So Lord Reid is trying to weaken the force of r_3 and r_2 by attacking them logically with s_3 and s_2:

s_2: Judges do not have the discretion to admit legally inadmissible evidence.
s_3: Counter argument to r_3 comprising of:

- $s_{3,1}$: If the records are not intended to prove the truth of their entries, what are they intended to prove? (I.e. they are irrelevant!)
- $s_{3,2}$: Either the records are admissible or not. There is no sense in which they can become admissible only after some other evidence to the same conclusion becomes admissible (see footnote 15).
- $s_{3,3}$: Such records are not public records which are admissible for reasons that they are open to the public for inspection and correction. The current law therefore does not support their admissibility.

[16] In other words, it seems that a reasonable flattening process, weighing $\{r_1, r_2, r_3, r_4\}$ against $\{s_0\}$ will decide in favour of the former and thus admit the records. Note that no rules are given at this stage of how the decision is made. In some logics, where labels are confidence numbers, we can give a rule; e.g. admit iff $r_1 + r_2 + r_3 + r_4 > s_0$, but not here.

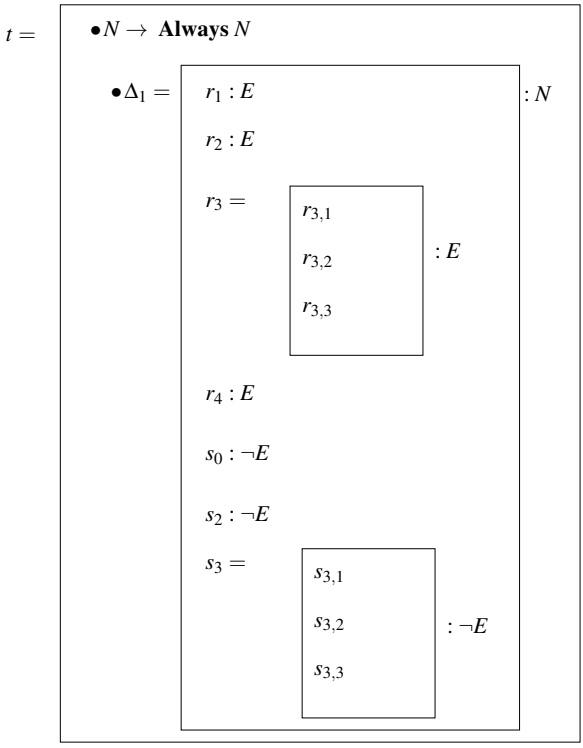


Fig. 3.13

Figure 3.13 shows the form of t, where E = admit evidence or 'use me'.

To strengthen his case (i.e. strengthen the overall labels for $\neg E$, Lord Reid is attacking the label r_3 by putting forward $s_{3,1}, s_{3,2}$ and $s_{3,3}$. Note that the reasoning in the different boxes can be of different kinds!

Note that one of the points Lord Reid is making is s_2, namely that trial judges do not have discretion to 'admit legally inadmissible evidence'.[17]

So the force of the argument is to influence the flattening process: we have r_1–$r_4 : E$ and $s_0, s_2, s_3 : \neg E$, which one wins?

In this case the evidence was not admitted.[18]

[17] A customer calls the bank wanting to transfer money. The bank needs to identify him. The clerk has authority to deny the customer service if he feels the identification is not complete or if the customer sounds suspicious. What the clerk cannot do is to say to the customer: you failed to identify yourself but you sound honest, so I will transfer the money for you.

[18] This decision was made by vote as described in the quote from Keane (2000) on our page 218.

Uglow continues:

> The House of Lords recognized the absurdity of their position but felt strongly that it was for the legislature to reform the law and create new exceptions. Parliament dealt with the problem of documentary hearsay with the Criminal Evidence Act 1965 which created an exception for trade and business records This was later extended by section 68 of the Police and Criminal Evidence Act 1984 and now by sections 23 and 24 of the Criminal Justice Act 1988. Such records have all been admissible in civil proceedings since the Civil Evidence Act 1968.
>
> Myers has been regularly followed in such cases as *Patel v Comptroller of Customs* ([1965] 3 All E.R. 593) where the appellant was convicted of making a false declaration to customs, having stated that the bags of seed were originally from India. The prosecution sought to prove that the seed originated in Morocco and adduced evidence that the bags were stamped with 'Produce of Morocco'. The Privy Council, following *Myers* held that these words were hearsay and inadmissible. Unlike *Myers*, there was no evidence that the writing was at all reliable, there being no testimony as to how or by whom the bags were marked.

The reader should note that the main thrust of the argument and logic of the Lord Reid example is in weakening and strengthening labels. Put schematically we have a master argument, say E which can prove a conclusion on D. E is a labelled argument containing various labels within labels. Among this maze of labels there is a label t containing another argument, say Δ. To attack E we can attack Δ. Our argument attacking Δ can itself be attacked by attacking some label s in it and so on. This is reminiscent of systems of abstract argumentation theory. We can give actual proof rules and labelling disciplines so that questions like export from one label to another can also be considered. For example:

"If you weaken t then D will not follow from E, and that would be a bad precedent."

One cannot argue in this way unless a specific labelled model is available. We think that we have seen enough to be convinced that labelling logics can play a central role here, though we would understand if the cautious reader would prefer to reserve judgement until more case studies are presented. See our paper Gabbay and Woods (2003).

Appendix B. Case Study: Modal Logic

Modal and temporal logic is a good motivating example for labels. In essence modal logic deals with information (i.e. formulas) related to different worlds or times and with patterns among these worlds. It is therefore very natural to name and explicitly refer to these worlds and the way they are related. We also have a distinguished world and time which is where we are. Thus an *LDS* approach, where we use labels to name worlds and a labelling language $\mathcal{A}$ to describe patterns of worlds comes very naturally indeed.

The *LDS* system for modal logic can be easily motivated from the traditional Kripke semantics for modal logic. A traditional Kripke structure has the form (S, R, a, V, h), where S is the set of possible worlds, R is the accessibility relation, $a \in S$ is the actual world and V is a function giving for $t \in S$ the domain V_t of

Fig. 3.14

world t. h is the assignment function giving for each t and each atomic predicate its extension in V_t. When we write a formula of modal logic, say $B = \neg A \wedge \Diamond A$, it is interpreted as saying 'I hold in the actual world a'. So the syntax of the modal language cannot directly say much about the nature of the Kripke model at worlds other than a. There is some indirect capability, through evaluating at a. For example, for B to hold at a we must have a point $t \in S, a < t, a \neq t$ such that $a \vDash \neg A$ and $t \vDash A$. We can state that explicitly by writing

$$\Delta_B = \{a : \neg A, a' : A, a < a', a \neq a'\}.$$

Δ_B is satisfied in a Kripke structure if a, a' can be instantiated (with a being the actual world), with the indicated R and $\neq$ relations validated and the corresponding wffs hold at the respective points.

The above, however, is the notation of an *LDS* for modal logic.

Thus we can adopt the view that *LDS* for modal logic arises from our desire to be more explicit in the syntax about relationships between possible worlds and formulas holding in them. We want explicit names for worlds beyond the implicit actual world which is traditionally available. Thus a *theory* or a *database* becomes a *configuration* of labelled wffs, see Definition 3.24.

Once we take this step, we can develop proof theory on such configuration, and study traditional concepts and machinery for this presentation. Let us look at some examples:

Example 3.21 (Some modal rules). We begin with the simple configuration of Fig. 3.21.

The modal axioms and the meaning of $\Box$ dictate to us that in the constellation displayed in Fig. 3.21, A must hold at s. Further, the meaning of $\Diamond$ tells us that there should exist a point r with $s < r$ such that $r : B$.

We can thus state two rules for manipulating modal databases.

$$(*1) \qquad \frac{t : \Box A; t < s}{s : A}$$

and

$$(*2) \qquad \frac{s : \Diamond B}{\text{Create } r, s < r \text{ and } r : B}.$$

Using the first rule we manipulate the constellation displayed in Fig. 3.21 into the one of Fig. 3.15 and using the second rule we further manipulate it into that

Fig. 3.15 AAA

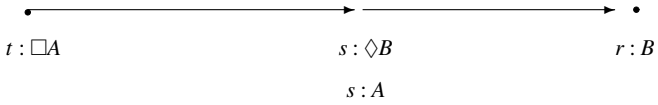


Fig. 3.16 BBB

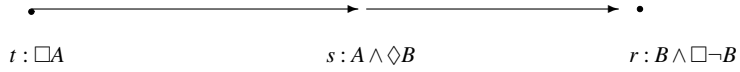


Fig. 3.17 CCC

displayed in Fig. 3.16. In the predicate case r depends on the free variables of B. The second rule is good for modal logics like **K**, **S4**, etc.

The axiom of Löb:

$$\Box(\Box A \to A) \to \Box A$$

corresponds to the modification rule

$$(*3) \qquad \frac{s : \Diamond B}{\text{Create } r; s < r \text{ and } r : B \wedge \Box \neg B}$$

thus in the logic with the Löb axiom we get from the configuration of Fig. 3.21 to the configuration in Fig. 3.17.

It is clear now how the rules work. They allow us to move from one configuration to another and the consequence relation is between configurations. For example, we have Fig. 3.21 $\vDash$ Fig. 3.16, in modal **K** and with Löb's axiom we have Fig. 3.21 $\vDash$ Fig. 3.17.

The above rules are elimination rules. We still need introduction rules

$$(*4) \qquad \frac{s : A; t < s}{t : \Diamond A}.$$

Table 3.1

Axioms	*LDS* Features
K axioms $\Box(A \to B) \to (\Box A \to \Box B)$ $\vdash A \Rightarrow\Rightarrow\Rightarrow \vdash \Box A$ $\Diamond A =_{\text{def}} \neg\Box\neg A$	The notion of basic constellation or a diagram, as in Sect. 3.2.1 Note that in the modal case the relation R in the diagram is binary. The *LDS* formulation contains also some simple rules for $\Box$ and $\Diamond$ some of which were shown in the figure above, rules (*1), (*2)
$\Box A \to \Box\Box A$	Transitivity of R in the constellation
$\Box(\Box A \to A) \to \Box A$	In modal semantics the axiom has no first order condition. It corresponds to the finiteness of the frame. In *LDS* it corresponds to the modification rule (*3).
$\Diamond A \wedge \Diamond B \to$ $\Diamond(A \wedge B) \vee \Diamond(A \wedge \Diamond B)$ $\vee\Diamond(B \wedge \Diamond A)$	Corresponds to the linearity of the relation R. This affects the basic rule (*2) as explained in Remark xxx

$$(*5) \qquad \frac{\begin{array}{c}\text{Create an arbitrary } s; t < s \\ \text{and Show } s : A\end{array}}{\begin{array}{c}\cdot \\ t : \Box A\end{array}}$$

Example for $\Box$ introduction:

Given $t : \Box(A \to B) \wedge \Box A$
Create $s, t > s$
Show $s : B$
Deduce $t : \Box B$.

The picture however is not as simple as it seems. In the usual formulations of modal logics, axioms correspond to conditions on the possible world relation.

In our presentation, axioms correspond to any one of a variety of features. Table 3.1 offers a selection.

We see here how a second order axiom, i.e. the axiom of Löb, which corresponds to a second order semantical condition, can become a simple movement in LDS. When *LDS* is translated into two sorted classical logic, the function symbols generating the labels may allow us to reduce the second order condition into first order, as is the case with McKinsey axiom $\Diamond\Box q \to \Box\Diamond q$, when added to **K** without transitivity.

Remark 3.22 (Linear frame modal logic). Suppose we deal with the modal logic for linear frames. What does it mean proof theoretically? It means that not all configurations are acceptable; only the linearly ordered ones.

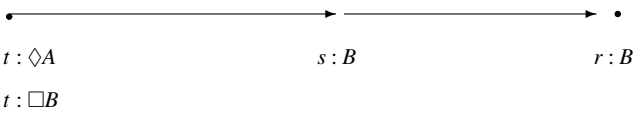


Fig. 3.18 DDD

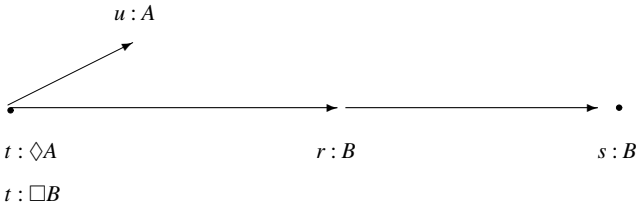


Fig. 3.19 EEE

Let ∂ be the formula (axiom of linearity)

$$\partial = \forall xy(x \leq y \vee y \leq x)$$

then a configuration is acceptable iff it validates ∂. Consider the configuration in Fig. 3.18. It is an acceptable one and can be expanded in three ways.

