Logical calculi for reasoning with binding

DOI:
10.6100/IR630460

Document status and date:
Published: 01/01/2007

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Logical Calculi for Reasoning with Binding

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op donderdag 15 november 2007 om 16.00 uur

door

Adrianus Hubertus Johannes Mathijssen

geboren te Breda

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. J.F. Groote

Copromotor:
dr. M.J. Gabbay

“It’s never too late to reinvent the bicycle.”
System of a Down - Innervision

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics), partially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 642.000.504.

IPA dissertation series 2007-17.
© 2007 Aad Mathijssen.

Typeset using LaTeX.
Printed by Printservice Technische Universiteit Eindhoven.
Cover design by Tony Glebbeek.

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Mathijssen, Adrianus Hubertus Johannes

Logical calculi for reasoning with binding / door Adrianus Hubertus Johannes Mathijssen. – Eindhoven : Technische Universiteit Eindhoven, 2007.
Proefschrift. – ISBN 978-90-386-1138-9
NUR 993
Subject headings: formal logic / proof theory / algebra
CR Subject Classification (1998): F.4.1, D.3.1, F.3.2

Acknowledgements

Many people have helped me in making this thesis possible. They have assisted me directly or indirectly, they motivated me, or they just plainly distracted me. I take this opportunity to thank all of them here.

First of all, I would like to thank my supervisors Jan Friso Groote and Jamie Gabbay. Most importantly they have both helped me in learning how to do research. Besides that, Jan Friso allowed me to go my own way, for which I am grateful. I would like to thank Jamie for introducing me to a new field research, namely that of nominal techniques. All papers on which this thesis is based are joint work between him and me, so it is safe to say that this thesis was not possible without his great help. I also enjoyed the countless discussions we had on the phone from all over the world.

I would also like to thank the reading committee of this thesis: Jos Baeten, Jan Willem Klop and Andrew Pitts. They have provided valuable feedback which helped me to improve the original manuscript. I am honored to have them as well as Herman Geuvers and Bart Jacobs as members of the promotion committee.

I have benefited greatly from my visits to the UK which enabled me to directly discuss research with people working in the same area. I am grateful to Maribel Fernández, Jamie Gabbay and Andrew Pitts for organising these events.

The past four years I have worked in the Design and Analysis of Systems group headed by Jan Friso Groote. I thank all colleagues from this group and also from the Visualization group. In particular, I thank former and present officemates Arjan Mooij, Olga Tveretina, Muck van Weerdenburg and Frank Stappers for tolerating me in their office and for listening to my nonsense. I also thank Frank van Ham, Danny Holten, Bas Ploeger, Hannes Pretorius, Dennie Reniers, Frank Stappers and Muck van Weerdenburg for the great discussions we had during lunch.

Thanks go to my friends and family who all helped to drag me out of the research world and into the real world.

Special thanks go to my parents Jan and Helma, my sister Colette and her friend Ron for their love and support throughout the years.

But most importantly, I thank my girlfriend Sonja for her love and support. She has greatly helped me in letting me concentrate on my 'saaie' research when this was necessary, and in forcing me to explore other things in life when it was not.

Contents

Acknowledgements **i**

List of Figures **vii**

1 Introduction **1**

1.1 Background and Motivation 1
1.2 Contents of the Thesis 2
1.3 Suggested Method of Reading 3
1.4 Origin of the Chapters 4

2 Equational Logic with Binders and Meta-Variables **5**

2.1 Introduction 5
2.2 Syntax 6
2.2.1 Terms and Signatures 6
2.2.2 Judgement Forms, Axioms and Theories 8
2.3 A Derivation System 10
2.3.1 Permutation and Substitution Actions 10
2.3.2 Inference Rules 13
2.4 Proof-Theoretical Results 19
2.5 α-Equivalence with Meta-Variables 25
2.6 Conclusions 28
2.6.1 Related Work 28
2.6.2 Future Work 31

3 A Semantics **33**

3.1 Introduction 33
3.2 Nominal Sets 34
3.3 Interpretations, Models and Validity 36
3.4 Completeness 38
3.4.1 Free Term Models 38
3.4.2 Completeness for Equality Derivations 40
3.4.3 The Status of Freshness Derivations 44

3.5 Conclusions 45
3.5.1 Related Work 45

4 Capture-Avoiding Substitution 47
4.1 Introduction 47
4.2 A Sort System 48
4.3 Substitution on Ground Terms 51
4.4 Theory SIMP: Substitution on Closed Terms 58
4.5 Substitution on Open Terms using SIMP 62
4.5.1 Nominal Rewriting 62
4.5.2 SIMPr: Explicit Substitution Rewritten 65
4.5.3 Confluence, Conservativity and Consistency 66
4.5.4 Failure of Completeness for SIMP 67
4.5.5 Strong Normalisation and Decidability 68
4.6 Substitution on Open Terms using SUB 70
4.6.1 Soundness and the Relation to SIMP 71
4.6.2 Decidability 72
4.6.3 ω-Completeness 79
4.6.4 Restricting the Sort System 81
4.7 Conclusions 82
4.7.1 Related Work 82
4.7.2 Future Work 84

5 One-and-a-halfth-Order Logic 85
5.1 Introduction 85
5.2 Sorts, Terms and Substitution 87
5.3 A Sequent Calculus 90
5.4 Proof-Theoretical Results 95
5.4.1 Equivariance and Substitution 96
5.4.2 Cut-Elimination 99
5.5 Relation to First-Order Logic with Equality 103
5.6 Conclusions 105
5.6.1 Related Work 105
5.6.2 Future Work 106

6 An Axiomatisation of One-and-a-halfth-Order Logic 109
6.1 Introduction 109
6.2 Sequent Derivability Implies FOL Derivability 110
6.3 FOL Derivability Implies Sequent Derivability 114
6.4 Equivalence of the Sequent Calculus and FOL 117
6.5 Conclusions 118

7 Conclusions 121

A ZFA Equivariance **123**
A.1 Introduction 123
A.2 ZFA Set Theory and Equivariance 124

Bibliography **127**

Summary **133**

Samenvatting **135**

Curriculum Vitae **137**

Index **139**

List of Figures

2.1 Derivation rules for freshness 13
2.2 Derivation rules for equality 14
2.3 An example derivation in nominal algebra 16
2.4 Syntax-directed rules for CORE 25

4.1 Axioms of theory SUB 51
4.2 α-equivalence on ground terms 56
4.3 Axioms of theory SIMP 58
4.4 Derivation rules for nominal rewriting 63
4.5 Derivation rules for the transitive reflexive closure of R 63
4.6 Substitution as a rewrite system SIMPr 65

5.1 Axioms of theory SUB for one-and-a-halfth-order logic 89
5.2 Sequent calculus for one-and-a-halfth-order logic 92
5.3 Example derivations in one-and-a-halfth-order logic 93
5.4 Another example derivation in one-and-a-halfth-order logic 95
5.5 Admissible sequent rules for one-and-a-halfth-order logic 101
5.6 Gentzen's sequent calculus for first-order logic 103

6.1 Axioms of theory FOL 110

A.1 Axioms of ZFA set theory 125

Chapter 1

Introduction

1.1 Background and Motivation

Logic is the study of deductive reasoning. In the field of *formal* logic we study deductive reasoning using the notions of *validity* and *derivability* of *formulae* expressed in a *formal language*. Let us consider derivability.

To construct a notion of derivability we choose a collection of *inference rules* that tells us how to conclude a formula from a number of *premises*. These premises are also formulae. If an inference rule has no premises, we call it an *axiom*. A *theorem* is a formula that can be derived from the axioms using the inference rules, while a *proof* of a theorem shows how we can derive such a theorem using the inference rules.

Probably the simplest formal logics are *propositional logics*, in which formulae are made up from atomic propositions and logical connectives. Things get a bit more complicated with formal logics known as *predicate logics*. Here formulae are extended with *terms* representing a domain of discourse. These terms may contain *(object-)variables* which can be quantified, i.e. variables can be bound by quantifiers; we call a variable free it is not bound by any quantifier. The notion of inference rules is non-trivial because we usually need to put *freshness* side-conditions on some rules to ensure that free variables do not become bound. Also we need to be able to rename variables in formulae, also known as α-conversion, and we need to be able to substitute terms for variables in a capture-avoiding way.

Propositional and predicate logics are heavily used in mathematics and computer science. When using these logics, we often write theorems and proofs involving *meta-variables* that range over formulae or terms. But these *schemas* of theorems and proofs are not part of the formal system themselves.

Logic teaches us that reasoning can and should be formalised, not only its conclusions. So if we use meta-variables in reasoning, we can and should ask 'what is the mathematics of this reasoning'?

A straightforward way to formalise these meta-level properties is to enrich formal logics with meta-variables. For propositional logics this is not hard, but in the setting of predicate logics we run into a number of problems:

- When is an object-variable fresh for a meta-variable? We only know this when the meta-variable is instantiated to a concrete formula or term (not mentioning meta-variables).
- What is suitable representation of α-conversion of object-variables in the setting with meta-variables?
- In the presence of meta-variables, substitution of terms for object-variables becomes non-trivial: what does it mean when we try to substitute a term for a variable in a meta-variable?

A number of different solutions have been proposed for these problems. The state of the art solution is the higher-order approach which uses a hierarchy of types to model the difference between object- and meta-level variables [vB01, Mei92]. An alternative solution is to reject object-variables altogether, as is done in approaches using combinators [CF58, Bar84] and cylindric algebras [HMT85, LS04].

Unfortunately, none of these solutions allows for a *natural* formalisation. By this we mean a formalisation of binding and meta-variables that is close to informal practise. A consequence of this is that the formalisation of schemas of theorems and proofs are not always simple refinements but sometimes requires a fair amount of emulation. In De Bruijn's words [dB91]:

> "I think that in formalizing mathematics, and in particular in preparing mathematics for justification, it is usually elegant as well as efficient to do everything in the *natural* way."

There have been solutions to the problem of binding and meta-variables that embrace the difference between object- and meta-variables. One of these solutions uses so-called nominal techniques [GP02], which has two levels of variables, and a built-in notion of freshness of variables with respect to meta-variables.

Nominal techniques have been successfully applied to unification [UPG04], term rewriting [FG07] and first-order logic [Pit03]. However, the original application of these techniques was purely to represent formal *syntax* with meta-variables.

1.2 Contents of the Thesis

In this thesis we explore a novel but very natural application of nominal techniques to represent formal *logics* with meta-variables. In this way, and consistent with de Bruijn's philosophy, we allow particularly natural and yet fully formal representations of the kind of reasoning which is pervasively used in informal mathematical practice.

The formal logics we explore are logics of equality and first-order predicate logic [Gen35, Pra65]. We extend these logics with meta-variables and freshness side-conditions such that object-variables, binders, meta-variables and freshness have representations that are very close to informal practice.