By rule (*2) we can create a point $u : A$, with $t < u$. In a non linear modal logic such as **K, S4**, etc. this would lead us to the configuration of Fig. 3.19.

One more step would allow us to have $u : A \wedge B$ and hence by $\Diamond$ introduction we get $t : \Diamond(A \wedge B)$.

However in the case of linear modal logic, Fig. 3.19 is not allowed. We need to consider five possibilities.

1. $t < u < r < s$
2. $t < u = r < s$
3. $t < r < u < s$
4. $t < r < u = s$
5. $t < r < s < u$

If, as a result of *each* of these possibilities, we end up with $t : \Diamond(A \wedge B)$ then we can conclude $t : \Diamond(A \wedge B)$.

We have to do that because our databases are linear and the above five configurations are all the minimal possible extensions in which u can be accommodated.

Fig. 3.20 FFF

We thus have to modify all the rules with 'Create' in them to mean:

Given initial configuration

Split proof into n branches according to all minimal allowed extensions in which the created u can be accommodated.

(*3) becomes (**3)

$t : \Diamond A$ in a configuration

$(**3)$ create u, consider all allowed minimal extensions of D with u in them. Put $u : A$ in and branch the proof. The ultimate goal of the overall proof must succeed in all branches.

The above is computationally very expensive. In the example previously given, we need to go to five configurations in order to make the simple move

$$(*6) \qquad \frac{t : \Box A \wedge \Diamond B}{t : \Diamond(A \wedge B)}$$

However our *LDS* proof discipline does not stop us from adopting (*6) as a rule. Recall that the *LDS* discipline tries to enjoy both worlds—the classical world through the labels and the special non-classical world through the language of the formulas in the labels. For each application, desired balance can be sought.

We now come to quantifier rules. We have already assumed that different labels will have different sets of elements in them. To appreciate what this means, we take our clue from modal logic. Consider Fig. 3.20.

At the label t, an x exists such that $t : \Diamond A(x,y)$ holds. This x depends on t and on y. We therefore need a Skolem function $c^t(y)$. The index t is read to mean that c^t was created at t. We thus get $t : \Diamond A(c^t(y),y)$. Hence we can create a node $s : A(c^t(y),y)$. We also must indicate whether $c^t(y)$ 'exists' at the node s. If it does exist at s (probably because of some rules) then we write $s : c^t(y)$. The difference

comes out in existential introduction. Suppose we have $s : E(c^t(y))$, can we infer $s : \exists x E(x)$? The answer depends whether $c^t(y)$ exists at s or not. Here are some rules:

$$(*7) \qquad \frac{s : c; s : E(c)}{s : \exists x E(x)} .$$

$$(*8) \qquad \frac{t : \exists x A(x, y_1, \ldots, y_n)}{t : A(c^t(y_1, \ldots, y_n), y_1, \ldots, y_n); t : c^t(y_1, \ldots, y_n)} .$$

$$(*9) \qquad \frac{t : \forall x A(x)}{t : A(u^t); t : u^t,} \; u^t \text{ is a universal constant.}$$

$$(*10) \qquad \frac{s : u^t; s : A(u^t); s : c^r}{s : A(c^r)} .$$

u^t is a new universal constant, r is arbitrary.

$$(*11) \qquad \frac{t : c^r; s : A(u^t)}{s : A(c^r)} \; u^t \text{ a universal constant.}$$

$$(*12) \qquad \frac{s : u^s; s : A(u^s)}{s : \forall x A(x)} \; u^t \text{ a universal constant.}$$

Rule (*9) is analogous to the classical logic rule which allows us to replace $\forall x A(x)$ by $A(u)$, where u is a universal constant, i.e. u is arbitrary. At any stage later in a classical logic proof, we can pass from $B(u)$ to $\forall u B(u)$ provided we discharged all additional assumptions. We can certainly pass from $B(u)$ to $B(c)$, c any constant. The same considerations apply to the labelled case except that we have to watch for the added complication that elements created in one label (world) (e.g. c^r, u^t) may not exist in another label (world). Imagine we have $t : \forall x A(x)$, this means A holds for all elements existing at t. We use rule (*9) and represent $t : \forall x A(x)$ by a universal constant u^t, i.e. we have now $t : u^t$ and $t : A(u^t)$. Suppose for some proof theoretical reason, $s : A(u^t)$ is obtained. Thus we really have that A holds at s for an arbitrary element existing at t. Suppose now that we know that the element c^r created at r, exists at t. This is written as $t : c^r$. Then we can deduce $s : A(c^r)$. This is rule (*11).

We now explain rule (*10). Start with $t : \forall x A(x)$, this by rules (*9) and (*12) is equivalent to having $t : u^t$ and $t : A(u^t)$. Suppose that by some proof manipulation we end up with $s : u^t$ and $s : A(u^t)$. This means that the universal constant u^t is in label s and so is $s : A(u^t)$. We understand that as a proof of $s : \forall x A(x)$ from $t : \forall x A(x)$ and so we allow ourselves to deduce $s : \forall x A(x)$. Therefore for any c^r, which exists at s displayed as $s : c^r$, we get $A(c^r)$ at s, i.e. $s : A(c^r)$. This entire chain is summarized as rule (*10).

So far these rules assume that somehow an element c^t created at t ends up available at label s, i.e. $s : c^t$ holds. How do elements move around? We need special rules for that and they differ from system to system. In other words the logic must tell us how elements skolemized in one label can be transported to another label. These are called *visa rules*. Here are two sample rules corresponding to the Barcan and converse Barcan formulas:

$$(b1) \qquad \frac{t : x^r, t < s}{s : x^r} \; x \text{ either a constant } c \text{ or a universal constant } u.$$

$$(b2) \qquad \frac{t : x^r, s < t}{s : x^r} \; x \text{ either constant } c \text{ or a universal constant } u.$$

Example 3.23 (Barcan formula revisited). Use (b2) to show that

$$t : \forall x \Box A(x) \vdash t : \Box \forall x A(x)$$

1. Start $t : \forall x \Box A(x)$.
2. $\forall$-Elimination at t yields $t : \Box A(u^t), t : u^t$.
3. Create an arbitrary $s, t < s$, and let u^s be arbitrary with $s : u^s$.
4. $t : u^s$ by visa rule (b2).
5. From (2) and (4) and rule (*10) we get $t : \Box A(u^s)$
6. From (5) we get $s : A(u^s)$.
7. $s : \forall x A(x)$, by (*12).
8. $t : \Box \forall x A(x)$, since s was arbitrary.

To present modal logic as an *LDS* we take the usual language of modal logic with $\Box$ and $\Diamond$ as our $\mathbf{L}$ and take sets D with a binary relation $\leq$ as our labels. It is convenient to think of $(D, <)$ as a configuration.

Definition 3.24.

1. A declarative unit is a pair $t : A$ where t is a label and A is a formula of the modal language. A labelled term has the form $t : c^s$, where t, s are labels and c is a constant or variable of the modal language. The double indices for terms are needed because c can be created at label s and be used or be present at label t. We write $t : c^s$ to denote that.
2. A configuration has the form $(D, \mathbf{f}, <, d, U)$, where $(D, <, d)$ is a finite ordered set, with $d \in D$ and $\mathbf{f}$ a function, $\mathbf{f} : D \mapsto$ wff and U a function to terms such that $\mathbf{f}(t) =$ a set U_t of formulas and a set of labelled terms.

$\mathbf{f}(t)$ and U_t together can be represented as

$$\{A_1, A_2, \ldots, c^{s_1}, c^{s_2}, \ldots, \}.$$

Note that the terms are labelled arbitrarily.

3. Queries have the form $s : A$.
4. Let Δ be a configuration and let (S, R, a, V, h) be a Kripke structure, with an assignment h to the variables and constants of the language. We say the structure *satisfies* Δ iff there is a mapping $g : D \to S$ such that the following holds

   a. $g(d) = a$
   b. $\partial, D \vdash x < y$ implies $g(s)Rg(y)$
      $\partial, D \vdash \neg(x < y)$ implies $\neg g(x)Rg(y)$
      $\partial, D \vdash x \neq y$ implies $g(x) \neq g(y)$
   c. $h(c^t) \in V_{g(t)}$
   d. $t : c^s \in D$ implies $h(c^s) \in V_{g(t)}$
   e. $t : A \in D$ implies $g(t) \vDash_h A$.

   Note that this clause makes a configuration inconsistent if one of the nodes (e.g. t) has an inconsistent $\mathbf{f}(t)$ set of formulas.

Definition 3.25. A modal *LDS* system is determined by the following components

1. A class $\mathcal{K}$ of ordered sets $(D, <, d)$ to be used in the configurations of the system, which can possibly but not necessarily be characterised by a formula ∂.
2. Inference rules. The inference rules manipulate configurations. These include creation and elimination of points $t \in D$ and the introduction and elimination of wffs. The rules are divided into the following categories.

   2.1. Introduction and elimination rules for connectives specialised for modal logic.
   2.2. Quantifier rules, including Skolemization, as intuitively defined in the discussion above.
   2.3. Individual elements *visa* permits (mobility of elements from one node to another). These have the general form

$$\frac{(D, \mathbf{f}, <, d, U)}{(D, \mathbf{f}', <, d, U')}$$

   where U' is like U except that for some c^s and $t \in D$, $U'_t = U_t \cup \{c^s\}$.

We write

$$t : A; t : c^s; t < r$$

to represent the information that in the configuration $(D, \mathbf{f}, <, d)$ which we are dealing with (i.e. re-writing), we have $t, s, r \in D, t < r$, and $\mathbf{f}(t)$ contains A and c^s.

The notation '$\Delta(t,s); t < s; s : B; s : c^r; s$ a new point' means that we are given a configuration $\Delta = (D, \mathbf{f}, <, d, U)$, with $t \in D$. We add a new point s to D to form $D' = D \cup \{s\}$ and extend $<$ by stipulating $t < s$ and let $\mathbf{f}'$ be like $\mathbf{f}$ and U on D and let $\mathbf{f}'(s) = \{B\}$ and $U_s = \{c^r\}$. Then '$\Delta(t,s), t < s, s : B, s : c^r$' denotes $\Delta' = (D', \mathbf{f}', \leq, d, U')$.

A rule of the form

$$\frac{t : A}{\text{Create } s, t < s; s : B; s : c^r}$$

should be understood as an abbreviation for the rule below:

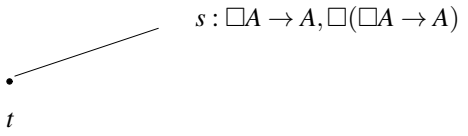
$$\frac{\Delta}{\Delta(t,s); t < s; s : B; s : c^r}$$

where $t : A$ appears in Δ and is the declarative unit triggering the rule, as explained above.

Example 3.26. Show $\Box(\Box A \to A) \vdash \Box A$.

1. Assume $t : \Box(\Box A \to A)$ and show $t : \Box A$.
2. Use an introduction rule. Create an $s, t < s$ and show $s : A$.

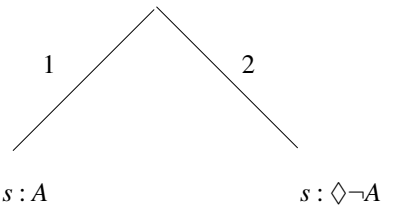
2.1. Use $\Box$ elimination rule.



2.2. Use classical logic to rewrite the entry at s.

$$s : \Diamond \neg A \vee A$$

2.3. Split into two cases



Case 1 is a success because we needed to show $s : A$.

We proceed with case 2 and show it leads to a contradiction:
Create a new point r:

Bring $\Box A \to A$ from s.

$$r : \Box A \to A$$

Use classical logic and get:

$$r : A$$

A contradiction, because we also have $r : \neg A$.

3. Since we showed (2) successfully, we conclude $t : \Box A$.

Example 3.27. Test whether $\Diamond \exists x A(x) \vdash \exists x \Diamond A x$

1. $t : \Diamond \exists x A(x)$
2. Create s, with $t < s; s : \exists x A(x)$
3. Skolemise and get $s : A(c^s)$.
4. $t : \Diamond A(c^s)$
5. $t : \exists x \Diamond A(x)$, to be derived only if there is a visa for c^s to be at t.

Example 3.28. Test whether $\exists x \Diamond A x \vdash \Diamond \exists A x$

1. $t : \exists x \Diamond A(x)$
2. $t : \Diamond A(c^t)$
3. Create s, with $t < s; s : A(c^t)$
4. $s : \exists x A(x)$, to be derived only if there is a visa for c^t to be at s.
5. $t : \Diamond \exists x A(x)$

Example 3.29. Our modal rules can be label dependent rules, for example $\Diamond$ can change meaning from world to world:

$$\frac{t : \Diamond_t A}{\text{Create } t < s_1 < \ldots < s_{n(t)}, s_{n(t)} : A}$$

The semantic condition for this modality is:

$$\|\Diamond A\|_t = 1 \text{ iff } \|A\|_{t+n(t)} = 1.$$

Different truth tables in different worlds.