Most of this thesis is devoted to the logic of equality with meta-variables. We give an overview of the topics per chapter:

- Chapter 2 introduces the formal language of the logic of equality with meta-variables, and provides a notion of *derivation* with the ability to impose axioms.
- Chapter 3 provides a *semantics* for the logic of equality based on so-called nominal sets, which is the natural model for names and binding in our setting. We will show that our notion of derivation is complete with respect to this semantics.
- Chapter 4 studies a set of axioms for *capture-avoiding substitution* with meta-variables. We show that this axiomatisation is very powerful: as well as being decidable, it is complete with respect to the usual notion of capture-avoiding substitution (without meta-variables).
- Chapter 5 introduces a sequent calculus for first-order logic with equality extended with meta-variables. We show that derivations in this so-called *one-and-a-halfth-order logic* precisely correspond to schemas of derivations in first-order logic, and we show that it satisfies proof-theoretical properties such as cut-elimination and consistency.
- Chapter 6 studies a set of *axioms for one-and-a-halfth-order logic*. We show that derivability in the logic of equality using this set of axioms is equivalent to derivability in the sequent calculus of one-and-a-halfth-order logic.
- Chapter 7 draws conclusions.

1.3 Suggested Method of Reading

We have written this thesis with the intention to be read from cover to cover. However, since one cannot expect every reader to be able to do this, we give some hints on how to read the parts he or she might be interested in.

Chapter 2 defines the logic of equality with meta-variables. It is fundamental to all remaining chapters of this thesis, even for Chapter 5 which introduces a sequent calculus for first-order logic with meta-variables.

After reading Chapter 2, the dependencies of the remaining technical chapters are as follows:

- Chapter 3 can be read independently from the other chapters.
- Each one of Chapters 4, 5 and 6 depends on the previous chapter.

For easy reference we provide the dependencies of the technical chapters in Table 1.1.

Chapter	**Needs**	**Needed by**
2. Equational Logic with Binders and Meta-Variables	-	3, 4, 5, 6
3. A Semantics	2	-
4. Capture-Avoiding Substitution	2	5
5. One-and-a-halfth-Order Logic	2, 4	6
6. An Axiomatisation of One-and-a-halfth-Order Logic	2, 5	-

Table 1.1 Dependencies of the chapters in this thesis

1.4 Origin of the Chapters

This thesis is based on the following publications:

[1] Murdoch J. Gabbay and Aad Mathijssen. A Formal Calculus for Informal Equality with Binding. In *WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation*, volume 4576 of *LNCS*, pages 162–176, Springer, 2007.

[2] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. In *ICTAC'2006: 3rd International Colloquium on Theoretical Aspects of Computing*, volume 4281 of *LNCS*, pages 198–212, Springer, 2006.

[3] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. To be published in *Formal Aspects of Computing.*

[4] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-Order Logic. In *PPDP'06: 8th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming*, pages 189–200. ACM Press, 2006.

[5] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-Order Logic. To be published in *Journal of Logic and Computation.*

Chapter 2 has its origin in all of these publications. Chapter 3 is based on the material in [1], but has been significantly expanded. Chapter 4 is based on [3], which is a completely reworked and improved versions of [2]. Chapters 5 and 6 are based on [5], which itself is a expanded version of [4].

Chapter 2

Equational Logic with Binders and Meta-Variables

2.1 Introduction

Perhaps *equality* is the simplest possible judgement form. Informal specification of logic and computation often involves equalities with *binding* and subject to conditions about *freshness*. For example:

λ-calculus:	$\lambda x.(tx) = t$	if $x \notin \mathit{fv}(t)$
First-order logic:	$\forall x.(\phi \supset \psi) = \phi \supset \forall x.\psi$	if $x \notin \mathit{fv}(\phi)$
π-calculus:	$\nu x.(P \mid Q) = P \mid \nu x.Q$	if $x \notin \mathit{fv}(P)$
Process algebra with data:	$\sum x.p = p$	if $x \notin \mathit{fv}(p)$

And for any binder $\zeta \in \{\lambda, \forall, \nu, \sum\}$:

Substitution:	$(\zeta y.u)[x \mapsto t] = \zeta y.(u[x \mapsto t])$	if $y \notin \mathit{fv}(t)$

Here $\mathit{fv}(t)$ denotes the free variables of t. It is not hard to extend this short list with many more examples.

In the equalities above there are *two* levels of variable:

- x and y are variables of the system being axiomatised, we call these *object-level* variables.
- t, u, ϕ, ψ, P, Q and p range over terms of that system's syntax, we call them *meta-level* variables.

Unfortunately these equalities are subject to freshness side-conditions $x \notin \mathit{fv}(t)$ which make them something other than 'just equalities'.

Ways have been developed to attain the simplicity and power of the theory of equality between terms. For example we can work with combinators [CF58], cylindric algebra [HMT85], higher-order algebra [Mei92] or higher-order logic [vB01].

Roughly speaking: combinatory approaches reject object-level variables entirely; cylindric approaches reject them as independent syntactic entities but enrich the language of term-formers to regain some lost expressivity; higher-order approaches model the difference between the two levels of variable using a hierarchy of types. These approaches do not permit a *direct* representation of the two-level structure which informal syntax displays in terms such as $\lambda x.t$ or $\forall x.\phi$.

In this chapter we introduce **Nominal Algebra**. This is a logic based on equality which *embraces* the two-level variable structure by representing it directly in its syntax. Informal equivalences can be represented as axioms almost symbol-for-symbol. For example the equalities from the beginning of this section are represented by:

λ-calculus:	$a\#X \vdash$	$\lambda[a](Xa) = X$
First-order logic:	$a\#X \vdash$	$\forall[a](X \supset Y) = X \supset \forall[a]Y$
π-calculus:	$a\#X \vdash$	$\nu[a](X \mid Y) = X \mid \nu[a]Y$
Proces algebra with data:	$a\#X \vdash$	$\sum[a]X = X$
Substitution:	$b\#X \vdash$	$(\zeta[b]Y)[a \mapsto X] = \zeta[b](Y[a \mapsto X])$

Here a and b are distinct *atoms* representing object-level variables; X and Y are *unknowns* representing meta-level variables. Each equality is equipped with a *freshness condition* of the form $a\#X$ that guarantees that X can only be instantiated to a term for which a is fresh. The rest of this chapter makes this formal.

Overview We introduce the syntax of our calculus in Section 2.2. In Section 2.3 we define the calculus itself, and provide some examples. In Section 2.4 we show a number of proof-theoretical results. In Section 2.5 we show that the calculus without any axioms corresponds to an existing notion of α-equality with meta-variables. We discuss related and future work in Section 2.6.

2.2 Syntax

We need a syntax in which expressions with meta-variables, such as $\lambda x.t$ and $x \notin fv(t)$, may be represented. We use nominal terms [UPG04] because they offer built-in support for meta-variables, abstraction, and freshness in a way that is close to informal practice.

2.2.1 Terms and Signatures

Definition 2.2.1. Fix a countably infinite collection of **atoms** $a, b, c, \ldots$ representing object-level variables. Fix a countably infinite collection of **unknowns** $X, Y, Z, \ldots$ representing meta-level variables. Fix **term-formers** f to each of which is associated some unique **arity** n which is a nonnegative number; write $\mathsf{f} : n$ to indicate that f has arity n. Assume these collections are disjoint.

Convention 2.2.2. We shall use a *permutative convention* that $a, b, c, \ldots$ range permutatively over atoms, so that for example a and b are always distinct.

In the rare circumstances where we do not want this behaviour, we choose another symbol such as a'.

For the purpose of α-conversion, we need to be able to rename atoms. We use *permutations* of atoms instead of the more common substitutions of atoms for atoms, because permutations have better mathematical properties; most notably, permutations are capture-avoiding by definition (see the Introduction of [GP02] and [Pit03] for a detailed exposition).

Definition 2.2.3. Let $\mathbb{A} = \{a, b, c, \ldots\}$. A **permutation** π of atoms is a total bijection $\mathbb{A} \to \mathbb{A}$ with **finite support**, meaning that for some finite set of atoms (which may be empty) $\pi(a) \neq a$, *but* for all atoms not in that set, $\pi(a) = a$.

Finite support is a mathematical notion of 'most': π is a bijection on atoms such that $\pi(a) = a$ for *most a*.

We introduce some notation for permutations that we will need later on.

Definition 2.2.4. Write ι for the **identity permutation** such that $\iota(a) = a$ always. Write $\pi \circ \pi'$ for **functional composition** and write $\pi^{\text{-}1}$ for **inverse**. This makes permutations into a group — write $\mathbb{P}$ for the set of all permutations. Write $(a\ b)$ for the permutation that **swaps** a and b, i.e. the permutation that maps a to b, b to a and all other c to themselves. Finally, write $a \in \pi$ when $\pi(a) \neq a$.

Using the above ingredients we can form terms.

Definition 2.2.5. **Terms** t, u, v are inductively defined by:

$$t \ ::= \ a \ \mid \ \pi \cdot X \ \mid \ [a]t \ \mid \ \mathsf{f}(t_1, \ldots, t_n)$$

We call $[a]t$ an **abstractor**; it uniformly represents the '$x.t$' or $x.\phi$' part of expressions such as '$\lambda x.t$' or '$\forall x.\phi$'. We call $\pi \cdot X$ a **moderated unknown**; it represents an unknown term on which a permutation of atoms is performed when it is instantiated. We write $\iota \cdot X$ just as X, for brevity.

In Section 2.3 we will see that in $\pi \cdot X$ the unknown X will get substituted for a term and then π will permute the atoms in that term. This notion is grounded in semantics [GP02] and permits a succinct treatment of α-renaming atoms (see Section 2.5 and [UPG04]).

Definition 2.2.6. A **signature** Σ is a set of term-formers with their arities.

Example 2.2.7. Here are some example signatures:

- $\{\mathsf{lam} : 1, \mathsf{app} : 2\}$ is a signature for the λ-calculus.

  We show how the terms in this signature relate to 'ordinary' syntax. For convenience identify atoms with *variable symbols*, then the syntax of the untyped λ-calculus is inductively defined by:

$$e \ ::= \ a \ \mid \ \lambda a.e \ \mid \ ee$$

The map -$'$ from untyped λ-term to nominal terms is inductively defined by:

$$a' = a \qquad (\lambda a.e)' = \mathsf{lam}([a](e')) \qquad (e_1 e_2)' = \mathsf{app}(e_1', e_2')$$

We generally sugar $\mathsf{lam}([a]t)$ to $\lambda[a]t$ and $\mathsf{app}(t, u)$ to tu.

- $\{\bot : 0, \supset: 2, \forall : 1, \approx: 2\}$ is a signature for first-order logic with equality (the symbol for equality inside the logic is $\approx$).