Let us give a quick proof that

$$t : \Box A; t : \Diamond B \vdash t : \Diamond (A \wedge B)$$

We tacitly assume that conjunction introduction is available (or indeed that classical logic can be used locally at node t).

1. Initial configuration
$$t : \Box A, \Diamond B$$
2. Create an $s, t < s$ with $s : B$ we get the configuration
$$t : \Box A, \Diamond B; s : B; t < s.$$
3. Move A to $s : A$, using the rule:
$$\frac{t : \Box A \quad t < s}{s : A}$$
we get the configuration
$$t : \Box A, \Diamond B; s : B, A; t < s.$$
4. Add $t : \Diamond(A \wedge B)$, using the rule
$$\frac{s : A, \quad t < s}{t : \Diamond A}$$
we get the configuration
$$t : \Box A, \Diamond B, \Diamond(A \wedge B); s : B, A; t < s.$$

We have thus proved that $\Box A, \Diamond B \vdash \Diamond(A \wedge B)$ because we started with the configuration in (1) with $t : \Box A, \Diamond B$ and we step by step manipulated it into the configuration in (4) which contained $t : \Diamond(A \wedge B)$.

Example 3.30. To show that the logic depends on the class $\mathcal{K}$ of orders, assume we want $(D, <, d)$ to be linearly ordered. We now try and prove
$$t : \Diamond A \wedge \Diamond B \vdash t : \Diamond(A \wedge B) \vee \Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A)$$

To show that assume

1. $t : \Diamond A, \Diamond B$
2. Create $s, t < s$ with
$$s : A$$
3. Create $r, t < r$ with
$$r : B$$

However, since only linear orders are allowed our options are
$$t < r < s,\ t < r = s,\ t < s < r.$$

In each of these options, the conclusion can be proved. The lesson to be learnt is that the class of allowed configurations can influence the proof theory.

Example 3.31. We show that

$$t : \Box A; t : \Diamond B \vdash t : \Diamond(A \wedge B)$$

1. Initial configuration
$$t : \Box A, \Diamond B$$
2. Create an $s, t < s$ with $s : B$ we get the configuration
$$t : \Box A, \Diamond B; s : B; t < s.$$
3. Move A to $s : A$, using the rule:
$$\frac{t : \Box A \quad t < s}{s : A}$$
we get the configuration
$$t : \Box A, \Diamond B; s : B, A; t < s.$$
4. Add $t : \Diamond(A \wedge B)$, using the rule
$$\frac{s : A, \quad t < s}{t : \Diamond A}$$
we get the configuration
$$t : \Box A, \Diamond B, \Diamond(A \wedge B); s : B, A; t < s.$$

We have thus proved that $\Box A, \Diamond B \vdash \Diamond(A \wedge B)$ because we started with the configuration in (1) with $t : \Box A, \Diamond B$ and we step by step manipulated it into the configuration in (4) which contained $t : \Diamond(A \wedge B)$.

Example 3.32 (Inconsistency in modal LDS*).* Let us see how a configuration can be inconsistent. This will clarify the notion of inconsistency for us.

1. Consider the configuration
$$\{t : A; s : \bot\}.$$
$\bot$ is the symbol for *falsity*. This configuration is *not* necessarily inconsistent, unless we have the rule
$$\frac{s : \bot}{s' : \bot}$$
According to the notion of satisfaction in item (4) of Definition 3.24, this rule is valid. However, if we change the notion of satisfaction in (4e) of the above definition and required only that

(e') for some $t \in D$, we have that $g(t) \models_h A$ for all $A \in \mathbf{f}(t)$, then the rule

$$\frac{t : \bot}{s : \bot}$$

would no longer be valid. That is, inconsistency in one world does not necessarily allow us to prove anything in any other world.

2. Let $\partial = \forall xyz(x = y \vee y = z \vee x = z)$. This says that the configuration can have at most two different points. Consider the configuration

$$\Delta = \{d : A \wedge \neg B; s : A \wedge B \wedge \Diamond\neg A, d < s\}$$

This is not consistent since there is no way we can create a third point for the item $s : \Diamond\neg A$. In other words, we cannot consistently apply the $\Diamond$-elimination rule.

We can formally apply the *LDS* rules and get the syntactical configuration

$$\Delta' = \{d : A \wedge \neg B, s : A \wedge B \wedge \Diamond\neg A, r : \neg A, d < s, s < r\}.$$

This will not have a model (as defined in item 4 of Definition 3.24).

The question is how do we detect inconsistency syntactically? Certainly the diagram D of Δ' together with ∂ prove $\bot$ in the theory of the order $<$. If we add the formal rule

$$\frac{\bot}{t : A}$$

for arbitrary t and A, we will have that Δ can prove in *LDS* any $t : A$. We can define proof theoretic inconsistency as the ability to prove everything.

This is reminiscent of the semantic tableaux method where inconsistency means not being able to successfully continue the tableaux construction along any path. However, the *LDS* proof theory is not model construction. Consider the database $\{d : \Diamond A\}$. The proof theory will turn it into $\{d : \Diamond A; d < s; s : A\}$. No need to apply any more rules. This is not a model yet. It is only a sort of *stable* database, where rule applications do not add anything to it.

Let us check how the inconsistency of the above Δ manifests itself in the traditional formulation. Δ can be expressed only through what holds at the actual world. So Δ would be the wff $C = A \wedge \neg B \wedge \Diamond(A \wedge B \wedge \Diamond\neg A)$. The condition ∂ on possible worlds will have to be formulated as a proof theoretical condition in the system.

A suitable Hilbert axiom or rule needs to be found. Say in this case the family of axioms:

$$A \wedge \Diamond\neg A \wedge \Diamond^n B \rightarrow B \vee \Box(\neg A \rightarrow B)$$

The inconsistency will arise from proving a contradiction using C and the suitable axioms.

In the *LDS* framework the notion of inconsistency must be defined using the *LDS* proof rules.

Example 3.33. The following are examples for making $\Box^t$ label dependent.

1. The basic meaning for $\Box A$ is 'always A'. It is not label dependent. The corresponding *LDS* rule is:

$$\frac{s : \Box A, s < r}{r : A}$$

2. The basic meaning for $\Box^t A$ is 'always A up to t'.
The corresponding rule is:

$$\frac{s : \Box^t A, s < r < t}{r : A}$$

3. The basic meaning of $\Box^t A$ is 'always A except at t'
The corresponding rule is:

$$\frac{s : \Box^t A, t < s, s \neq t}{s : A}$$

4. The basic meaning of $s : \Diamond^t A$ is 'A is true at t and $s < t$'. The corresponding rules are:

$$\frac{s : \Diamond^t A}{s \leqq t}$$

and

$$\frac{s : \Diamond^t A}{t : A}$$

Discussion
Advantages of the *LDS* proof discipline for modal logics:

- No conceptual problem in Skolemising and theorem proving, which is a major problem for modal logic.
- The principles involved are more general, good for any *LDS*.
- We can handle systems which have semantics which is higher order (Löb's system is higher order because it has to be characterised by a higher order condition on the Kripke frame, namely it being a finite irreflexive partial ordering). This is due to the fact that we can bring the semantics into the syntax.
- The modal logic can locally change from world to world, all we have to do is to make our rules label dependent (i.e. containing labels as parameters).

Appendix C. Actions, Labels and Proofs

This appendix shows how the labels can be sequences of actions, and how we can do labelled proof theory with them.

C.1 Insurance Example

We begin with a modified insurance example. The example shows that we must deal with data that have the capability of changing because of actions we can take. So the effective database is the actual data together with the sequences of actions available to us.

Our friend John is a drifter. He lives in a van and gets the odd job from time to time. He is a fairly responsible, honest and rational person but he does not seem to settle down and is always short of money. Unlike other drifters, he does try to take care of his van, and more importantly, does maintain an insurance policy with the local insurance company. Because of his lifestyle, the premium is relatively high and the terms of the insurance are that John has to pay all the premium for any year by the 31st January of that year. Our story is set in January 15th of this year. John has not yet paid the insurance premium for the year. He does not have the money to pay, but he hopes to raise it by the 31st. It is common practice among insurance companies that coverage during January is valid provided the premium is indeed paid by January 31st. Thus if John has an accident on January 15th, he is covered provided he pays the premium by January 31st, otherwise he is not covered.

Our story begins when John, in bad visibility and heavy rain, happens to bump into Mr Rich's expensive car, causing extensive damage. The circumstances of the accident are not clear cut and there are arguments both ways about whose fault it is. It would help Mr Rich to collect damages for his car, however, if John co-operates in not emphatically insisting that it was Mr Rich's fault. Mr Rich cannot claim damages from John personally, because John has no money. However, if John pays his premium by January 31st, and clearly admits fault, then Mr Rich can claim the full value from John's insurance company. Mr Rich does not want to claim from his own insurance company because he would lose his 65% no-claims bonus. Mr Rich wants to persuade John to formally admit fault. Indeed, Mr Rich can make it worthwhile for John to do so. However, John is a straight person and would need to be persuaded that at least it is possible (though not necessary) to look at the accident as his fault.

Let us describe now the realities of the situation semiformally:

- we can assume that John's premium is $ 500.
- The damage to Mr Rich's car is $8000.
- Mr Rich's loss of the no-claims bonus is worth $3000 (spread over several years).
- The damage to John's van is assumed to be negligible.
- John's extra insurance premium, should he admit technical fault, is also negligible. His insurance company will continue the policy since he had never had an accident before. The change in premium will not be much, since his lifestyle did not support a serious premium reduction anyway.

It is obvious that it is worth Mr Rich's while to help John pay his premium, and persuade him to admit fault. If John does not pay his premium and Mr Rich tries to claim from him, the inconvenience to John cannot be easily quantified, as he is a

drifter and his views of his future are not so clear, and he can deny fault and may be assigned a free lawyer by the court. In fact, Mr Rich's insurance company is very likely to decide not to go to court under the circumstances.

We need some notation to formally describe the practical reasoning situation at the time of the accident:

1. Let Δ_a be the database containing the facts of the accident. We assume Δ_a is a non-monotonic labelled database, capable of proving with the appropriate label that John could reasonably accept fault[19]
2. Consider the following dictionary:

   - q = John is at fault
   - q_1 = John formally accepts fault
   - c = John is covered by insurance
   - r = John's insurance pays damages to Mr Rich.
   - p = John's van causes damage to Mr Rich's car.

The complete set of data, Δ_w, at the time of the accident is as follows (w = January 15th).

1. Δ_a
2. $p \wedge q_1 \wedge c \rightarrow r$
3. p.

Mr Rich wants to make r true.[20] To achieve this he needs to make q_1 and c true. Since John has no money, Mr Rich will have to pay John's insurance premium. Making q_1 true is more difficult. John will not do that unless there is a reasonable case for showing q from Δ_a. Of course there may be an equally reasonable case for showing $\neg q$ from Δ_a. To give John an incentive to admit to q_1, Mr Rich offers him additional money, ostensibly for the 'shock' he suffered during the incident. This would be a private arrangement in the form of cash. The following matrix, Table 3.2 seems reasonable to Mr Rich:

The matrix shows that such an arrangement is rational, provided John, who is relatively honest, can bring himself to make q_1 true. This would depend on the strength of the argument from Δ_a in favour of q.

Let us further formalise the actions involved. We represent actions by bold letters $\mathbf{m}, \mathbf{a}, \mathbf{b}, \mathbf{c}$, etc. and adopt the artificial intelligence view that actions $\mathbf{x}$ have

[19] Such databases are generally inconsistent and for many statements A they can prove both A and $\neg A$ with appropriate labels. See Gabbay and Hunter (1991, 1993). We do not have to assume that John and Mr Rich agree on the database. A labelled database can code both John's and Mr Rich's versions and arguments. So we can have $\Delta \vdash t : A$ and $\Delta \vdash s : \neg A$, where t is a label representing one way of reasoning from some version of the data and s represents another way. Thus all we want to assume is that there is a label (a way) of showing that Δ_a proves that it is John's fault.

[20] Notice that we use the phrase 'make r true'. We shall argue in a later section that taking actions is part of the reasoning and proof process. See Sect. C.4 for oscillating non-monotonic proofs with actions.

Table 3.2

	deal	no deal
John	Receives \$1500 from Mr Rich	No money to receive. May incur possible cost if Mr Rich's insurance tries to claim damages.
Mr Rich	Retrieves \$6500 (\$8000 claim less \$1500 given to John.	Retrieves \$5000 (\$8000, from own insurance less \$3000 - no claim bonus lost

preconditions, denoted by $\alpha_{\mathbf{x}}$ and postconditions denoted by $\beta_{\mathbf{x}}$. The preconditions need to hold in order for the action to be taken and once taken the postconditions are guaranteed to hold.[21]

We therefore have the following actions:

1. Mutual action **m**
   John and Mr Rich agree to the scenario of the proposed deal.
   Precondition $\alpha_{\mathbf{m}}$ is a reasonable proof of q from Δ, delivered by Mr Rich in a subtle way. The postcondition is an agreed action scenario, first action **a** then actions **b** and **c**.
2. Action **a**
   Mr Rich gives \$1500 to John.
   The preconditions are that John and Mr Rich agree to have a deal.
   The postconditions are that John pays the premium before January 31st and that he formally admits fault, (i.e. John has to do actions **b** and **c** below).
3. Action **b**
   John pays the premium before January 31st.
   The preconditions are that John gets \$1500. The postconditions are c (that John is covered at the time of the accident).
4. Action **c**
   John formally admits fault.
   The precondition is action **a**.
   The postcondition is q_1.

The following tree (Fig. 3.21) contains some (but not all of the) possible scenarios for John and Mr Rich.

At time w, John can choose not to make an agreement. In this case Mr Rich's insurance company will sue John and John will either win or lose the case.

[21] Our model will allow for actions to be taken anyway, even if the preconditions are not satisfied. This often happens in real practical reasoning situations, where the penalty of ignoring the preconditions is less than the reward of the postconditions. We can improve our model by allocating a *counteraction* to any action, where the counteraction is enabled if the action was performed without its preconditions holding.

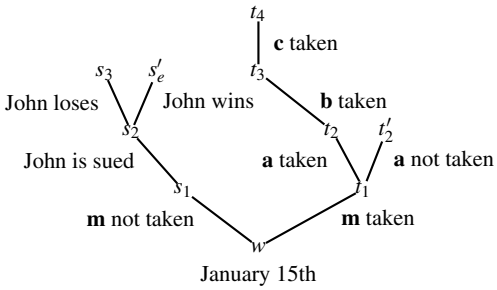


Fig. 3.21

An alternative time t_1 there is agreement, and so part of the database Δ_{t_1} at time t_1 is the agreed scenario $(t_1, \ldots, t_4)$. Of course Mr Rich may change his mind and branch off to scenario $(t'_2, \ldots)$, which may be similar to scenario $(s_1, \ldots)$.

The above discussion formalised the example intuitively. The formalisation is not completely precise but is good enough to show us what kind of features are at play. Let us highlight them.

1. **The dynamic aspect**
   A practical reasoning situation is not just two people arguing in some logic. Such situations are static. In a more realistic situation the people want to execute or to block actions. Actions have preconditions and postconditions, so the reasoning is stretched over a period of real time and is intimately involved with the actions.
2. Agreed action scenarios and a projected future is also part of the data.[22]
3. The whole model must be procedural and effective. The past must be clearly and algorithmically displayed as to how it was generated by actions. The future is branching by the kind of actions available, together with a planned partial future.[23]
4. The model does not involve possible worlds, only theories Δ and a process of revision.
5. Actions need to be formally recognised and named. Time ticks because actions are taken.[24] Alternative actions generate alternative histories.
6. Sequences of actions can be preconditions and postconditions to other actions.

[22]This is an important point, enabling our model to handle contrary-to-duty obligations. See Gabbay (2008a).

[23]Here non-monotonic logic can interface with decision theory. We can use utility and decision theory considerations to decide on our actions. For example John can estimate his chances of winning a legal claim before deciding whether to make a deal with Mr Rich.

[24]We shall use Gabbay's executable temporal logic model, Barringer et al. (1994).

C.2 The PhD Example

John is registered as a PhD student at the local university. The university used to be a community college and has become a university through some political changes which upgraded all community colleges into universities. They can now award PhD degrees in certain areas. However, the rules and regulations as well as the general feel of the place are still very much like a local community school.

John's advisor is the departmental bigshot Mr Grossman. He is a very conservative person who treats his students like a grammar school master. To get a PhD in Mr Grossman's department one needs to satisfy three conditions:

1. Take three masters courses (without exams but with proper attendance and coursework)
2. Submit a thesis
3. Defend the thesis in an oral exam (viva).

The last two conditions are standard in all major universities. The first condition is to make sure the students continue with their training and appear knowledgeable when released into the world.

John is a hard working and clever student. He is always neat and clean, but is not generally well dressed and does not always show up to Mr Grossman's course. When he does attend the course, he asks questions that Mr Grossman sometimes cannot answer. Mr Grossman is not upset by this; indeed he values John as a good student and tries to help him. In due course, John submitted a thesis and defended it successfully in his oral exam. The college, however, refused to award the degree of a PhD to John on the grounds that John did not fulfil all the requirements. College administration claimed that John did not attend three courses; his attendance of Mr Grossman's course was too low to count as a third course.[25]

The following is a formalisation of the story so far:

Dictionary
C_i = John attends course $i, i = 1,2,3$
T = John completes a thesis
V = John passes oral examination
P = John gets awarded PhD

The university rule B states that:

$$B = C_1 \wedge C_2 \wedge C_3 \wedge T \rightarrow (V \rightarrow P).$$

The very literally minded administration said that P cannot be deduced because C_3 was not made true.

[25] Some colleges do not allow for the oral exam to take place unless the following holds:

1. the courses were attended;
2. fees were paid;
3. all books were returned to the library.

Our practical reasoning situation has to do with arguments trying to establish that John can be deemed as having attended the course.

The course had 22 contact hours delivered during evening time as 11 lectures of two hours each. Being still a community college in practice, students' attendance is logged in and their homework submission properly recorded.

John has not attended the first seven lectures, but did submit the homework. He did attend lectures 8 and 9. During lecture 10 he was ill and he came only briefly to lecture 11 during which he asked Mr Grossman some penetrating questions and forced him to stop the lecture and try to correct his mistakes.

The course material is cumulative (like mathematics) in the sense that one cannot understand lecture 11 without mastering the material of the previous lectures.

On the basis of the above story Mr Grossman was arguing that John should be deemed as having attended the course.

The formulas involved in this story are postconditions of actions. C_3 is the postcondition of the action *attending the courses*, where the preconditions are probably anchored in some regulations. For example, we may have *x attends* a course iff *x* shows up at seven lectures out of eleven and submits all homework with a possibility of missing two more lectures (i.e. attends only five) for reasons of health.

Mr Grossman's argument is probably to bypass the regulations on the grounds that John knows the course material very well, even better than Mr Grossman himself.

I think the proper way of formalising the situation is to use fuzzy logic, giving the predicate 'attend' a fuzzy meaning. The details are not important for us now; the important point to accept is that we have a dynamic action context here and the whole argument is to enable an action to take place, namely making C_3 true, to get the PhD awarded.

The reasoning involved in deriving C_3 from the particular circumstances is non-monotonic. If the College approach is very formal then we cannot accept that John attended the course. If we are allowed a little bit of common-sense reasoning, we can say that John can be deemed to have attended the course since he obviously knows the material and it is the last remaining requirement for his PhD. At this point our reasoning can be non-monotonic and can depend on how much information we have. Recall that the class was an evening class. We might be reluctant to credit John with the course if we have the additional information that he missed the first seven lectures because he spent his evenings in the pub. We may change our minds yet again if we get the further information that John was actually doing community service in the pub (on behalf of the sheriff) making sure that underage teenagers are not served alcoholic drinks.

In general, if we are prepared not to follow the rules formally and mercilessly we will use our common-sense and consider all the data about John and make a decision. If Δ is the data about John and the course, we will use our common-sense to decide whether $\Delta|\sim C_3$ or not.

We now conclude the PhD example. We shall look at it again later in the action proof theory section.

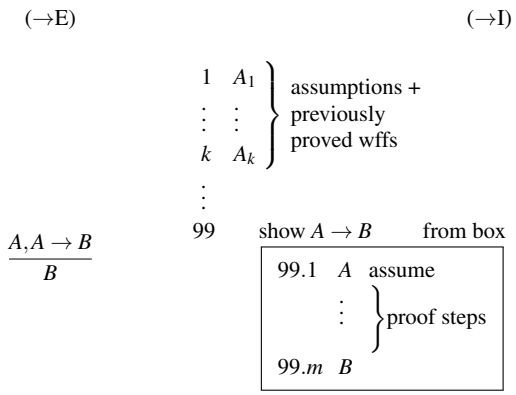


Fig. 3.22 Traditional proof theory for $\to$

C.3 Simple Non-monotonic Oscillating Proof Theory

Equipped with an effective expectation non-monotonic consequence $|\sim$ and with an effective natural deduction monotonic system, we can formulate an effective non-monotonic proof theory. We need one more component for our proof theory. We need a list of actions, with their preconditions and postconditions. These actions will be intimately involved in the proof process.

The intuitive idea is that proofs stretch over time and the deductive steps also involve taking actions that generate postconditions that can be used to continue the proof. All of this is done against a stream of incoming data which may non-monotonically discredit the assumptions. We call this kind of phenomena oscillating proof theory. We shall explain our proof theory in stages, using progressively more complex examples.[26]

Our starting point is a language with $\to$ alone and we assume we have the natural deduction $\to$ Elimination ($\to$E) and $\to$ Introduction ($\to$I) rules for $\to$.

The traditional use of these rules can be summarised in Fig. 3.22.

It is helpful for our examples of this section to think of $\to$ as intuitionistic implications. There are two reasons for that. The first is that the rules of Fig. 3.22 actually define the intuitionistic $\to$. This, however, is not our main reason. The intuitionistic $\to$ has, as one of its interpretations, the open future temporal interpretation, where we have a branching future time and we reads $X \to Y$ as 'whenever in the future X becomes true we must also have that Y is true'. This is the **S4** reading of $\to$ as $\Box(X \supset Y)$, where $\supset$ is material implication.

[26]Think of the last G. W. Bush – Al Gore election controversy over Florida. It stretches over time, many players take actions and it involves a lot of legal as well as practical common-sense reasoning.

This view suits us very well as we are going to read $X \to Y$ as 'for any sequence of future actions if X becomes available then Y can be deduced'.[27]

The idea of the non-monotonic proof procedure (without actions) is to allow, in the middle of the monotonic steps, also some non-monotonic ones. To do these non-monotonic steps correctly, we need to know exactly at each stage of the proof what is the official database at that stage so that we know what non-monotonic deductions are available.

This idea is best illustrated in an example.

C.3.1 Non-monotonic Proofs Without Actions

We choose a variation of our thesis example of Sect. C.1.

We begin with a dictionary and some monotonic and non-monotonic data.

Dictionary:
T = complete thesis
V = pass oral examination
P = get awarded PhD
A = get assistant's job
I = get instructor's job
W = be well dressed

Data:
β = university regulation $T \to (V \to P)$
$\mathsf{N}_1 = \beta \wedge T \mid\!\sim A$
$\mathsf{N}_2 = \beta \wedge T \wedge V \mid\!\sim I$
$\mathsf{N}_3 = \beta \wedge T \mid\!\sim \neg I$
$\mathsf{N}_4 = \beta \wedge T \mid\!\sim W$
To prove:
$\alpha = T \to (A \wedge (V \to P \wedge I))$

Figure 3.23 shows how α can be proved non-monotonically.

Line 2.2 derives A non-monotonically from the official database containing $\{T\}$ alone. When we add assumption 2.5.1, the official database is $\{T, V\}$. A is no longer non-monotonically derivable. At stage 2.5.1 when V is about to be added to the data available at stage 2.4, the database is as follows:

$$\Delta_{2.4} = \{T, A, \neg I, W\}.$$

T is official data. $A, \neg I$ and W are derived non-monotonically. When we add V as input we need to *revise* $\Delta_{2.4}$. We may adopt a sceptical policy and have $\Delta_{2.5.1}$ be just $\{T, V\}$ i.e. drop out all the non-monotonic consequences or we can adopt a more

[27] Our use of words here is not precise. See Gabbay (2001) and *Agenda Relevance*, in Gabbay and Woods (2003/2005).

show $\alpha = T \to (A \wedge (V \to (P \wedge I)))$ from box

1	$T \to (V \to P)$		data
2	show $T \to (A \wedge (V \to (P \wedge I)))$		from box
	2.1	T	assumption
	2.2	A	from (1), (2.1) and $\mathcal{N}_1$
	2.3	$\neg I$	from (1), (2.1) and $\mathcal{N}_3$
	2.4	W	from (1), (2.1) and $\mathcal{N}_4$
	2.5	show $V \to P \wedge I$	from box
		2.5.1 V	assumption
		2.5.2 I	from (1), (2.1), (2.5.1) and $\mathcal{N}_2$, (2.3) and (2.4) are ignored
		2.5.3 $V \to P$	from (2.1) and (1)
		2.5.4 P	from (2.5.1) and (2.5.3)
		2.5.5 $P \wedge I$	from (2.5.2) and (2.5.4)
	2.6	$A \wedge (V \to (P \wedge I))$	from (2.2) and (2.5)

Fig. 3.23

tolerant, compromise policy, and keep the non-monotonic data as a low priority *defeasible* data, to be thrown out only if they contradict any newly derived data (in the 2.5 box).