  We sugar $\bot()$ to $\bot$, $\supset(\phi, \psi)$ to $\phi \supset \psi$, $\forall([a]\phi)$ to $\forall[a]\phi$ and $\approx(t, u)$ to $t \approx u$.

When we define axioms for the λ-calculus and first-order logic, we shall extend these signatures with a term-former for representing capture-avoiding substitution.

Remark 2.2.8. Consistent with previous work on nominal rewriting [FG07] we do not impose an a priori sort system on terms. We prefer to leave that to later (e.g. Chapters 4 and 5) when we consider specific applications of nominal algebra.
Although this allows us to write 'silly terms' like $\lambda(tu)$ and $\forall(t \approx u)$, it simplifies the presentation, and our results trivially specialise to the more specific cases.

Definition 2.2.9. Write $t \equiv u$ for **syntactic identity** of terms.

Note that if $\pi = \pi'$ then $\pi \cdot X \equiv \pi' \cdot X$, since permutations are represented by themselves. There is no quotient by abstraction so for example $[a]a \not\equiv [b]b$.
Syntactic identity $t \equiv u$ emphasises the difference from provable equality $t = u$, which is a logical assertion defined in Subsection 2.2.2, and object-level equality $t \approx u$, which is a term.

Definition 2.2.10. Say that a term t is **closed** when it does not contain any *unknowns*.

A closed term may still mention atoms, e.g. the terms a and $[a]b$ are closed and the terms X and $[a]X$ are not.

Definition 2.2.11. Write $a \in t$ for 'a **occurs in (the syntax of)** t', and $X \in t$ for 'X **occurs in (the syntax of)** t'. Similarly write $a \notin t$ and $X \notin t$ for 'does not occur in the syntax of t'.

Occurrence $a \in t$ is literal, e.g. $a \in [a]a$ and $a \in \pi \cdot X$ when $a \in \pi$.

2.2.2 Judgement Forms, Axioms and Theories

Definition 2.2.12. A **freshness** is a pair $a\#t$ of an atom a and a term t. Call a freshness $a\#X$ (so $t \equiv X$) **primitive**. Write Δ and ∇ for finite sets of *primitive* freshnesses and call them **freshness contexts**.

Intuitively we should read $a\#t$ as meaning '$a \notin fv(t)$' or in words 'a is fresh for t'. A reason this notion is quite subtle in nominal techniques is the unknowns X; $a\#X$ is *not* necessarily true even though $a \notin X$ *is* a fact of the syntax. An unknown X represents an unknown term in the syntax; $a\#X$ has the quality of a *promise* or *assertion* about what term that can be, or put another way, about what we can instantiate X to.

Definition 2.2.13. We may drop set brackets in sets of freshnesses, e.g. writing $a\#t, b\#u$ for $\{a\#t, b\#u\}$. Also, we may write $a\#t, u$ for $a\#t, a\#u$. Furthermore, for any set of freshnesses S write $a \in S$ when a occurs anywhere in S, and $X \in S$ when X occurs anywhere in S.

Definition 2.2.14. An **equality** is a pair $t = u$ where t and u are terms.

Equalities will be used to state that two terms are *provably equal*.

Definition 2.2.15. Nominal algebra has two *judgement forms*:

- A **freshness judgement form** $\Delta \vdash a\#t$ is a pair of a freshness context Δ and a freshness $a\#t$.
- An **equality judgement form** $\Delta \vdash t = u$ is a pair of a freshness context Δ and an equality $t = u$.

We may write $\emptyset \vdash a\#t$ as $\vdash a\#t$, and $\emptyset \vdash t = u$ as $\vdash t = u$.

Definition 2.2.16. A **theory** $\mathsf{T} = (\Sigma, Ax)$ is a pair of a signature Σ and a possibly infinite set of *equality* judgement forms Ax in that signature; we call them the **axioms**.

We do not allow freshness judgements as axioms; we shall see that they can be expressed using equalities instead (see Subsection 3.4.3).

Example 2.2.17. Here are some nominal algebra theories:

- LAM has signature $\{\mathsf{lam} : 1, \mathsf{app} : 2, \mathsf{sub} : 2\}$ and two axioms

$$\begin{array}{lrl} (\beta) & & \vdash\ (\lambda[a]Y)X = Y[a \mapsto X] \\ (\eta) & a\#X & \vdash\ \lambda[a](Xa) = X \end{array}$$

  where we sugar $\mathsf{sub}([a]t, u)$ to $t[a \mapsto u]$.

- FOL has signature $\{\bot : 0, \supset : 2, \forall : 1, \approx : 2, \mathsf{sub} : 2\}$ and seven axioms

$$\begin{array}{lrll} (\mathbf{MP}) & & \vdash\ \top \supset X & = X \\ (\mathbf{Mer}) & & \vdash\ ((((X \supset Y) \supset (\neg Z \supset \neg W)) \supset Z) \supset V) & \\ & & \qquad \supset ((V \supset X) \supset (W \supset X)) & = \top \\ (\mathbf{Qinst}) & & \vdash\ \forall[a]X \supset X[a \mapsto Y] & = \top \\ (\mathbf{Qdist}) & & \vdash\ \forall[a](X \wedge Y) \Leftrightarrow \forall[a]X \wedge \forall[a]Y & = \top \\ (\mathbf{Qextr}) & a\#X & \vdash\ \forall[a](X \supset Y) \Leftrightarrow X \supset \forall[a]Y & = \top \\ (\mathbf{Esubst}) & & \vdash\ Z \approx Y \wedge X[a \mapsto Y] \supset X[a \mapsto Z] & = \top \\ (\mathbf{Erefl}) & & \vdash\ X \approx X & = \top \end{array}$$

Here we use standard classical logic sugar for $\top$, $\neg$, $\wedge$ and $\Leftrightarrow$.

Axioms (**MP**) and (**Mer**) characterise propositional logic; axioms (**Qinst**), (**Qdist**) and (**Qextr**) characterise quantification; and axioms (**Esubst**) and (**Erefl**) characterise object-level equality.

Chapter 6 treats this axiomatisation in detail.

- SUB gives substitution term-former sub the correct behaviour in theories LAM and FOL. It is a family of theories, one for each signature Σ that includes sub, with axioms

$$\begin{array}{rrll}
(\mathbf{var}{\mapsto}) & & \vdash & a[a\mapsto X] = X \\
(\#{\mapsto}) & a\#Y & \vdash & Y[a\mapsto X] = Y \\
(\mathsf{f}{\mapsto}) & & \vdash & \mathsf{f}(Y_1,\dots,Y_n)[a\mapsto X] = \mathsf{f}(Y_1[a\mapsto X],\dots,Y_n[a\mapsto X]) \\
(\mathbf{abs}{\mapsto}) & b\#X & \vdash & ([b]Y)[a\mapsto X] = [b](Y[a\mapsto X]) \\
(\mathbf{ren}{\mapsto}) & b\#Y & \vdash & Y[a\mapsto b] = (b\ a)\cdot Y \\
(\eta{\mapsto}) & b\#X & \vdash & [a]\mathsf{sub}(X,a) = X
\end{array}$$

  For each term-former f (including sub), there is one axiom ($\mathsf{f}{\mapsto}$). Note the heavy use of freshness side-conditions to manage the relationship between atoms and unknowns.

  This axiomatisation is the topic of Chapter 4.

- CORE is a family of theories with no axioms; there is one such theory for each signature Σ. It has built-in α-equivalence, so for example $\lambda[a]a$ is equal to $\lambda[b]b$.[1]

  Theory CORE is discussed in Section 2.5.

Similar developments for other systems with binding, such as the process algebra with data [Gro97, Lut02, GMR$^+$07] and the π-calculus [Par01] from the Introduction of this chapter should also be possible.

2.3 A Derivation System

In this section we define notions of derivation which represent freshness assumptions on meta-variables (Figure 2.1), and permit axioms involving abstraction that are conditioned by freshness assumptions (Figure 2.2), just like we do in informal practice.

2.3.1 Permutation and Substitution Actions

Before we introduce our calculus, we elaborate on two important prequisites for the instantiation of axioms; we need to be able to *permute atoms in terms*, and *substitute terms for unknowns* in a capturing way.

[1] α-equivalence is expressed as a derivation rule: the (**perm**) rule from Figure 2.2. The (**perm**) rule is discussed in detail in Subsection 2.3.2.

Definition 2.3.1. The **(object-level) permutation action** $\pi \cdot t$ on terms is inductively defined by:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \qquad \pi \cdot [a]t \equiv [\pi(a)](\pi \cdot t)$$
$$\pi \cdot \mathsf{f}(t_1, \ldots, t_n) \equiv \mathsf{f}(\pi \cdot t_1, \ldots, \pi \cdot t_n)$$

Intuitively, π propagates through the structure of t until it reaches an atom or a moderated unknown. The reader should read the above definition as *syntactic sugar*. For instance, this means that $\pi \cdot [a](\pi' \cdot X)$ is *not* an actual term but sugar for $[\pi(a)]((\pi \circ \pi') \cdot X)$.

Composition and identity of permutations naturally extend to terms as shown in the following lemma.

Lemma 2.3.2. $(\pi \circ \pi') \cdot t \equiv \pi \cdot (\pi' \cdot t)$ *and* $\iota \cdot t \equiv t$.

Proof. By induction on the structure of t, using Definition 2.3.1. □

Substitution is the mechanism by which unknowns become terms, and is necessary in algebra in order to define instances of axioms.

Definition 2.3.3. A **(meta-level) substitution** σ is a function from unknowns to terms with **finite support** meaning that for some finite set of unknowns $\sigma(X) \not\equiv X$, and for all other unknowns $\sigma(X) \equiv X$.

We need some notation.

Definition 2.3.4. We write $[t_1/X_1, \ldots, t_n/X_n]$ for the substitution σ such that $\sigma(X_i) \equiv t_i$ for $1 \leq i \leq n$, and $\sigma(Y) \equiv Y$ for $Y \not\equiv X_i$.

Write $[]$ for the **empty substitution** such that $t[] \equiv t$. Write $\sigma \circ \sigma'$ for **composition of substitutions**, i.e. $t(\sigma \circ \sigma') \equiv (t\sigma)\sigma'$.

Write $a \in \sigma$ if there exists an X such that $a \in \sigma(X)$, and similarly write $a \notin \sigma$ if there is no such X. For example $a \in [a/X]$ and $a \notin []$.