According to this policy $\Delta_{2.5.1}$ has the form $\{T, V$ hard data; $A, \neg I, W$, defeasible data$\}$.

$\Delta_{2.5.2}$ will be $\{T, V$, hard data, I defeasible high priority; A, W defeasible low priority$\}$.

In $\Delta_{2.5.2}$ we threw out $\neg I$ because it contradicts the non-monotonically derived I at 2.5.2.

Obviously we need a notion of a structured database and an operation of revision of such databases relative to incoming input. Both the data and the input need to be labelled, the label indicating how they are proved and what their priority is. This will enable us to calculate at each line $(x_1, \ldots, x_n)$ in the proof what is the database $\Delta_{(x_1,\ldots,x_n)}$.

C.4 Oscillating (Action) Proofs

Let us now turn our attention to what we shall call *oscillating proof theory*, where actions are also involved and the assumptions available keep on changing. Imagine that at state (time) w_0 we have a database Δ_0 with various data and non-monotonically derived conclusions including the following

$$t_1 : A$$
$$t_2 : B \to (A \to C).$$

We need to get B in order to derive C.

The labels t_1 and t_2 give us information about the status of A and of $B \to (A \to C)$ in the database.

Imagine now that among the actions available to me there is an action **a**, whose precondition is D and postcondition is B.[28] If I can prove D from Δ_0, I can execute action **a**, get the postcondition B and assuming the 'update' B does not discredit (revise out) the assumptions A and $B \to (A \to C)$ then I shall be able to derive C. Our oscillating (action) proof shall then be the following:

Data

1. Δ_0
2. $t_1 : A$
3. $t_2 : B \to (A \to C)$
4. Action **a** with $\alpha_{\mathbf{a}} = D$ and $\beta_{\mathbf{a}} = B$.

To prove: $x : C$ (C with some label x).

Proof.
Steps $1 \ldots k$: Show D
Step $k+1$: Execute action **a**
Step $k+2$: Update current theory with $\beta_{\mathbf{a}} = B$.
Step $k+3, k+4, \ldots$: Assuming our two data items $A, B \to (A \to C)$ survive the update of step $k+2$ we can deduce C.

Note that we are assuming that the proofs take real time (as in a legal process) and that during that time actions can be executed, yielding their postconditions as data. The new data may revise out some existing assumptions. So if we are unlucky, some other 'agency' may execute an action between steps $k+1$ and $k+2$ which will revise A out and we will not be able to get C.

Thus we should prove our conclusions 'quickly' before our assumptions 'disappear'. Also notice that we listed the actions as part of the data; available actions are data since when executed they can bring in their postconditions.[29]

Note that after revision our theories remain $\vdash$ consistent. Thus the revision is carried out in the $\vdash$ logic (or a labelled version of this logic). When we use labels we can allow our theories to be inconsistent and thus have the option of not revising, but living with inconsistency. See Gabbay and Hunter (1991, 1993).

Let us now turn to a more complex example.

Consider the vocabulary of Sect. C.3.1, and let us assume that in order to take the oral examination (V) the student needs to take the course C. C is a precondition for the action of taking the exam. Our analysis of the situation will be a simplified one, as we only want to illustrate for now the kind of action proof theory needed.

The full development of our model will be done later.

---

[28] Our version here is still a simplified one. In general the pre- and postconditions for actions must be labelled and may contain other actions.

[29] In my paper on abduction (Gabbay 1991a) and in my *LDS* book (Gabbay 1996), I have stressed that one can put in databases as 'data' not only pure data but also mechanisms for obtaining data such as abduction processes or (in our case) actions.

The actions involved are **c** (*take the course*), **v** (*take the exam*), and **t** (*submit a thesis*).

There are no preconditions for **c** and the postconditions can be C but not necessarily so. Actually the postcondition is $t : C$, where t is a label showing among other things, that the university administration accepts C into the databases. There are no preconditions for **t** (submitting a thesis) and the postcondition can be T (completed thesis) if it is approved, but there is no guarantee of that.

The precondition for **v** is $t : C$ and a possible postcondition is $t' : V$, where t' is a label containing the relevant information. Again, there is no assurance that we get V, i.e. that the student will pass the exam.

Note that these actions are non-deterministic in their postconditions. The postcondition Y may not obtain and we may get $\neg Y$. This is a new twist to what we had before in earlier sections but it is more realistic.

We are now ready to do our proof theory. Our starting state (time) is w_0. At this time we have available as data the following:
Θ_{w_0}: *Data at time* w_0.

1. Action data: **t**, **c**, **v**.
2. Proof theoretic data: β, N_1–N_4, as in the beginning of Sect. C.3.1.

At time w_0 we are able to construct the derivation of Fig. 3.23 and have available

$$\alpha = T \rightarrow (A \wedge (V \rightarrow (P \wedge I))).$$

The meaning of $\rightarrow$ is temporal. $X \rightarrow Y$ means that whenever X is in the database (is available) then Y is available. This makes $\rightarrow$ like intuitionistic implication, but not exactly so. The database is non-monotonic and so may have things thrown out of it, when X is put into it. Moreover, the 'whenever' is not semantic but syntactic. X becomes available as a postcondition of some action and not as part of some semantic evaluation in a Kripke model.

John (our student) is interested in making P true, so he needs to have T. He executes action **t** and moves to state $w_0\mathbf{t}$ and hopefully T becomes true. Non-monotonically it follows that A and $\neg I$ become true (if no other actions are taken simultaneously by other 'agents'). Now John wants to make V true. He needs to have the precondition C for the action **v** to non-monotonically hold so that he can perform **v** and get its postcondition V to (hopefully) become true. So he will perform action **c** and if successful, will perform **v**. Thus at state (time $w_0\mathbf{tcv}$ we will have V holding as well, and we can deduce $P \wedge I$.

Note that we assume that the update with V does not throw out T.

C.5 Proofs with Evolving Non-montonicity

The above considerations and proof theory assume that the non-monotonic rules are fixed and do not change. This is not realistic. Since the proof process stretches

over time and involves actions, it is quite possible that by the time we get to the end of the proof, the rules themselves change. Going back to the PhD example, it may be the case that by the time our student John submits his thesis, the required number of courses to take has been reduced (or increased) from three to two (resp. from three to four). John will argue in case of an increase that he should be exempt from having to take a fourth course because such a course was not required when he started his PhD. This sounds reasonable but not necessarily because of any principle involved. Think what happens in the opposite case. No one can credibly argue, in case the number of courses has been reduced, that John should nevertheless take three courses!

It seems that what is to be done depends on the application at hand. For this reason our proof system and model need to be able to handle changes in non-monotonic rules occurring during the proof process with a built in flexibility for adopting different policies. Once we start thinking along these lines, we realise that really this happens a lot in practice. We remember governments trying to change tax laws to plug loopholes and people rushing to do their bit before the law changes. We can also remember changes in immigration rules and refugees rushing into the country ahead of some deadline. There are many such examples and our system must accommodate such phenomena.

It seems that the most realistic way to handle rule changes in our current model is to allow for actions of the form $\mathbf{n} = (\mathsf{N}_{1,i}, \mathsf{N}_{2,j}), i = 1,\ldots,k; j = 1,\ldots,m$. The action $\mathbf{n}$ replaces the non-monotonic rules $\mathsf{N}_{1,1},\ldots,\mathsf{N}_{1,k}$ by the rules $\mathsf{N}_{2,1},\ldots,\mathsf{N}_{2,m}$. The action $\mathbf{n} = (\varnothing, \mathsf{N}_{2,j})$ simply adds the rules $\mathsf{N}_{2,j}, j = 1,\ldots,m$ and $\mathbf{n} = (\mathsf{N}_{1,i}, \varnothing)$ simply deletes $\mathsf{N}_{1,i}, i = 1,\ldots,k$.

These actions can also have preconditions $\alpha_{\mathbf{n}}$ and postconditions $\beta_{\mathbf{n}}$.

Let us consider the proof of Fig. 3.23 with the added complication that he action

$$\mathbf{n} = (\mathsf{N}_3; \mathsf{N}_6)$$

is available where

$$\mathsf{N}_6 = \beta \wedge T \mid\!\sim I.$$

The rule N_6 says that it is possible to make a student, who submitted a thesis, an instructor and not just an assistant contrary to what rule N_1 says. For reasons of consistency N_3 must be deleted. The action $\mathbf{n}$ has the precondition that the thesis is outstanding (denoted by E). Thus $\alpha_{\mathbf{n}} = E$. There are no postconditions.

Let us now consider what can happen if action $\mathbf{n}$ is taken in the middle of the proof described in Fig. 3.23. Line 2.1 of this proof can be assumed to be the result of action $\mathbf{t}$ = *get thesis*. If immediately after line 2.1 action $\mathbf{n}$ is taken, i.e. rule N_3 is deleted and rule N_6 is added, it is up to us whether we still allow A and $\neg I$ to be deduced or forbid any deduction from N_3 and use only N_6.

It seems reasonable to leave both options open in the formal proof theory model. It also seems reasonable that if action $\mathbf{n}$ is taken before $\mathbf{t}$, then rule N_3 is definitely cancelled.

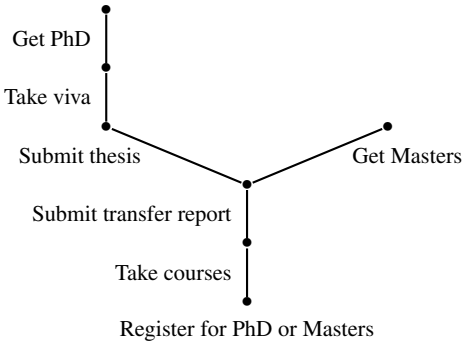


Fig. 3.24

Let us consider therefore the following principle.

- Let N be a rule of the form $A_1 \wedge \cdots \wedge A_k | \sim B$. Then N can be deleted by an action whenever the official database is not $\{A_1, \ldots, A_k\}$. If the official database is $\{A_1, \ldots, A_k\}$ and an action deleting N is taken then it is optional (up to the application area) whether N can still be used or not.

This principle does not completely solve the problem of changing rules. Let us reconsider the PhD example. A student registers for a PhD when the rules require 3 courses, i.e. the rule is (see Sect. C.2);

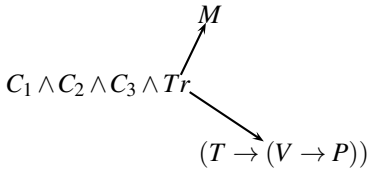
$$B = C_1 \wedge C_2 \wedge C_3 \wedge T \rightarrow (V \rightarrow P).$$

After registration, say when the student has already done C_1, C_2 and was ready to submit his thesis (T) the rules change. The student certainly has not *completed* the sequence of actions required for a PhD. He can still reasonably and forcibly claim he should be treated according to the old rule, which was valid when he *started*!

This kind of argument will not hold for a non-monotonic understanding. If the understanding when he started his PhD was to give an assistantship when a student achieves a thesis, this defeasible non-monotonic understanding can change at any time before the student submitted his thesis (e.g. on lack of budget grounds). The student would have reasons for sympathy if the rule is cancelled after he submitted his thesis, and we may or may not still invoke the rule, depending on circumstances.

The above considerations lead us to consider as action units not single actions but sequences or trees of actions and consider them *committed* at the start of the sequence. Figure 3.24 is an example of action tree. (Each point has a branch leading to failure which we ignore).

If M denotes *Get masters* and Tr denotes *Submits transfer* report, we can write the rule as



or if we use a special *choice disjunction symbol* $\uplus$, we can write

$$C_1 \wedge C_2 \wedge C_3 \wedge Tr \rightarrow (M \uplus (T \rightarrow (V \rightarrow P))).$$

These options will be explored later in our work.

Remark 3.34. The action/proof point of view gives us a way of explaining away a conceptual contamination in Prolog. Prolog is not a purely logical language. To function as a proper programming language it needs to contain imperatives. So one can write clauses of the form

Famous if Thesis $\wedge$ *Print-Thesis*.

To ask the goal *?Famous* one must ask whether *?Thesis* and then go ahead and *print the thesis*. This is not a Horn logic clause. However, we can write in our system the database with the following data at the initial state w_0.

1. Thesis
2. Action *print* with precondition Thesis and postcondition Printed-Thesis
3. Thesis $\rightarrow$ (Printed-Thesis $\rightarrow$ Famous)

We reason as follows at w_0

4. Printed-Thesis $\rightarrow$ famous from (1) and (3)
5. Perform action *print* and so at w_0*print* we get Printed-Thesis
6. At w_0*print* get Famous from (4) and (5).