Definition 2.3.5. The **(meta-level) substitution action** $t\sigma$ on terms is inductively defined by:

$$a\sigma \equiv a \qquad (\pi \cdot X)\sigma \equiv \pi \cdot \sigma(X) \qquad ([a]t)\sigma \equiv [a](t\sigma)$$
$$\mathsf{f}(t_1, \ldots, t_n)\sigma \equiv \mathsf{f}(t_1\sigma, \ldots, t_n\sigma)$$

Intuitively, σ propagates through the structure of t until it reaches an atom or a moderated unknown. σ 'evaporates' on an atom, and acts on the unknown X of a moderated unknown $\pi \cdot X$. The moderating permutation π then passes into the term substituted in that position. We suggest a reading of $\pi \cdot X$ as 'permute π in whatever X eventually becomes'.

Note that meta-level substitution does not avoid capture; $([a]X)[a/X] \equiv [a]a$. In this sense X is 'meta' and really does represent an unknown *term*. There is an exact and deliberate analogy here with context substitution, which is the substitution used when we write 'let - be a in $\lambda a.$-', to obtain $\lambda a.a$.

Definition 2.3.6. Give substitution and permutation actions higher precedence than abstraction and any of the sugared term-formers, and put substitution before permutation.

Note how substitution interacts with permutation in the case of an unknown, for example $((a\ b)\cdot X)[b/X] \equiv (a\ b)\cdot b \equiv a$. So π in X is 'waiting for a substitution to arrive', as also made formal in the following property:

Lemma 2.3.7. *$\pi \cdot t\sigma \equiv (\pi \cdot t)\sigma$.*

Proof. By induction on the structure of t, using Definitions 2.3.1 and 2.3.5. The case of $t \equiv \pi' \cdot X$ uses Lemma 2.3.2. □

Another permutation action is useful.

Definition 2.3.8. The **meta-level permutation action** t^π on terms t is inductively defined by:

$$a^\pi \equiv \pi(a) \qquad (\pi' \cdot X)^\pi \equiv \pi \circ \pi' \circ \pi^{-1} \cdot X \qquad ([a]t)^\pi \equiv [\pi(a)]t^\pi$$
$$\mathsf{f}(t_1, \ldots, t_n)^\pi \equiv \mathsf{f}({t_1}^\pi, \ldots, {t_n}^\pi)$$

Also for this permutation action, composition and identity of permutations extend to terms.

Lemma 2.3.9. *$t^{\pi\circ\pi'} \equiv {t^{\pi'}}^{\pi}$ and $t^\iota \equiv t$.*

Proof. By induction on the structure of t, using Definition 2.3.8. □

In the presence of substitution, the two permutation actions $\pi \cdot t$ and t^π are interdefinable; however, sometimes one is more natural than the other, we shall point out how later (Remark 2.3.15).

Lemma 2.3.10. *Given a term t, let σ be a substitution that maps each $X \in t$ to $\pi \cdot X$, and let σ' be a substitution that maps each $X \in t$ to $\pi^{-1} \cdot X$.*
Then $\pi \cdot t \equiv t^\pi\sigma$ and $t^\pi \equiv (\pi \cdot t)\sigma'$.

Proof. By induction on the structure of t, using Definitions 2.3.1, 2.3.8 and 2.3.5 of $\pi \cdot t$, t^π and $t\sigma$. The only interesting case is when $t \equiv \pi' \cdot X$. Then we need to show $\pi \cdot (\pi' \cdot X) \equiv (\pi' \cdot X)^\pi\sigma$. Using Definitions 2.3.8 and 2.3.5 we obtain $(\pi' \cdot X)^\pi\sigma \equiv (\pi \circ \pi' \circ \pi^{-1}) \cdot (\pi \cdot X)$ for the right-hand-side. By Definition 2.3.1 this is equivalent to $(\pi \circ \pi' \circ \pi^{-1} \circ \pi) \cdot X$, which is equivalent to $(\pi \circ \pi') \cdot X$ by basic permutation group theory. Again by Definition 2.3.1 this is equivalent to $\pi \cdot (\pi' \cdot X)$, which we needed to show. The proof of $(\pi' \cdot X)^\pi \equiv (\pi \cdot (\pi' \cdot X))\sigma'$ follows similar lines. □

Definition 2.3.11. We extend notation for t^π, $\pi \cdot t$ and $t\sigma$ to freshness contexts Δ as follows:

$$\begin{array}{rcl} \Delta^\pi & \text{is} & \{\pi(a)\#X \mid a\#X \in \Delta\} \\ \pi \cdot \Delta & \text{is} & \{\pi(a)\#\pi \cdot X \mid a\#X \in \Delta\} \\ \Delta\sigma & \text{is} & \{a\#\sigma(X) \mid a\#X \in \Delta\} \end{array}$$

Note that Δ^π is a freshness context, but $\pi \cdot \Delta$ and $\Delta\sigma$ need not be.

$$\frac{}{a\#b}\,(\#\mathbf{ab}) \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi\cdot X}\,(\#\mathbf{X})\quad(\pi\neq\iota)$$

$$\frac{}{a\#[a]t}\,(\#[]\mathbf{a}) \qquad \frac{a\#t}{a\#[b]t}\,(\#[]\mathbf{b}) \qquad \frac{a\#t_1\ \cdots\ a\#t_n}{a\#\mathsf{f}(t_1,\ldots,t_n)}\,(\#\mathsf{f})$$

Figure 2.1 Derivation rules for freshness

2.3.2 Inference Rules

Definition 2.3.12. Define **derivability on freshnesses** (in some signature Σ) in natural deduction style by the rules in Fig. 2.1. Here we have used the following conventions:

- a and b range over *distinct* atoms (see Convention 2.2.2);
- π ranges over permutations;
- X ranges over unknowns;
- t and $t_1,\ldots,t_n$ range over terms;
- f ranges over term-formers; there is one copy of the rule for each term-former.

We use similar conventions henceforth.

Write $\Delta\vdash a\#t$ when a derivation of $a\#t$ exists according to the rules in Fig. 2.1 using the elements of Δ as assumptions; say that $a\#t$ **is derivable from** Δ. Write $\Delta\nvdash a\#t$ when $a\#t$ is not derivable from Δ.

Finally, we write $\Delta\vdash S$ for a set of freshnesses S when $\Delta\vdash a\#t$ for each $a\#t\in S$.

Note that the $(\#\mathbf{X})$ rule excludes the identity permutation ι. While there is no mathematical reason for this, there is a nice *computational* one: the algorithm obtained by reading the rules bottom-up, must terminate.

Example 2.3.13. In the signature of theory LAM (Example 2.2.17) we know $\vdash a\#\lambda[b]b$ and $a\#X\vdash a\#X(\lambda[a]Y)$:

$$\dfrac{\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#[b]b}\,(\#[]\mathbf{b})}{a\#\lambda[b]b}\,(\#\mathsf{f}) \qquad \dfrac{a\#X \quad \dfrac{\dfrac{}{a\#[a]Y}\,(\#[]\mathbf{a})}{a\#\lambda[a]Y}\,(\#\mathsf{f})}{a\#X(\lambda[a]Y)}\,(\#\mathsf{f})$$

The following are *non*-derivable freshnesses in this signature:

$$\nvdash a\#a \qquad \nvdash a\#X(\lambda[a]Y) \qquad \nvdash a\#(\lambda[a]b)a$$

$$\frac{}{t = t}\,(\mathbf{refl}) \qquad \frac{t = u}{u = t}\,(\mathbf{symm}) \qquad \frac{t = u \quad u = v}{t = v}\,(\mathbf{tran})$$

$$\frac{t = u}{[a]t = [a]u}\,(\mathbf{cong}[]) \qquad \frac{t = u}{\mathsf{f}(t_1, \ldots, t, \ldots, t_n) = \mathsf{f}(t_1, \ldots, u, \ldots, t_n)}\,(\mathbf{congf})$$

$$\frac{\nabla^\pi \sigma}{t^\pi \sigma = u^\pi \sigma}\,(\mathbf{ax}_{\nabla \vdash \mathbf{t}=\mathbf{u}}) \qquad \frac{a\#t \quad b\#t}{(a\ b) \cdot t = t}\,(\mathbf{perm})$$

$$\frac{\begin{array}{c}[a\#X_1, \ldots, a\#X_n] \quad \Delta \\ \vdots \\ t = u\end{array}}{t = u}\,(\mathbf{fr}) \quad (n \geq 1,\ a \not\in t, u, \Delta)$$

Figure 2.2 Derivation rules for equality

In the signature of theory FOL (Example 2.2.17), derivable freshnesses are:

$$\vdash a\#\forall[a]P \qquad a\#T \vdash a\#(a \approx a)[a \mapsto T] \qquad a\#X \vdash b\#(b\ a) \cdot X.$$

Non-derivable freshnesses in this signature are:

$$\nvdash a\#\forall[b]P \qquad \nvdash a\#(a \approx a)[a \mapsto T] \qquad a\#X \nvdash a\#(b\ a) \cdot X.$$

In order to define derivability on equalities recall notation $\pi \cdot t$, $t\sigma$ and t^π for actions on terms t from Definitions 2.3.1, 2.3.5 and 2.3.8. Also recall notation $\Delta\sigma$ and Δ^π for the extensions to freshness contexts Δ from Definition 2.3.11. Finally, recall from Definition 2.2.12 that as well as Δ we write ∇ for freshness contexts.

Definition 2.3.14. Define **derivability on equalities** (between terms in the signature of T) by the rules in Fig. 2.2. Write $\Delta \vdash_{\mathsf{T}} t = u$ when a derivation of $t = u$ exists using the rules in Figure 2.2 (and the ones in Figure 2.1), such that:

- for each instance of $(\mathbf{ax}_{\nabla \vdash \mathbf{t}=\mathbf{u}})$, $\nabla \vdash t = u$ is an axiom of T;
- in the derivations of freshnesses (introduced by instances of $(\mathbf{ax}_{\nabla \vdash \mathbf{t}=\mathbf{u}})$ and $(\mathbf{perm})$) the assumptions used are from Δ only.

Say that $t = u$ **is derivable from Δ in T**.
Write $\Delta \nvdash_{\mathsf{T}} t = u$ when there does not exist such a derivation.

The $(\mathbf{fr})$ rule allows us to introduce freshness assumptions $a\#X_1, \ldots, a\#X_n$ into the derivation of $t = u$ from assumptions Δ, for atoms a that do not occur at all in

t, u or Δ. The square brackets denote *discharge* in the sense of natural deduction (as in implication introduction) of these extra assumptions $a\#X_1, \ldots, a\#X_n$.

The rules (**refl**), (**symm**) and (**tran**) ensure that equality is an equivalence relation, and rules (**cong[]**) and (**congf**) ensure that it is a congruence. The rules ($\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}}$), (**perm**) and (**fr**) are discussed in more detail below.

Examples of derivability of equality can be found in the rest of this section.