Thus our Prolog deduction is really the above (1)–(6) done in a goal-directed way.

C.6 Example: Dialogue Logic

Sections C.3.1 and C.4 introduced into the proof theory of monotonic intuitionistic implication, two additional components:

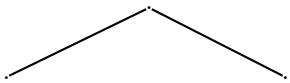
1. the non-monotonic deduction, and
2. the action as a logical move.

To show the plausibility and naturalness of such additions, we recall the well known dialogue semantics or game semantics for intuitionistic logic, see Felscher (1985).

We illustrate this approach via a simple example. To show that $a \wedge b \to a \wedge b$ is a theorem of intuitionistic logic, we need two players, the proponent (say J) and the opponent, (say T). The proponent puts forward $a \wedge b \to a \wedge b$ and the opponent attacks and the game continues according to rules. The proponent has a winning strategy exactly for those formulas which are theorems of the logic.

Here is the game for this case as presented in Felscher (1985, p. 345), using my notation.

0. $a \wedge b \to a \wedge b$, (J)
1. $a \wedge b$, attack of T
2. a, counterattack by J on first conjunct of 1.
3. a, defence by T against 2.
4. b, counterattack by J on second conjunct of 1.
5. b, defence by T against 4.
6. $a \wedge b$, defence against 1 by J.



7. a, T attacks first conjunct of 6	7. b, T attacks second conjunct of 6
8. a, J defends against 7 in view of 3.	8. b, J defends against 7 in view of 5.

The moves of J and T are regulated. It is very easy and natural to allow additional actions in the database and allow additional moves, for example a player x activates action $\mathbf{d}(x)$ and adds postcondition $A(x)$, etc.

It is clear that the new system of Sects. C.3 and C.4, when represented in the dialogue approach, can be viewed as just another dialogue logic with a twist, namely more moves for the players..

One point to clarify. In the dialogue interaction, the 'time' involved is the sequence of moves. In the examples of Sects. C.3 and C.4, the time involved is 'real' time. Does this make a difference? No, it does not. We all know that intuitionistic logic is complete for the Kripke future-time semantics, where at any time $t, t \vDash A \to B$ means 'whenever A becomes true, so does B'. Dialogue moves correspond to semantic tableaux moves against Kripke time model and so can be interpreted as real time.

So in our new kind of logic, modus ponens can take real time and actions can be involved.

C.7 Summary

The following summarises what we need.

Non-monotonic theories can be viewed as comprising of two parts:

1. The hard core official monotonic part Δ_H which is a body of hard (verified, confirmed and non-defeasible) data.
2. The additional defeasible or hypothetical part, which can be added to the hard core part using various conventions and non-monotonic mechanisms which we symbolise by $\mathbb{M}$.

Thus we can represent this situation by writing:

$$\Delta = \Delta_H + \mathsf{M}$$

The non-monotonic principles involved in M can vary from one application to another. M can contain a variety of rules such as emergency regulations, abductive mechanisms which add explanation data to the hard data of Δ_H or belief revision rules (which change Δ_H by adding a hypothetical additional assumption A and change Δ_H to maintain consistency). It is for this reason that we formally presented Δ as $\Delta_H + \mathsf{M}$, where M represents the mechanisms involved and $+$ represents the application of M to Δ_H.

To develop a proper proof theory in such a context we need the following machinery:

1. We need a clear concept of a *logical system*, able to accommodate monotonic and non-monotonic mechanisms and backed by solid intuitions.
2. We need a good theory of non-monotonicity, which is compatible with the notion of logical system.
3. We need an executable imperative logical action theory.
4. We need a suitable notion of a logical database (comparable to what is traditionally known as logic theory) which may be different from the traditional notion (of a theory as a set of formulas) and which is more suitable for our proof theory. We shall see that we need to take our databases as labelled sets of wffs or as sequences of formulas at the least, or in general any structure imposed by an application. The database must contain actions available to be executed if the preconditions hold and that will generate the postconditions.
5. We need various algorithms for belief revision for modelling the conditional. The belief revision part deals with adding the antecedent of the conditional into the database.
6. We need defeasible reasoning principles to deal with conflicting evidence and proofs.
7. We need a suitable monotonic proof methodology which can be easily and naturally modified to accommodate (1)–(6) to form our final proposed non-monotonic and conditional proof system.

8. In addition to the above we need to start with an initial simplified intuitive vision (image/picture) in our mind of how the non-monotonic and conditional reasoning is taking place in real practical situations.

Appendix D. Labelled Revision and Actions

D.1 Compromise Revision

We saw in previous sections that an algorithmic revision process is needed to be applied whenever we add something to the database. This revision process needs to be tightly controlled and the labelled theory of compromise revision presented in this appendix does exactly that, see Gabbay (1999) and Gabbay et al. (2010).

We begin with an example to show what is needed.

Example 3.35. Let us consider the data in Box 2.1 in the proof of Fig. 3.23. We label the data according to how they were derived as follows:

(1) $h_0 : T \to (V \to P)$ (hard assumption)
(2.1) $h_1 : T$ (hard assumption)
(2.2) $\mathsf{N}_1 h_0 h_1 : A$ (obtained from h_0, h_1 using rule N_1)
(2.3) $\mathsf{N}_3 h_0 h_1 : \neg I$
(2.4) $\mathsf{N}_4 h_0 h_1 : W$

When we add to the hard assumption(input)

(2.5.1) $h_2 : V$

we get an inconsistent database, because we can derive

(2.5.2) $\mathsf{N}_2 h_0 h_1 h_2 : I$

This contradicts (2.3). So some database revision is required.[30] In Fig. 3.23 we intuitively said that (2.1) and (2.4) are ignored since they were non-monotonically derived from the smaller hard database $\{h_0, h_1\}$ and we now have a new database $\{h_0, h_1, h_2\}$. This is a sceptical view. A more compromising view may wish to retain as much as possible from the database, thus retaining (2.4) but rejecting (2.3).

The story can be further complicated if we had the following additional non-monotonic rule

N_5 $\beta \wedge T \wedge V \wedge W \mid\!\sim R$

[30] We can allow the database to remain inconsistent as long as we know what actions to take in response to the inconsistency. Database revision is only one option.

where R is:

R get radio interview.

In other words, a well dressed student with a PhD gets a radio interview.

Formally, W is not a hard fact and is not derivable from the hard facts of the new database. Having agreed to adopt the compromise view and accept that W still holds in the new database, can we now deduce

(2.5.6) $\mathsf{N}_5 h_0 h_1 h_2 (\mathsf{N}_4 h_0 h_1)$: R

Suppose we agree and say yes, we accept R. What if we have another hard input

$$h_3 : \neg R.$$

Do we now, to maintain consistency, just block the derivation of (2.5.6) or do we throw out (2.4) (i.e. not allow W to persist) because it leads to inconsistency?

Obviously in the general case we need to linearly order the data according to some priority, dependent on the labels, and have an algorithm of what to do (throw out).[31]

Let us arrange the data according to decreasing strength from top to bottom with later input being stronger than earlier input. The more specific non-monotonic deduction having higher priority than the specific ones. We get the following:

$$\begin{aligned}
&h_3 : \neg R\\
&h_2 : V\\
&h_1 : T\\
&h_0 : T \to (V \to P)\\
&h_0 h_1 : V \to P\\
&h_0 h_1 h_2 : P\\
&\mathsf{N}_2 h_0 h_1 h_2 : I\\
&\mathsf{N}_5 h_0 h_1 h_2 (\mathsf{N}_4 h_0 h_1) : R\\
&\mathsf{N}_1 h_0 h_1 : A\\
&\mathsf{N}_3 h_0 h_1 : \neg I\\
&\mathsf{N}_4 h_0 h_1 : W
\end{aligned}$$

We need not, at this stage, specify an exact algorithm for strictly linearly ordering all the assumptions. It is sufficient to say that any formula provable from atoms by the rules gets a label and these can be linearly ordered.[32]

---

[31] Note that we never actually throw things out, we only mark them as *not active*. In the next round of inputs it may be possible to activate them again as the original reason for inconsistency may itself disappear.

[32] The official data (hard facts) can be linearly ordered with the latest incoming data item having higher priority. The non-monotonic rules can be linearly ordered according to the nature of the application at hand. Given this all deduced formulas are resource derivation labelled using the label propagation rules, e.g.

If $\varphi_1, \ldots, \varphi_k$ is the ordered database in increasing strength then we can define a consistent revision of it by induction as follows.

$\Delta_k = \{\varphi_k\}$ if classically consistent,[33] otherwise $\Delta_k = \varnothing$.

Assume Δ_j is defined, let $\Delta_{j-1} = \Delta_j \cup \{\varphi_{j-1}\}$ if classically consistent and $\Delta_{j-1} = \Delta_j$ otherwise.

The above process is valuable and fruitful also in ordinary classical belief revision for monotonic systems. Consider the labelled database Δ below

$$\Delta = \begin{cases} t_i = A \to C \\ t_j : \neg A \to B \\ s : A \end{cases}$$

and let the input be

$$x = r : \neg A.$$

The new database is inconsistent in classical logic and can prove everything. The labels keep control of the proof process and we know that

$$\Delta + x \vdash \begin{cases} t_i s : C \\ t_j r : B \end{cases}$$

We can linearly order the formulas we get and apply our revision algorithm. This will throw out A, but compromise on keeping C.

D.2 Actions and the Flow of Time

The discussion in Sect. C.4 clearly indicates that we should view time as progressing forward by virtue of the actions being taken. If w denotes *now* then the next moments will have the form $w\mathbf{a}_1, \ldots, \mathbf{a}_n$, where $\mathbf{a}_i, i = 1, \ldots, n$ are some actions taken simultaneously. Our view is similar in spirit to that of Ray Reiter (2002).

Recall the actions **c**, **v** and **t** of Sect. C.4 and let $\mathbf{a}_1, \mathbf{a}_2$ be some arbitrary actions which may take place simultaneously with the above three actions. Then Fig. 3.25 illustrates one possibility of how our future could be at w.

$$\frac{\alpha : A; \beta : A \to B}{\beta\alpha : B}$$

We can compare labels using the following principles:

1. Formulas proved from a bigger set of hard assumption have higher preferences (being more specific).
2. For the same hard facts specificity we can look at more non-monotonic rule specificity,
3. If all is equal we can order lexicographically on the labels.
4. Same labels mean same proof, i.e. same formula, so our ordering is complete.

[33] We can use any other notion of consistency here, as long as it is decidable.

c = take a course
preconditions: none
postconditions: C

t = get thesis
pre-condition: none
postcondition: *T*

v = take exam
pre-condition: C
postcondition: *V*

$\mathbf{a}_i$: other actions *i*

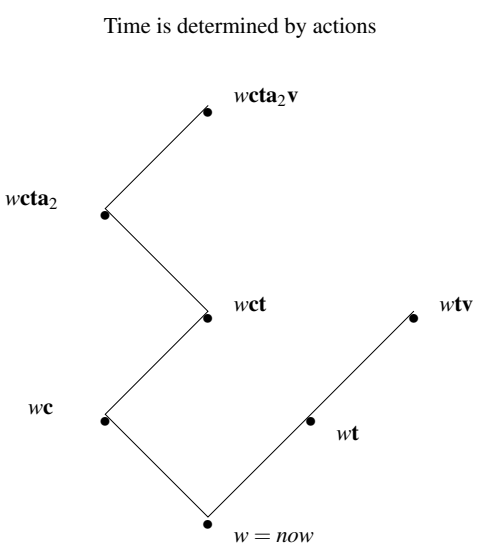


Fig. 3.25

Let Δ_x be the theory associated with state (moment) x. Then if action **a** is taken at state x and $\beta_{\mathbf{a}}$ are the postconditions of **a**, we have that $\Delta_{x\mathbf{a}} = \Delta_x + \beta_{\mathbf{a}}$, where '+' denotes our compromise revision process.

Here are some examples:

$$\Delta_w = T \to (V \to P)$$
$$\Delta_{w\mathbf{t}} = T \to (V \to P), T, \neg I, W$$
$$\Delta_{w\mathbf{ctv}} = T \to (V \to P), T, V, I, \text{ (maybe) } W, C.$$

In general, $\Delta_{x\mathbf{a}} = \Delta_x +$ postconditions of **a**.

Thus the flow of time is discrete, branching into the future and linear into the past. Time moves forward by actions being taken, and thus we have:

- $t < s$ iff for some n and some sequence of actions $\mathbf{a}_1, \ldots, \mathbf{a}_n$ we have $s = t\mathbf{a}_1, \ldots, \mathbf{a}_n$.

Since in real applications people make agreements about sequences of actions in the future, we must allow, at each time (state) point t for a planned and agreed future path H_t. H_t is part of the state description (data). Thus a time flow around w = now may look like Fig. 3.26.

Note that theory at time w, Δ_w contains data on planned and agreed future. This is important because part of everyday practical reasoning now takes into account agreed commitments for the future of now.

From the point of view of $w =$ now, the past is linear; it is what has actually happened. See Fig. 3.28. The two figures, Figs. 3.26 and 3.28 can be combined together.

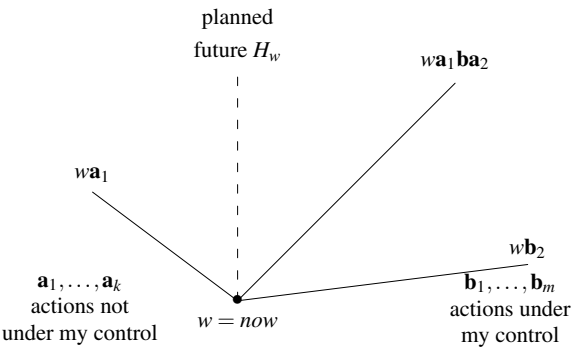


Fig. 3.26 Future Options

A more realistic action/time model must allow the actions' precondition to query the past and allow the action's postconditions to be non-deterministic and include both formulas and further actions to be executed. To query the past (for the preconditions) we need a temporal language, say using *Since*, *Yesterday* and names of moments of time $w_1, w_2, w_3, \ldots, w_n$ (w_n means 'the time now is w_n'), etc. The temporal truth table for $Since(A,B)$ also written $S(A,B)$, is

- $t \vDash S(A,B)$ iff for some $s < t, s \vDash A$ and $\forall y(s < y < t \rightarrow y \vDash B)$.
- $t \vDash YA$ iff predecessor of $t \vDash A$.
- $t \vDash w_n$ iff $t = w_n$.

Given an action $\mathbf{a}$, its precondition $\alpha_{\mathbf{a}}$ is a Boolean combination of formulas built up from the language of the theories involved using connectives and temporal operators. At moment t, $t \vDash \alpha_{\mathbf{a}}$ enables the action $\mathbf{a}$. The postcondition $\beta_{\mathbf{a}}$ of the action is a non-deterministic outcome being a set $\beta_{\mathbf{a}}$ of pairs of the form $\beta_{\mathbf{a}} = \{A_i, \{\mathbf{b}_1^i, \ldots, \mathbf{b}_k^i\})$, where A_i is a wff to be added as an update input into the theory Δ_t and $\{\mathbf{b}_1^i, \ldots, \mathbf{b}_{k_i}^i\}$ is a set of actions committed to be performed, if the outcome of the action is the ith non-deterministic possibility.

To have such a model we must assume that at each moment of time t, we have not only a theory Δ_t but also a set $\mathbf{A}_t$ of actions committed to be performed at t. We also assume that the passage from moment t to the next moment $t\mathbf{a}$ is done by performing an action $\mathbf{a} = (\alpha_{\mathbf{a}}, \beta_{\mathbf{a}})$ such that $\Delta_t | \sim \alpha_{\mathbf{a}}$ and such that $(A_i, \{\mathbf{b}_1^i, \ldots, \mathbf{b}_{k_i}^i\}) \in \beta_{\mathbf{a}}$ and $\Delta_{t\mathbf{a}} = \Delta_t + A$ and $\mathbf{A}_{t\mathbf{a}} = (\mathbf{A}_t - \{\mathbf{a}\}) \cup \{\mathbf{b}_1^i, \ldots, \mathbf{b}_{k_i}^i\}$.

The choice of which action $\mathbf{a} \in \mathbf{A}_t$ to execute can be made either arbitrarily or using some decision theory considerations as discussed in Sect. D.3.

We note of course that the future can be unpredictable and non-deterministic. This can be generated in our model by the following devices.

- Actions can be assumed to have non-deterministic outcomes.
- We assume several agents working together or in opposition taking independent actions.
- Attribute random actions to the underlying model itself (nature).

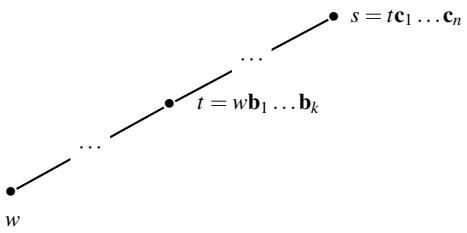


Fig. 3.27

The aspect of this unpredictability we want to focus on is the way the future changes.

Figure 3.27 describes a future of $w =$ now determined by all possible sequences of actions. If all is deterministic, then when we take action $\mathbf{t}$ and move to node $w\mathbf{t}$, the future as seen from $w\mathbf{t}$ is the same as the future of $w\,\mathbf{t}$ as seen from $w =$ now. If, however, the future is unpredictable and non-deterministic, then when we perform action $\mathbf{t}$, and move from w to $w\,\mathbf{t}$, we find that our options for the future have changed and the future as seen from $w\mathbf{t}$ (after having moved there by taking action $\mathbf{t}$) is no longer the same as what we have expected before taking action $\mathbf{t}$ (i.e. when we were at w).

For example, action $\mathbf{v}$ may not be available, and thus the node $w\mathbf{tv}$ is no longer accessible to node $w\mathbf{t}$.

The above discussion is an example of what we call *reactive semantics*. When we move from world w to world $w\mathbf{t}$, the existing connection between $w\mathbf{t}$ and $w\mathbf{tv}$ get cut off. So the model changes from under us as we move along it.

This idea turned out to be very fruitful in many application areas, including contrary to duties, modal logic, automata theory and more, see Crochemore and Gabbay (2011), Gabbay (2008a), and Gabbay (2008b).

D.3 Decision Theory

We saw in Fig. 3.26 that we face a branching future, depending on which actions we choose to execute. The actions we take do not have a deterministic outcome but we can estimate intuitively how likely they are to give us the desired outcome. At this point we can import a decision theory model into our system to help us choose the actions we want to execute. For reference, see Jeffrey (1983).

We start by giving some definitions to fix our notation.

Definition 3.36 (Non-deterministic dynamical systems). A dynamic system s has the form $\mathcal{D} = (S, \mathbf{A}, \mathbb{M}, U, \mathbf{f})$ where S is a (possibly infinite) non-empty set of states, $\mathbf{A}$ is a (possibly infinite) non-empty set of action names (or words), $\mathbb{M}$ is a transition function, associating with each pair $(s, \mathbf{a}) \in S \times \mathbf{A}$ a subset $\mathbb{M}(s, \mathbf{a}) \subseteq S$. U is a utility function associating with each triple $(s, \mathbf{a}, s')$ in $S \times \mathbf{A} \times S$ a real valued number

$U_{(s,\mathbf{a})}(s')$ and $\mathbf{f}$ is a family of probability density functions associating for each $(s,\mathbf{a})$ in $S \times \mathbf{A}$ the density $\mathbf{f}_{(s,\mathbf{a})}$ on $\mathbb{M}(s,\mathbf{a})$, i.e. for each $t \in \mathbb{M}(s,\mathbf{a}), \mathbf{f}_{(s,\mathbf{a})}(t)$ is a real number between 0 and 1.

We require that

$$\sum_{t \in \mathbb{M}(s,\mathbf{a})} \mathbf{f}_{(s,\mathbf{a})}(t) = 1.$$

There are two ways of looking at our system:

1. As a non-deterministic automaton operating on states S and alphabet $\mathbf{A}$ and outputting the functions $U_{(s,\mathbf{a})}$ and $\mathbf{f}_{(s,\mathbf{a})}$.
2. As a context dependent decision system under risk and uncertainty, with states S, actions $\mathbf{A}$, and context dependent utility and probability functions.

The reader should note that there are differences between our use of the Decision Theory model and the traditional use of it. The traditional use gives you a choice of actions $\mathbf{a}_1, \ldots, \mathbf{a}_k$ and some attributes against which to measure the utility of the outcomes of the actions with a view of making a choice. There is no other context. In our case we have a rich practical reasoning context with the actions serving as means of introducing postconditions and influencing the proof process. It is a much more complex environment and it may turn out that the notion of maximising utility is not central in our set up, and that the predominant feature in choosing a sequence of action may be rooted in the proof process.

D.4 Case Study: The Conditional

Our model can now give formal semantics for the conditional. For the purpose of this section, let us distinguish between conditionals involving time and straightforward timeless conditionals. For the timeless conditionals, the proof theory and the revision mechanism already developed above can already provide a reasonable model. We can safely use the Ramsey Test for the conditional symbol $>$. The language of previous section contain the implication $\rightarrow$ not the corner $>$. We have two options. One is to add the $>$ to the language or we can use $\rightarrow$ as our conditional.

In the first option we simply adopt the Ramsey Test (RT) below:

(RT) $\quad \Delta|\sim A > B$ iff $\Delta + A|\sim B$

Gärdenfors' triviality result is not applicable to our case because our theories are non-monotonic, and we do not expect the Gärdenfors postulate below (K^*4) to hold.

(K^*4) If A is consistent with Δ then $\Delta \cup \{A\} \subseteq \Delta + A$.

See Gärdenfors' book (1988, Chapter 7).

If we wish to use $\rightarrow$ itself as a conditional, we need to investigate its properties. We need not worry about any triviality result even if we impose (RT), because of non-monotonicity. Furthermore, the proof theory as presented in Sect. 3.1 does not satisfy (RT) anyway.

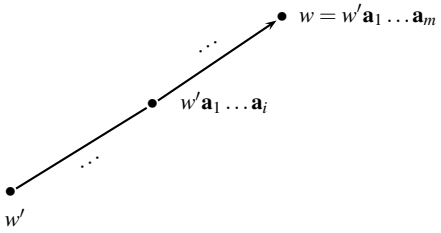


Fig. 3.28

Consider the following database:

$$\Delta = \{B \to (\neg A \to C), A, A \to B\}.$$

Clearly $\Delta \vdash \neg A \to C$.

However $\Delta + \neg A$ may not prove C unless we have compromise revision which allows for B to remain in the revised database.

Thus although by definition our proof process satisfies

- $\Delta + A |\sim B$ implies $\Delta |\sim A \to B$

the converse may not hold. We may have $\Delta |\sim A \to B$ but $\Delta + A \not|\sim B$.

The proper general setting for the conditional must involve time. We address this case next.

In the literature, there is a distinction between two main types of conditionals, the indicative and the subjunctive. The indicative has the form

- If A then B

which we understand as follows (where w = now).

- If A will be true at point t (in the future of w) then B will be true at point s (in the future of t or at t itself).

The subjunctive conditional has the form

- If A had been true then B would be [would have been] true,

which we understand to mean in our model as follows:

- If A were true at w' (in the past of w) then B would be [would have been true] at w [at t in the future of w].

We must remember that all points of our model are results of actions. Thus in the analysis of the indicative conditional we have a situation as in Fig. 3.27, where n may be 0, i.e. $s = t$ or $k = 0$, i.e. $w = t$ or both.

In the case of the subjunctive conditional we may have the case of Fig. 3.28.

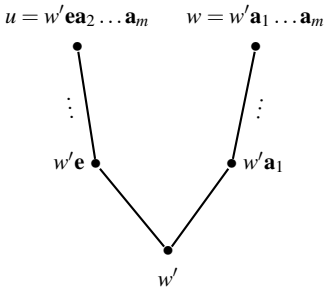


Fig. 3.29

To check whether the conditional

'if A were true at w', then B would have been true at w'

holds at w, we look at $\Delta'_{w'} = \Delta_{w'} + A$ and check whether $(\Delta'_{w'})\mathbf{a}_1, \ldots, \mathbf{a}_m |\sim B$.

Similarly to credibly say at w:

If A at t then B at s

we must have a vision of a sequence of actions that can credibly lead from t to s and $\Delta_s |\sim B$ if $\Delta_t |\sim A$.

We are not saying that the above are the only type of conditionals, we just wanted to show the reader what kind of semantics our model can give the conditional. The appropriate chapter in the book will study the conditional in more detail.

For example our action model allows us to give semantics to

- If A were true at w' I would not have done $\mathbf{a}_1$ but would have done $\mathbf{e}$ instead, and I would have had B now.

The above suggests an alternative history as in Fig. 3.29, where u is an alternative point to now (same distance from w').

Figure 3.29 does not show the whole story. In the alternative history, the theory at w' is $\Delta_{w'} + A$ and hence Δ_u is the result of applying the action sequence $\mathbf{e}\mathbf{a}_2 \ldots \mathbf{a}_m$ to $(\Delta_{w'} + A)$.