The ($\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}}$) rule: instantiating axioms

($\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}}$) allows us to permutatively rename atoms and to instantiate unknowns. This gives the effect that atoms in axioms can be understood to range over *any* (distinct) atoms, and unknowns can be understood to range over *any* terms. So

$$\frac{}{(\lambda[b]a)b = a[b \mapsto b]}\,(\mathbf{ax}_\beta) \qquad \frac{}{(\lambda[b]b)a = b[b \mapsto a]}\,(\mathbf{ax}_\beta)$$

are valid derivations in theory LAM (Example 2.2.17). The right derivation shows that substitution does not avoid capture, reflecting informal practice.

The use of the ($\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}}$) rule introduces new proof obligations on the freshness side-conditions in ∇, as the following derivations show:

$$\frac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{\lambda[a](ba) = b}\,(\mathbf{ax}_\eta) \qquad \frac{a\#a}{\lambda[a](aa) = a}\,(\mathbf{ax}_\eta)$$

The left derivation is valid but the right one is not, because $a\#a$ is not derivable.

Note that instantiation of axioms ($\mathbf{abs}{\mapsto}$) and ($\mathbf{ren}{\mapsto}$) from theory SUB (Example 2.2.17), which both mention distinct atoms a and b, can never identify these atoms. For example,

$$\frac{c\#X}{([c]Y)[c \mapsto X] = [c](Y[c \mapsto X])}\,(\mathbf{ax}_{\mathbf{abs}\mapsto})$$

is not a valid instance of ($\mathbf{abs}{\mapsto}$) (even when $c\#X$ is derivable) since permutations are bijective: there is no π such that both $\pi(a) = c$ and $\pi(b) = c$.

In informal practice, derivations are often presented in a calculational style, e.g. the sequence of equalities

$$\lambda x.(((\lambda x.y)x)x) =_\beta \lambda x.(yx) =_\eta y$$

represents that two terms can be related by (reading from left to right) first applying β-conversion followed by η-conversion. This is fully formally represented by the derivation in Figure 2.3. Reading the derivation bottom-up, the instance of (**tran**) in the conclusion introduces two equalities that correspond directly to the two equations in the above sequence of equalities. The derivation of the right equality takes care of η-equality: it instantiates the (η) axiom, and shows that the

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda[a]b)a = b[a \mapsto a]}\,(\mathbf{ax}_\beta) \qquad \dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{b[a \mapsto a] = b}\,(\mathbf{ax}_{\#\mapsto})}{(\lambda[a]b)a = b}\,(\mathbf{tran})}{((\lambda[a]b)a)a = ba}\,(\mathbf{congf})}{[a](((\lambda[a]b)a)a) = [a](ba)}\,(\mathbf{cong[]})}{\lambda[a](((\lambda[a]b)a)a) = \lambda[a](ba)}\,(\mathbf{congf}) \qquad \dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{\lambda[a](ba) = b}\,(\mathbf{ax}_\eta)}{\lambda[a](((\lambda[a]b)a)a) = b}\,(\mathbf{tran})$$

Figure 2.3 An example derivation in nominal algebra

freshness side-condition is satisfied. The derivation of the left equality takes care of the β-equality: it instantiates the (β) axiom, which introduces a substitution, and we show how this substitution is applied.

Remark 2.3.15. Another version of the $(\mathbf{ax}_{\nabla \vdash \mathbf{t}=\mathbf{u}})$ rule is possible, which uses the object-level action $\pi \cdot t$ instead of the meta-level action t^π:

$$\dfrac{\pi \cdot \nabla\sigma}{\pi \cdot t\sigma = \pi \cdot u\sigma}\,(\mathbf{ax}'_{\nabla \vdash \mathbf{t}=\mathbf{u}}).$$

However in this case, atoms in substitution σ are renamed according to π. For example, from the axiom $[a]X = [b]X$ it is immediate that $[b]a = [a]a$ is derivable with $(\mathbf{ax}_{[\mathbf{a}]\mathbf{X}=[\mathbf{b}]\mathbf{X}})$ where we choose $\pi = (b\ a)$ and $\sigma = [a/X]$. It is also derivable with $(\mathbf{ax}'_{[\mathbf{a}]\mathbf{X}=[\mathbf{b}]\mathbf{X}})$ but we must choose $\pi = (b\ a)$ and $\sigma = [b/X]$. We find this version less natural.

The (perm) rule: α-equivalence

The (**perm**) rule provides us with a concise way of expressing α-equivalence. To illustrate this, the following derivations are valid in CORE (the theory with no axioms):

$$\dfrac{\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#[b]b}\,(\#[]\mathbf{b}) \qquad \dfrac{}{b\#[b]b}\,(\#[]\mathbf{a})}{[a]a = [b]b}\,(\mathbf{perm}) \qquad\qquad \dfrac{\dfrac{a\#X}{a\#[b]X}\,(\#[]\mathbf{b}) \qquad \dfrac{}{b\#[b]X}\,(\#[]\mathbf{a})}{[a](b\ a)\cdot X = [b]X}\,(\mathbf{perm})$$

So $\vdash_{\mathsf{CORE}} [a]a = [b]b$ and $a\#X \vdash_{\mathsf{CORE}} [a](b\ a)\cdot X = [b]X$. To see that the instances of (**perm**) are valid, we note that $[a]a \equiv (b\ a)\cdot[b]b$ and $[a](b\ a)\cdot X \equiv (b\ a)\cdot[b]X$.

As another example, we show how we use (**perm**) rule to rename a bound variable in a λ-term. Consider the following derivation in the λ-calculus:

$$(\lambda x.xx)(\lambda x.\lambda y.xy) =_\beta (\lambda x.\lambda y.xy)(\lambda x.\lambda y.xy) =_\beta \lambda y.(\lambda x.\lambda y.xy)y =_\beta \lambda y.\lambda z.yz.$$

In the last step the bound variable x is implicitly renamed during β-reduction to avoid capture. Nominal algebra makes this explicit. We present the nominal algebra derivation of this last step as a calculation:

$$\begin{array}{ll}
 & \lambda[b]\underline{(\lambda[a]\lambda[b]ab)b} \\
= & \quad \{ \text{ axiom } (\beta) \ \} \\
 & \lambda[b]\underline{(\lambda[b]ab)[a \mapsto b]} \\
= & \quad \{ \text{ axiom } (\mathsf{f}{\mapsto}) \ \} \\
 & \lambda[b]\lambda(\underline{[b]ab})[a \mapsto b] \\
= & \quad \{ \ (\mathbf{perm}), \text{ since } \vdash b\#[c]ac \text{ and } \vdash c\#[c]ac \ \} \\
 & \lambda[b]\lambda\underline{([c]ac)[a \mapsto b]} \\
= & \quad \{ \text{ axiom } (\mathbf{abs}{\mapsto}), \text{ since } \vdash c\#b \ \} \\
 & \lambda[b]\lambda[c]\underline{(ac)[a \mapsto b]} \\
= & \quad \{ \text{ axiom } (\mathsf{f}{\mapsto}) \ \} \\
 & \lambda[b]\lambda[c](\underline{a[a \mapsto b]})(c[a \mapsto b]) \\
= & \quad \{ \text{ axiom } (\mathbf{var}{\mapsto}) \ \} \\
 & \lambda[b]\lambda[c]b(\underline{c[a \mapsto b]}) \\
= & \quad \{ \text{ axiom } (\#{\mapsto}), \text{ since } \vdash a\#c \ \} \\
 & \lambda[b]\lambda[c]bc
\end{array}$$

In each step of the calculation, we have indicated in the hint which derivation rule is applied and which freshness constraints it had to satisfy (if any), and we have underlined the subterm on which the axiom is applied. From this information the full derivation can be reconstructed using (**cong**[]), (**cong**f) and (**tran**).

As a final example we show that we can rename an atom which is substituted for using explicit substitution term-former sub from theory SUB (Example 2.2.17):

Lemma 2.3.16. $b\#X \vdash_{\mathsf{CORE}} X[a \mapsto T] = ((b\ a)\cdot X)[b \mapsto T]$

Proof. De-sugaring, we derive $\mathsf{sub}([a]X, T) = \mathsf{sub}([b](b\ a)\cdot X, T)$ from $b\#X$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a\#[a]X}(\#[]\mathbf{a}) \qquad \dfrac{b\#X}{b\#[a]X}(\#[]\mathbf{b})}{[b](b\ a)\cdot X = [a]X}(\mathbf{perm})}{[a]X = [b](b\ a)\cdot X}(\mathbf{symm})}{\mathsf{sub}([a]X, T) = \mathsf{sub}([b](b\ a)\cdot X, T)}(\mathbf{cong}\mathsf{f})$$

□

In Section 2.5 we will show that derivability in CORE precisely corresponds to α-equivalence on nominal terms, in the sense of nominal unification [UPG04, Figure 2] and nominal rewriting [FG07, p.13].

Remark 2.3.17. Instead of expressing (**perm**) as a derivation rule, we could also have added it as a mandatory axiom, such as

$$a\#X, b\#X \vdash (a\ b)\cdot X = X.$$

We did not do this here in order to keep a strict distinction between derivation rules and axioms.

The (fr) rule: introducing fresh atoms

The (**fr**) rule allows us to introduce a fresh atom into the derivation. It mirrors the generation of a fresh name in rules such as the $\forall$ right-introduction rule 'from $\Gamma \vdash \phi$ derive $\Gamma \vdash \forall x.\phi$ provided x is not free in Γ'. In a sequent style presentation of nominal algebra, (**fr**) would be

$$\frac{\Delta, a\#X_1, \ldots, a\#X_n \vdash t = u}{\Delta \vdash t = u} \quad (n \geq 1, a \notin t, u, \Delta).$$

To illustrate the extra power (**fr**) gives us, we consider a theory C with one axiom $a\#X \vdash X = a$.

Lemma 2.3.18. *We can derive* $\vdash_{\mathsf{C}} X = Y$ with (**fr**)*, but we cannot* without *it.*

Proof. The following is a derivation of $\vdash_{\mathsf{C}} X = Y$ with (**fr**):

$$\dfrac{\dfrac{\dfrac{[a\#X]^1}{X=a}\,(\mathbf{ax_{a\#X\vdash X=a}}) \qquad \dfrac{\dfrac{[a\#Y]^1}{Y=a}\,(\mathbf{ax_{a\#X\vdash X=a}})}{a=Y}\,(\mathbf{symm})}{X=Y}\,(\mathbf{tran})}{X=Y}\,(\mathbf{fr})^1$$

In the above derivation, the superscript number one ^{1} is an annotation associating the instance of the rule (**fr**) with the assumptions it discharges in the derivation.

In order to show that it is impossible to derive $\vdash_{\mathsf{C}} X = Y$ without (**fr**), we show the more general property that we can never conclude equations of the form $X = t$ or $t = X$, where $t \not\equiv X$, without the use of (**fr**).

We proceed by contradiction. Suppose we *can* conclude equations of the form $X = t$ and $t = X$ where $t \not\equiv X$. Looking at the structure of the derivation rules, it is easy to see that the (**refl**), (**cong[]**) and (**congf**) rules could not have been applied. Also we can see that we could never have applied the $(\mathbf{ax_{a\#X\vdash X=a}})$ and (**perm**) rules: the derivations of the freshness side-conditions belonging to these rules require the presence of at least one freshness assumption, but this is impossible since we work in the empty freshness context. So the only rules that could have been applied are (**symm**) and (**tran**).

We only consider the $X = t$ case, the $t = X$ case is completely analogous. Suppose the derivation of $X = t$ concludes in:

- (**symm**). Then $X = t$ is derived from $t = X$, which fits the pattern.
- (**tran**). Then $X = t$ is derived from $X = u$ and $u = t$. Now there are two cases: either $u \not\equiv X$, in which case the first equation fits the pattern, or $u \equiv X$, in which case the second equation fits the pattern.

So using (**symm**) and (**tran**) we cannot conclude the derivation. We have arrived at a contradiction and the result follows. $\square$

Another example that shows the extra power of the (**fr**) rule is derivability of $X[a \mapsto a] = X$ in theory SUB (Example 2.2.17).

Lemma 2.3.19. $\vdash_{\mathsf{SUB}} X[a \mapsto a] = X$.

Proof. We derive $X[a \mapsto a] = X$ as follows:

$$\begin{prooftree}
\AxiomC{}
\RightLabel{$(\#[]\mathbf{a})$}
\UnaryInfC{$a\#[a]X$}
\AxiomC{$[b\#X]^1$}
\RightLabel{$(\#[]\mathbf{b})$}
\UnaryInfC{$b\#[a]X$}
\RightLabel{$(\mathbf{perm})$}
\BinaryInfC{$[b](b\ a)\cdot X = [a]X$}
\RightLabel{$(\mathbf{symm})$}
\UnaryInfC{$[a]X = [b](b\ a)\cdot X$}
\RightLabel{$(\mathbf{congf})$}
\UnaryInfC{$X[a \mapsto a] = ((b\ a)\cdot X)[b \mapsto a]$}
\AxiomC{$[b\#X]^1$}
\RightLabel{$(\#\mathbf{X})$}
\UnaryInfC{$a\#(b\ a)\cdot X$}
\RightLabel{$(\mathbf{ax_{ren\mapsto}})$}
\UnaryInfC{$((b\ a)\cdot X)[b \mapsto a] = X$}
\RightLabel{$(\mathbf{tran})$}
\BinaryInfC{$X[a \mapsto a] = X$}
\RightLabel{$(\mathbf{fr})^1$}
\UnaryInfC{$X[a \mapsto a] = X$}
\end{prooftree}$$

The instance of the $(\mathbf{ren}\mapsto)$ axiom is valid since we have used the fact that $X \equiv (a\ b)\cdot(b\ a)\cdot X$ in the right-hand side of the equation. $\square$

We conjecture that we cannot derive $X[a \mapsto a] = X$ in SUB *without* (**fr**). Intuitively this is because to α-rename a so that we can use $(\mathbf{ren}\mapsto)$, we need an atom fresh for X.

2.4 Proof-Theoretical Results

We provide a number of proof-theoretical results for freshness and equality that will be used throughout this thesis.

Lemma 2.4.1. *It is decidable whether $\Delta \vdash a\#t$, for any Δ, a and t.*

Proof. We observe that the derivation rules of Figure 2.1 are syntax-directed. $\square$

The rules for freshness also hold in the opposite direction:

Lemma 2.4.2.

1. *If* $\Delta \vdash a\#X$ *then* $a\#X \in \Delta$.
2. *If* $\Delta \vdash a\#\pi \cdot X$ *then* $\Delta \vdash \pi^{-1}(a)\#X$.
3. *If* $\Delta \vdash a\#[b]t$ *then* $\Delta \vdash a\#t$.
4. *If* $\Delta \vdash a\#\mathsf{f}(t_1, \ldots, t_n)$ *then* $\Delta \vdash a\#t_i$*, for all* i*,* $1 \leq i \leq n$.

Proof. By an induction on the structure of the derivation rules in Figure 2.1. □

Definition 2.4.3. We naturally extend notation for t^π and Δ^π to theories: given a theory $\mathsf{T} = (\Sigma, Ax)$, write T^π for (Σ, Ax^π) such that $\nabla^\pi \vdash t^\pi = u^\pi \in Ax^\pi$ if and only if $\nabla \vdash t = u \in Ax$.

Lemma 2.4.4. *If* $\Delta \vdash_{\mathsf{T}} t = u$ *then* $\Delta \vdash_{\mathsf{T}^\pi} t = u$.

Proof. By induction on derivations. The only non-trivial case is $(\mathbf{ax}_{\nabla \vdash \mathbf{t}=\mathbf{u}})$, where after applying the inductive hypothesis we need to show that

$$\Delta \vdash_{\mathsf{T}^\pi} \nabla^{\pi'}\sigma \quad \text{implies} \quad \Delta \vdash_{\mathsf{T}^\pi} t^{\pi'}\sigma = u^{\pi'}\sigma.$$

By Lemma 2.3.9, it is equivalent to show that

$$\Delta \vdash_{\mathsf{T}^\pi} \nabla^{\pi^{\pi' \circ \pi^{-1}}}\sigma \quad \text{implies} \quad \Delta \vdash_{\mathsf{T}^\pi} t^{\pi^{\pi' \circ \pi^{-1}}}\sigma = u^{\pi^{\pi' \circ \pi^{-1}}}\sigma.$$

This follows by $(\mathbf{ax}_{\nabla^\pi \vdash \mathbf{t}^\pi = \mathbf{u}^\pi})$ taking permutation $\pi' \circ \pi^{-1}$ and substitution σ. □

The following result uses the principle of *logical ZFA equivariance*: if an assertion is true of some arguments, then it is also true of those arguments with some atoms permuted provided the axiom of choice is not used (in the cases we are interested in, it is not). This principle is treated formally in Appendix A.

Theorem 2.4.5 (Meta-level equivariance)**.** *For any* π*:*

1. *if* $\Delta \vdash a\#t$ *then* $\Delta^\pi \vdash \pi(a)\#t^\pi$*;*
2. *if* $\Delta \vdash_{\mathsf{T}} t = u$ *then* $\Delta^\pi \vdash_{\mathsf{T}} t^\pi = u^\pi$.

Proof. For the second case suppose that $\Delta \vdash_{\mathsf{T}} t = u$. By ZFA equivariance also $\Delta^\pi \vdash_{\mathsf{T}^\pi} t^\pi = u^\pi$. Then by Lemma 2.4.4 we obtain

$$\Delta^\pi \vdash_{\mathsf{T}^{\pi^{\pi^{-1}}}} t^\pi = u^\pi.$$

Using Lemma 2.3.9 we can easily show that $\mathsf{T}^{\pi^{\pi^{-1}}}$ is syntactically equivalent to T, so we obtain $\Delta^\pi \vdash_{\mathsf{T}} t^\pi = u^\pi$ as required.

The proof of the first case is simpler, since it does not refer to theory T. □

We can permute atoms in freshnesses and equations at the object-level without changing the freshness context:

Theorem 2.4.6 (Object-level equivariance)**.** *For any π:*

1. *if $\Delta \vdash a\#t$ then $\Delta \vdash \pi(a)\#\pi \cdot t$;*
2. *if $\Delta \vdash_{\mathsf{T}} t = u$ then $\Delta \vdash_{\mathsf{T}} \pi \cdot t = \pi \cdot u$.*

Proof. By induction on the structure of derivations. We consider the most interesting cases only. Suppose the derivation concludes in...

- $(\#\mathbf{X})$. Then $a\#\pi' \cdot X$ is derived from $\pi'^{-1}(a)\#X$, for some $\pi' \neq \iota$, and we need to show $\pi(a)\#\pi \cdot (\pi' \cdot X)$. By Lemma 2.3.2, this is equivalent to $\pi(a)\#(\pi \circ \pi') \cdot X$. We continue by case distinction:
  - If $\pi \circ \pi' = \iota$ then $\pi(a)\#(\pi \circ \pi') \cdot X$ is equivalent to the assumption $\pi'^{-1}(a)\#X$, since $\pi = \pi'^{-1}$ by basic permutation group theory.
  - If $\pi \circ \pi' \neq \iota$ then $\pi(a)\#(\pi \circ \pi') \cdot X$ follows from $(\pi \circ \pi')^{-1}(\pi(a))\#X$ by $(\#\mathbf{X})$ (which may now be applied). This is equivalent to the assumption $\pi'^{-1}(a)\#X$, since $(\pi \circ \pi')^{-1}(\pi(a)) = \pi'^{-1}(a)$.
- $(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$. Then $t^{\pi'}\sigma = u^{\pi'}\sigma$ is derived and $\Delta \vdash_{\mathsf{T}} \nabla^{\pi'}\sigma$. We need to derive $\pi \cdot t^{\pi'}\sigma = \pi \cdot u^{\pi'}\sigma$ from Δ. By Lemma 2.3.7, $\pi \cdot t^{\pi'}\sigma = \pi \cdot u^{\pi'}\sigma$ is equivalent to $(\pi \cdot t^{\pi'})\sigma = (\pi \cdot u^{\pi'})\sigma$. Now let σ' map each $X \in \nabla, t, u$ to $\pi \cdot X$, then by Lemma 2.3.10 it suffices to derive
$$t^{\pi'\pi}(\sigma' \circ \sigma) = u^{\pi'\pi}(\sigma' \circ \sigma).$$
By Lemma 2.3.9, this is equivalent to $t^{\pi\circ\pi'}(\sigma' \circ \sigma) = u^{\pi\circ\pi'}(\sigma' \circ \sigma)$. Now this follows from $\nabla^{\pi\circ\pi'}(\sigma' \circ \sigma)$ by $(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$ with permutation $\pi \circ \pi'$ and substitution $\sigma' \circ \sigma$. By Lemmas 2.3.7, 2.3.10 and 2.3.9 this is equivalent to $\pi \cdot \nabla^{\pi'}\sigma$. We are done since this follows from Δ by the inductive hypothesis.
- $(\mathbf{fr})$. Then $\Delta, a\#X_1, \ldots, a\#X_n \vdash_{\mathsf{T}} t = u$ for some $a \notin \Delta, t, u$ and we assume the inductive hypothesis of this derivation. If $\pi(a) = a$ there is no problem since then $a \notin \Delta, \pi \cdot t, \pi \cdot u$ and we may extend the derivation with $(\mathbf{fr})$.
  However, suppose $\pi(a) \neq a$ and so (possibly) $a \in \pi \cdot t, \pi \cdot u$. We observe that the predicate

  > "if the labelled tree Π is a valid derivation of $\Delta \vdash_{\mathsf{T}} t = u$, then for all permutations π' there are derivations of $\Delta \vdash_{\mathsf{T}} \pi' \cdot t = \pi' \cdot u$"

  has free variables Π, Δ, T, t and u.
  By ZFA equivariance (Theorem A.2.5), the precidate above holds of $\Pi^{(a'\ a)}$, $\Delta^{(a'\ a)}$, $\mathsf{T}^{(a'\ a)}$, $t^{(a'\ a)}$ and $u^{(a'\ a)}$ (the informal notation $\Pi^{(a'\ a)}$ denotes Π in which all atoms are permuted according to $(a'\ a)$). Now using Lemma 2.4.4 we deduce the inductive hypothesis of $\Delta, a'\#X_1, \ldots, a'\#X_n \vdash_{\mathsf{T}} t = u$ for any $a' \notin \Delta, t, u, \pi$. Then $\Delta, a'\#X_1, \ldots, a'\#X_n \vdash_{\mathsf{T}} \pi \cdot t = \pi \cdot u$ and we extend the derivation with $(\mathbf{fr})$ to deduce $\Delta \vdash_{\mathsf{T}} \pi \cdot t = \pi \cdot u$ as required.