The revision involved in such considerations and set up is more complex than the traditional one. We are revising entire histories; namely several theories together. First we do a traditional single theory revision, moving from $\Delta_{w'}$ to $\Delta_{w'} + A$. Then we want to apply action $\mathbf{e}$. This is not so simple. What if the preconditions $\alpha_{\mathbf{e}}$ of $\mathbf{e}$ do not hold? Are we expected to further revise $\Delta_{w'} + A$ by $\alpha_{\mathbf{e}}$ to enable action $\mathbf{e}$, or are we supposed to apply the action $\mathbf{e}$ anyway?, and just further revise $\Delta_{w'} + A$ by $\beta_{\mathbf{e}}$. What if $\alpha_{\mathbf{e}}$ is inconsistent with A? What do we do when upon revising by A some postconditions of previous actions disappear?, do we propagate the effects

backwards into the past? All these questions and more will have to be addressed in future work.[34]

We will just give one example here.

Consider a configuration of states as in Fig. 3.18 and assume further that $w' = w''\mathbf{d}$, where action $\mathbf{d}$ has as postcondition $\neg A$. If we revise and replace $\Delta_{w'}$ by $\Delta_{w'} + A$ then do we further revise the history and pretend that $(\Delta_{w'} + A)$ was obtained from $\Delta_{w''}$ by some other action whose postcondition yields upon revising $\Delta_{w''}$ the theory $\Delta_{w'} + A$?

How far do we propagate our 'conditional input'?

Obviously the problem arises because we have here multiply connected nodes and we need guidelines on how far to propagate the interference. There is a way around the difficulties using the method of *revision through translation*, see Gabbay et al. (2010). We can translate our entire model into a suitable theory τ in classical logic, (which describes all the details of our model). If we let $*$ denote the translation and $\Delta^* \circ C$ denote a suitable belief revision operator in classical logic, then we can endow a belief revision operation $+$ in our model through the following definition:

- $\Delta + A = (\text{def})\{B \mid \Delta^* \circ (A^* \wedge \tau) \vdash B^*\}$.

In other words we do the revision in the classical logic translation and bring back the result.

References

Allen, C. 2001. *A practical guide to evidence*, 2nd edn. London: Cavendish Pub.

Anderson, A.R., and N.D. Belnap. 1975. *Entailment*. Princeton: Princeton University Press.

Barringer, H., M. Fisher, D. Gabbay, R. Owens, and M. Reynolds. 1994. *The imperative future*. Chichester: Research Studies Press.

Basin, D., M. D'Agostino, D. M. Gabbay, S. Matthews, and L.K. Vigano, eds. 2000. *Labelled deduction*. Norwell: Kluwer.

Carnielli, W., M. Coniglio, D. Gabbay, P. Gouveia, and C. Sernadas. 2007. *Analysis and synthesis of logics*. Amsterdam: Springer.

Crochemore, M., and D. Gabbay. 2011. Reactive automata. *Information and Computation*. DOI: 10.01.16.j.ic.2011.01.002, pp. 692–704.

Cross, Sir Rupert. 1999. *Sir Rupert Cross on evidence*. Butterworth: Oxford, UK.

van Eemeren, F.H., and R. Grootendorst. 1992. *Argumentation, communication and fallacies*. Hillsdale: Lawrence Erlbaum Associates.

Felscher, W. 1985. Dialogues as a foundation for intuitionistic logic. In *Handbook of philosophical logic*, vol 3, ed. D. Gabbay and F. Guenthner. Dordrecht: Kluwer. Also in Volume 5 of the second edition, 2002.

[34] Consider

- If I were older and more mature I would have accepted the marriage proposal from John
- John would not have proposed to you in that case.

Here there is a conflict of preconditions.

Fitting, M. 1983. *Proof methods for modal and intuitionistic logic*. Dordrecht: Kluwer.
Gabbay, D.M. 1981. *Semantical investigations in Heyting's intutionistic logic*. Dordrecht: D Reidel.
Gabbay, D.M. 1985. Theoretical foundations for non monotonic reasoning. In *Expert systems, logics and models of concurrent systems*, ed. K. Apt, 439–459. Berlin: Springer Verlag.
Gabbay, D.M. 1991a. Abduction in labelled deductive systems, a conceptual abstract. In *ECSQAU 91*, ed. R. Kruse and P. Siegel, 3–12. LNCS 548. Heidelberg: Springer Verlag.
Gabbay, D.M. 1991b. Theoretical foundations for non monotonic reasoning. Part 2: Structured non-monotonic theories. In *SCAI '91, Proceedings of the Third Scandanavian Conference on AI*, pp. 19–40. Amsterdam: IOS Press.
Gabbay, D.M. 1991c. Modal and temporal logic programming II. In *Logic programming—Expanding the horizon*, ed. T. Dodd, R.P. Owens, S. Torrance, 82–123. New York: Ablex.
Gabbay, D.M. 1992a. Theory of algorithmic proof. In *Handbook of logic in theoretical computer science*, Vol. 1, ed. S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, 307–408. Oxford: Oxford University Press.
Gabbay, D.M. 1992b. How to construct a logic for your application. In *Proceedings of the 16th German AI Conference, GWAI 92*, pp. 1–30. LNAI 671. Berlin: Springer.
Gabbay, D.M. 1992c. Modal and temporal logic programming III, Metalevel features in the object language. In *Non-classical logic programming*, ed. L.F. del Cerro and M. Penttonen, 85–124. Oxford: Oxford University Press.
Gabbay, D.M. 1993a. Labelled deductive systems and situation theory. In *Situation theory and applications*, Vol 3, ed. P. Aczel, D. Israel, Y. Katagin, and S. Peters, 89–118. Stanford: CSLI.
Gabbay, D.M. 1993b. A general theory of structured consequence relations. In *Substructural logics*, ed. P. Schröder-Heister and K. Dosen, 109–151. Studies in Logic and Computation. Oxford: Oxford University Press.
Gabbay, D.M. 1996. *Labelled deductive systems*, Vol. 1. Oxford: Oxford University Press.
Gabbay, D.M. 1998. *Fibring logics*. Oxford: Oxford University Press.
Gabbay, D.M. 1999. Compromise revision, a position paper. In *Dynamic worlds*, ed. B. Fronhofer and R. Pareschi, 111–148. Dordrecht: Kluwer.
Gabbay, D.M. 2001. Dynamics of practical reasoning: A position paper. In *Advances in modal logic 2*, ed. K. Segerberg, M. Zakhryaschev, M. de Rijke, and H. Wansing, 197—242. Stanford: CSLI Publications, CUP.
Gabbay, D.M. 2008a. Reactive Kripke models and contrary-to-duty obligations. In *DEON-2008, Deontic logic in computer science*, ed. Ron van der Meyden and Leendert van der Torre, 155–173. LNAI 5076. Springer: Heidelberg.
Gabbay, D.M. 2008b. Reactive Kripke semantics and arc accessibility. In *Pillars of computer science: essays dedicated to Boris (Boaz) Trakhtenbrot on the occasion of his 85th birthday*, ed. Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, 292–341. LNCS 4800. Berlin: Springer-Verlag. Earlier version published in *Proceeding of CombLog04* (http://www.cs.math.ist.utl.pt/comblog04/), ed. W. Carnielli, F.M. Dionesio, and P. Mateus, 7–20. Centre of Logic and Computation University of Lisbon, 2004, ftp://logica.cle.unicamp.br/pub/e-prints/comblog04/gabbay.pdf. Also published as a book report, by CLC (Centre for Logic and Computation), Instituto Superior Tecnico, Lisbon l, 2004, ISBN 972-99289-0-8.
Gabbay, D. 2009. Fibring argumentation frames. *Studia logica* 93(2–3):231–295.
Gabbay, D. 2010. Sampling logic and argumentation. *Journal of the Indian Council of Philosophical Research* 28(2):233–255. Special issue: Logic and Philosophy Today, ed. Amitabha Gupta and Johan van Benthem.
Gabbay, D.M. 2011. Dung's argumentation is equivalent to classical propositional logic with the Peirce-Quine dagger. *Logica universalis* 5(2):255–318. DOI: 10.1007/s11787-011-0036-3.
Gabbay, D.M. 2012. Equational approach to argumentation networks. *Argumentation and Computation* 3(2–3):87–142
Gabbay, D.M. 2011c. Introducing equational semantics for argumentation networks. In *Proceedings of the ECSQARU 2011*, ed. W. Liu. LNAI 6717, 19–35. Berlin/Heidelberg: Springer-Verlag.

Gabbay, D., and A. d'Avila Garcez. 2004. Fibring neural networks. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAA'04)*, San Jose, CA, 342–347. AAAI Press.
Gabbay, D.M., and A. Hunter. 1991. Making inconsistency respectable. Part 1: A logical framework for inconsistency in reasoning. In *Fundamentals of AI research*, LNCS 535. Berlin: Springer-Verlag.
Gabbay, D.M., and A. Hunter. 1993. Making inconsistency respectable. Part 2. Meta-level handling of inconsistency. In *LNCS* 747, Berlin: Springer-Verlag.
Gabbay, D., and N. Olivetti. 2000. *Goal directed proof theory*. Norwell: Kluwer.
Gabbay, D.M., and R.J.G.B. Queiroz. 1992. Extending the Curry-Howard interpretation to linear, relevance and other resource logics. *Journal of Symbolic Logic* 57:1319–1366.
Gabbay, D., and J. Williamson. 2004. Recursive causality in Bayesian networks and self fibring. In *Laws and models of science*, ed. D. Gillies, 173–247. College Publications: London.
Gabbay, D., and J. Woods. 2003. The laws of evidence and labelled deduction. In *Phi-news*, pp. 5–46, October 2003, http://phinews.ruc.dk/phinews4.pdf. Updated version in Gabbay, D.M., P. Canivez, S. Rahman, and A. Thiercelin, eds. *Approaches to legal rationality, logic, epistemology, and the unity of science* 20, 1st edn, 295–331. Springer: Heidelberg. DOI:10.1007/978-90-481-9588-6_15.
Gabbay, D.M., and J. Woods. 2003/2005. *A practical logic of cognitive systems*. A multi-volume series with Elsevier, including *Agenda relevance*, 508pp, 2003 and *The reach of abduction*, 2005.
Gabbay, D., O. Rodrigues, and A. Russo. 2010. *Revision acceptability and context*. Berlin: Springer.
Gärdenfors, P. 1988. *Knowledge in flux*. Cambridge, MA: The MIT Press. Second edition, College Publications, 2008.
Hamblin, C.L. 1970. *Fallacies*. London: Methuen.
Jeffrey, R. 1983. *The logic of decision*, 2nd edn. Chicago: University of Chicago Press.
Keane, A. 2000. *The modern law of evidence*. London: Butterworth.
Lambert, K., and W. Ulrich. 1980. *The nature of argument*. New York: Macmillan.
Makinson, D. 1994. General patterns in nonmonotonic reasoning. In *Handbook of logic in artificial intelligence and logic programming*, Vol. 3, ed. D.M. Gabbay, C.J. Hogger, and J.A. Robinson, 35–110. Oxford: Oxford University Press.
Makinson, D., and L. van der Torre. 2001. Constraints for input/output logics. *Journal of Philosophical Logic* 30:155–185.
Metcalfe, G., N. Olivetti, and D. Gabbay. 2008. *Proof theory for fuzzy logics*. New York: Springer.
Ng, R., and V. Subrahmanian. 1994. Dempster–Shafer logic programs and stable semantics. In *Logical methods*, ed. J.N. Crossley and J. Be Remmel, 654–704. Boston: Birkhauser.
Nute, D. 1994. Defeasible logic. *Handbook of logic in artificial intelligence and logic programming*, Vol. 3, ed. D.M. Gabbay, C.J. Hogger, J.A. Robinson, 353–398. Oxford: Oxford University Press.
de Queiroz, R., A. G. de Oliveira, and D. Gabbay. 2011. *The Functional Interpretation of Logical Deduction*. Singapore: World Scientific.
Reiter, R. 2002. *Knowledge in action*. Cambridge, MA: MIT Press.
RCCJ. 1993. Report of the Royal Commission of Criminal Justice, M2263, Ch 8, para 26. London: HMSO.
Scott, D. 1974. Completeness and axiomatizability in many valued logics. In *Proceedings of Tarski Symposium*, pp. 411–436. Providence: American Mathematical Society.
Tarski, A. 1936. On the concept of logical consequence (in Polish). Translation in *Logic semantics metamathematics*. Oxford: Oxford University Press, 1956.
Uglow, S. 1997. *Textbook on evidence*. London: Sweet and Maxwell.
Vermeir, D., E. Laenens. 1990. An overview of ordered logic in *Abstracts of the third logical biennial*, Varga, Bulgaria.
Vigano, L. 1999. *Labelled non-classical logics*. Kluwer. Dordrecht.
Walton, D. 1990. *Practical reasoning*. Savage: Rowman & Littlefield.
Woods, J. 1988. Are fallacies theoretical entities. *Informal Logic* 10:67–76.
Woods, J., and D. Walton. 1989. *Fallacies selected papers 1972–1982*, Dordrecht.

Index

D.M. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic: Volume 17*, Handbook of Philosophical Logic 17, DOI 10.1007/978-94-007-6600-6,