□

Note that instead of using ZFA equivariance in the (**fr**) case of the above proof, we could have proven the theorem by an induction on the *depth* of derivations. We prefer equivariance however, because it allows for a more compact and readable proof.

Definition 2.4.7. We write $\mathrm{ds}(\pi, \pi')$ for the **difference set** $\{a \mid \pi(a) \neq \pi'(a)\}$ of π and π'. We write $\mathrm{ds}(\pi, \pi')\#t$ for the set of freshnesses $\{a\#t \mid \pi(a) \neq \pi'(a)\}$.

Lemma 2.4.8. *If* $\Delta \vdash \mathrm{ds}(\pi, \pi')\#t$ *then* $\Delta \vdash_{\mathsf{CORE}} \pi \cdot t = \pi' \cdot t$.

Proof. We work by induction on the number of elements in $\mathrm{ds}(\pi, \pi')$. If this set is empty then $\pi = \pi'$ and the result follows easily by (**refl**). Now suppose $a \in \mathrm{ds}(\pi, \pi')$. We construct a partial derivation of the proof obligation:

$$\dfrac{\pi \cdot t = ((\pi(a)\ \pi'(a)) \circ \pi') \cdot t \qquad \dfrac{\pi(a)\#\pi' \cdot t \quad \pi'(a)\#\pi' \cdot t}{((\pi(a)\ \pi'(a)) \circ \pi') \cdot t = \pi' \cdot t}\,(\mathbf{perm})}{\pi \cdot t = \pi' \cdot t}\,(\mathbf{tran})$$

The following proof obligations remain:

1. $\pi \cdot t = ((\pi(a)\ \pi'(a)) \circ \pi') \cdot t$ follows from $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi')\#t$ by the inductive hypothesis, provided $|\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi')| < |\mathrm{ds}(\pi, \pi')|$. This condition is satisfied, since $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi') = \mathrm{ds}(\pi, \pi') \setminus \{a\}$. Finally, the remaining proof obligation $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi')\#t$ follows from assumption $\mathrm{ds}(\pi, \pi')\#t$.
2. $\pi(a)\#\pi' \cdot t$ follows from $\pi'^{-1}(\pi(a))\#t$ by object-level equivariance (Theorem 2.4.6). Now if $\pi'^{-1}(\pi(a)) \in \mathrm{ds}(\pi, \pi')$, this follows from assumptions $\mathrm{ds}(\pi, \pi')\#t$. It turns out that this is the case: $\pi'^{-1}(\pi(a)) \in \mathrm{ds}(\pi, \pi')$ when $\pi(\pi'^{-1}((\pi(a))) \neq \pi(a)$, and, using the fact that $\neq$ is invariant under permutation, this follows from the assumption $\pi(a) \neq \pi'(a)$.
3. $\pi'(a)\#\pi' \cdot t$ follows from $a\#t$ by object-level equivariance (Theorem 2.4.6). This follows directly from assumption $\mathrm{ds}(\pi, \pi')\#t$, since $a \in \mathrm{ds}(\pi, \pi')$.

□

Derivability of equalities satisfies the following congruence property:

Lemma 2.4.9 (Congruence)**.** *For any X, v:*

$$\textit{If} \quad \Delta \vdash_{\mathsf{T}} t = u \quad \textit{then} \quad \Delta \vdash_{\mathsf{T}} v[t/X] = v[u/X].$$

Proof. By an induction on the structure of v. □

We can substitute terms for unknowns provided those terms violate no freshness assumptions made on the unknowns:

Theorem 2.4.10 (Meta-level substitution)**.** *For any Δ', Δ, σ such that $\Delta' \vdash \Delta\sigma$:*

1. *if $\Delta \vdash a\#t$ then $\Delta' \vdash a\#t\sigma$;*
2. *if $\Delta \vdash_{\mathsf{T}} t = u$ then $\Delta' \vdash_{\mathsf{T}} t\sigma = u\sigma$.*

Proof. Natural deduction derivations are such that the conclusion of one derivation may be 'plugged in' to an assumption in another derivation. For (**#X**) we use object-level equivariance (Theorem 2.4.6). For (**fr**) we might have to use ZFA equivariance (Theorem A.2.5) to rename the freshly chosen atom a if it is mentioned by σ. □

The above condition $\Delta' \vdash \Delta\sigma$ ensures that $\Delta\sigma$ is *consistent*, in the sense that $a\#\sigma(X)$ is derivable for each $a\#X \in \Delta$.

Corollary 2.4.11. *For closed t, u, if $\vdash_{\mathsf{T}} t = u$ then it has a derivation that does not mention any unknowns or instances of* (**#X**) *or* (**fr**)*.*

Proof. Let Π be a derivation of $\vdash_{\mathsf{T}} t = u$. Now take c to be an atom that does not occur anywhere in Π, and let Π' be Π in which:

- each unknown X is mapped to c;
- each instance of (**#X**) is replaced by (**#ab**); that is, each instance of (**#X**) is of the form

$$\frac{[\pi^{-1}(a)\#X]}{a\#\pi\cdot X}\ (\mathbf{\#X}),$$

  where square brackets denote discharge of the freshness assumption. This is replaced by

$$\frac{}{a\#c}\ (\mathbf{\#ab}).$$

  Note that we write c instead of $\pi(c)$ since $\pi(c) = c$.

Using Theorem 2.4.10 it is not hard to see that Π' is a valid derivation of $t = u$ that does not mention unknowns. Since the instances of (**fr**) do not discharge any assumptions anymore (they have been removed in the previous step), these instances be safely removed. The result follows. □

Another corollary of Theorem 2.4.10 is that we can *weaken* the freshness context in derivations:

Corollary 2.4.12 (Weakening)**.** *If $\Delta \subseteq \Delta'$ then:*

1. *if $\Delta \vdash a\#t$ then $\Delta' \vdash a\#t$;*
2. *if $\Delta \vdash_{\mathsf{T}} t = u$ then $\Delta' \vdash_{\mathsf{T}} t = u$.*

Proof. By meta-level substitution (Theorem 2.4.10), using the empty substitution $\sigma = []$, the proof obligations follow from $\Delta' \vdash \Delta$. So for each $a\#X \in \Delta$, we must show $\Delta' \vdash a\#X$. This is trivial since $\Delta \subseteq \Delta'$. □

Sometimes the freshness contexts Δ may also be *strengthened*. We need some terminology in order to state this.

Definition 2.4.13. Let Δ' and Δ be freshness contexts, and S be a set of terms. Say that Δ' **freshly extends** Δ **with atoms not in** S when $\Delta \subseteq \Delta'$ and $a \notin \Delta, S$ for each $a\#X \in \Delta' \setminus \Delta$. If S is empty we just say that Δ' freshly extends Δ.

For example:

- $b\#X, a\#X$ freshly extends $a\#X$.
- $b\#X, a\#X$ freshly extends $a\#X$ with atoms not in a.
- $b\#X, a\#X$ does not freshly extend $a\#X$ with atoms not in b.
- $a\#Y, a\#X$ does not freshly extend $a\#X$.

Theorem 2.4.14 (Strengthening)**.**

1. *If $\Delta' \vdash a\#t$ and Δ' freshly extends Δ with atoms not in a, t, then $\Delta \vdash a\#t$.*
2. *If $\Delta' \vdash_{\mathsf{T}} t = u$ and Δ' freshly extends Δ with atoms not in t, u, then $\Delta \vdash_{\mathsf{T}} t = u$.*

Proof. For the freshness case, we inductively transform a derivation of $\Delta' \vdash a\#t$ to a derivation of $\Delta \vdash a\#t$:

- If $\Delta' \vdash a\#X$ by assumption, then $a\#X \in \Delta'$. We proceed by case distinction:
  - If $a\#X \in \Delta$, then $\Delta \vdash a\#X$ by assumption.
  - If $a\#X \in \Delta' \setminus \Delta$, then the result holds vacuously: $a\#X$ does not freshly extend Δ with respect to a, X, since $a \in \Delta, a, X$.
- $(\#\mathbf{X})$: Suppose $\Delta' \vdash a\#\pi \cdot X$ is derived using $(\#\mathbf{X})$, $\pi \neq \iota$, and Δ' freshly extends Δ with atoms not in $a, \pi \cdot X$. Then $\Delta' \vdash \pi^{-1}(a)\#X$ by assumption and Δ' freshly extends Δ with atoms not in $\pi^{-1}(a), X$. By the inductive hypothesis $\Delta \vdash \pi^{-1}(a)\#X$, and by $(\#\mathbf{X})$ we conclude $\Delta \vdash a\#\pi \cdot X$.
- $(\#\mathbf{ab})$ and $(\#[]\mathbf{a})$ carry over directly.
- $(\#[]\mathbf{b})$ is straightforward using the inductive hypothesis and the fact that if $a \notin [b]t$ then $a \notin t$.
- $(\#\mathsf{f})$ is straightforward using the inductive hypothesis and the fact that if $a \notin \mathsf{f}(t_1, \ldots, t_n)$ then $a \notin t_i$ for all i.

For the equational case, we note that the $(\mathbf{fr})$ rule precisely introduces a 'Δ' freshly extending Δ with atoms not in t, u'. We just extend the derivation of $\Delta' \vdash_{\mathsf{T}} t = u$ with instances of the rule $(\mathbf{fr})$ to obtain a derivation of $\Delta \vdash_{\mathsf{T}} t = u$.

We do this as follows. Let $a_1, \ldots, a_n$ be the atoms mentioned in $\Delta' \setminus \Delta$, and let $\Delta_{a_i} = \{a_i\#X \mid a_i\#X \in \Delta' \setminus \Delta\}$ for each a_i. Then

$$\Delta' = \Delta \cup \Delta_{a_1} \cup \cdots \cup \Delta_{a_n}.$$

$$\frac{}{a \approx_{\Delta} a}\,(\mathbf{Ax}) \qquad \frac{\Delta \vdash \mathrm{ds}(\pi', \pi)\# X}{\pi \cdot X \approx_{\Delta} \pi' \cdot X}\,(\mathbf{Ds})$$

$$\frac{t \approx_{\Delta} u}{[a]t \approx_{\Delta} [a]u}\,(\mathbf{Absaa}) \qquad \frac{(b\ a)\cdot t \approx_{\Delta} u \quad \Delta \vdash b\# t}{[a]t \approx_{\Delta} [b]u}\,(\mathbf{Absab})$$

$$\frac{t_1 \approx_{\Delta} u_1 \quad \cdots \quad t_n \approx_{\Delta} u_n}{\mathsf{f}(t_1, \dots, t_n) \approx_{\Delta} \mathsf{f}(u_1, \dots, u_n)}\,(\mathbf{F})$$

Figure 2.4 Syntax-directed rules for CORE

From the definition of freshly extending, we also know $a_i \notin t, u, \Delta, \Delta_{a_j}$ for each i, j, $j \neq i$. We extend the derivation of $\Delta' \vdash_{\mathsf{T}} t = u$ to one of $\Delta \vdash_{\mathsf{T}} t = u$ using n instances of (**fr**), each instance i corresponds to a_i removing Δ_{a_i}. □

2.5 α-Equivalence with Meta-Variables

In Example 2.2.17 we defined CORE as a family of nominal algebra theories with no axioms (one for each signature). In Subsection 2.3.2 we have given some examples to show that the (**perm**) rule expresses α-equivalence with meta-variables.

In this section we will prove that theory CORE precisely corresponds to the existing syntax-directed notion of α-equivalence on nominal terms from [UPG04, FG07]. We will use this correspondence to show that CORE-equality is decidable.

Decidability of equality in CORE is important, because we want to show that CORE is *consistent* (does not equate *all* terms; Corollary 2.5.5). Furthermore, in Chapter 4 we want to show that equality up to axioms for capture-avoiding substitution is decidable. If we had been unable to determine equality and inequality of terms *without* any axioms then our project would be doomed from the start.

Definition 2.5.1 was introduced in [UPG04, Figure 2]; the proofs are modelled on a method presented in [FG07, p.13].

Recall from Definition 2.4.7 that we write $\mathrm{ds}(\pi, \pi')$ for the difference set of π and π'.

Definition 2.5.1. Let $t \approx_{\Delta} u$ be an ordered tuple of a term t, a freshness context Δ, and a term u. Let the **derivable equalities of** $t \approx_{\Delta} u$ be inductively defined by the rules in Figure 2.4.

Syntax-directed equality $\approx_{\Delta}$ is transitive:

Lemma 2.5.2. *If $t \approx_{\Delta} u$ and $u \approx_{\Delta} v$ then $t \approx_{\Delta} v$.*

Proof. By induction on the *size* of t (we do not count permutations in the size).

For each t we exploit the syntax-directed nature of the rules of Figure 2.4 to determine the possibilities for u and v.

For the abstraction case $t \equiv [a]t'$ we need the following properties:

- If $t \approx_{\vartriangle} u$ then $\pi \cdot t \approx_{\vartriangle} \pi \cdot u$. This follows by induction on the structure of derivations of $t \approx_{\vartriangle} u$.
- If $\Delta \vdash \mathrm{ds}(\pi, \pi')\#t$ then $\pi \cdot t \approx_{\vartriangle} \pi' \cdot t$. By induction on the structure of t.
- If $t \approx_{\vartriangle} u$ then $\Delta \vdash a\#t$ if and only if $\Delta \vdash a\#u$. By induction on the structure of t.

Full details can be found in [FG07, Proof of Lemma 23(2)]. □

We use Lemma 2.5.2 to show that CORE is equivalent to the syntax-directed equality of Figure 2.4.

Theorem 2.5.3 (Equivalence of CORE and $\approx_{\vartriangle}$)**.** *$\Delta \vdash_{\mathsf{CORE}} t = u$ if and only if $t \approx_{\vartriangle} u$ is derivable using the rules of Figure 2.4.*

Proof. The left-to-right direction is by induction on the structure of nominal algebra derivations of $\Delta \vdash_{\mathsf{CORE}} t = u$. By the inductive hypothesis it suffices to show:

- $t \approx_{\vartriangle} t$ (is derivable). This follows by a trivial induction on t.
- If $t \approx_{\vartriangle} u$ then $u \approx_{\vartriangle} t$. By induction on derivations of $t \approx_{\vartriangle} u$. Except for (**Absab**), all cases are trivial using the inductive hypothesis, since they are symmetric. For the remaining case, suppose $[a]t \approx_{\vartriangle} [b]u$ is derived from $(b\ a) \cdot t \approx_{\vartriangle} u$ and $\Delta \vdash b\#t$. We need to show $[b]u \approx_{\vartriangle} [a]t$. By transitivity of $\approx_{\vartriangle}$ (Lemma 2.5.2), this follows from

  $$[b]u \approx_{\vartriangle} [b](b\ a) \cdot t \quad \text{and} \quad [b](b\ a) \cdot t \approx_{\vartriangle} [a]t.$$

  By (**Absaa**) $[b]u \approx_{\vartriangle} [b](b\ a) \cdot t$ follows from $u \approx_{\vartriangle} (b\ a) \cdot t$; by the inductive hypothesis this follows from the assumption $(b\ a) \cdot t \approx_{\vartriangle} u$. By (**Absab**) $[b](b\ a) \cdot t \approx_{\vartriangle} [a]t$ follows from $t \approx_{\vartriangle} t$ and $\Delta \vdash a\#(b\ a) \cdot t$. We have already shown $t \approx_{\vartriangle} t$. $\Delta \vdash a\#(b\ a) \cdot t$ follows from the assumption $\Delta \vdash b\#t$ by object-level equivariance (Theorem 2.4.6).
- If $t \approx_{\vartriangle} u$ and $u \approx_{\vartriangle} v$ then $t \approx_{\vartriangle} v$. This is Lemma 2.5.2.
- If $t \approx_{\vartriangle} u$ then $[a]t \approx_{\vartriangle} [a]u$. This is (**Absaa**).
- If $t \approx_{\vartriangle} u$ then $\mathsf{f}(t_1, \ldots, t, \ldots, t_n) \approx_{\vartriangle} \mathsf{f}(t_1, \ldots, u, \ldots, t_n)$. This is an instance of (**F**), using the fact that $t_i \approx_{\vartriangle} t_i$ for all i.
- If $\Delta \vdash a\#t$ and $\Delta \vdash b\#t$ then $(a\ b) \cdot t \approx_{\vartriangle} t$. By induction on t.
- If $t \approx_{\vartriangle, a\#X_1, \ldots, a\#X_n} u$ where $a \notin t, u, \Delta$ then $t \approx_{\vartriangle} u$. By straightforward induction on the structure of derivations of $t \approx_{\vartriangle, a\#X_1, \ldots, a\#X_n} u$. The case of (**Absab**) uses strengthening (Theorem 2.4.14) to strengthen the assumption $\Delta, a\#X_1, \ldots, a\#X_n \vdash c\#t$ to $\Delta \vdash c\#t$.

For the right-to-left direction we work by induction on derivations of $t \approx_{\Delta} u$. By the inductive hypothesis it suffices to show:

- $\Delta \vdash_{\mathsf{CORE}} a = a$. This is an instance of (**refl**).
- If $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$ then $\Delta \vdash_{\mathsf{CORE}} \pi \cdot X = \pi' \cdot X$. This is a direct instance of Lemma 2.4.8.
- If $\Delta \vdash_{\mathsf{CORE}} t_i = u_i$ for $1 \leq i \leq n$, then $\Delta \vdash_{\mathsf{CORE}} \mathsf{f}(t_1, \ldots, t_n) = \mathsf{f}(u_1, \ldots, u_n)$. Using a number of instances of (**tran**) and (**congf**).
- If $\Delta \vdash_{\mathsf{CORE}} t = u$ then $\Delta \vdash_{\mathsf{CORE}} [a]t = [a]u$. This is (**cong**[]).
- If $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t = u$ and $\Delta \vdash b\#t$ then $\Delta \vdash_{\mathsf{CORE}} [a]t = [b]u$. Suppose that Π and Π' are derivations of $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t = u$ and $\Delta \vdash b\#t$ respectively. Then the following is a derivation of $\Delta \vdash_{\mathsf{CORE}} [a]t = [b]u$:

$$
\dfrac{
 \dfrac{
   \dfrac{\dfrac{\overset{\vdots\ \Pi'}{b\#t}}{b\#[a]t}\,(\#[]\mathbf{b}) \qquad \dfrac{}{a\#[a]t}\,(\#[]\mathbf{a})}{[b](b\ a)\cdot t=[a]t}\,(\mathbf{perm})
 }{[a]t=[b](b\ a)\cdot t}\,(\mathbf{symm})
 \qquad
 \dfrac{\overset{\vdots\ \Pi}{(b\ a)\cdot t=u}}{[b](b\ a)\cdot t=[b]u}\,(\mathbf{cong}[])
}{[a]t=[b]u}\,(\mathbf{tran})
$$

□

As direct corollaries of Theorem 2.5.3, we obtain syntactic criteria for determining equality in CORE, and consistency of CORE.

Corollary 2.5.4 (Decidability of CORE)**.** $\Delta \vdash_{\mathsf{CORE}} t = u$ *precisely when one of the following holds:*

1. $t \equiv a$ *and* $u \equiv a$.
2. $t \equiv \pi \cdot X$ *and* $u \equiv \pi' \cdot X$ *and* $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$.
3. $t \equiv [a]t'$ *and* $u \equiv [a]u'$ *and* $\Delta \vdash_{\mathsf{CORE}} t' = u'$.
4. $t \equiv [a]t'$ *and* $u \equiv [b]u'$ *and* $\Delta \vdash b\#t'$ *and* $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t' = u'$.
5. $t \equiv \mathsf{f}(t_1, \ldots, t_n)$ *and* $u \equiv \mathsf{f}(u_1, \ldots, u_n)$ *and* $\Delta \vdash_{\mathsf{CORE}} t_i = u_i$ *for* $1 \leq i \leq n$.

Proof. By Theorem 2.5.3 it suffices to inspect the rules for $t \approx_{\Delta} u$, which are just a rendering of the above criteria in terms of derivation rules. □

Corollary 2.5.5 (Consistency of CORE)**.** *For all* Δ *there are* t *and* u *such that* $\Delta \nvdash_{\mathsf{CORE}} t = u$.

Proof. By Corollary 2.5.4, $\Delta \vdash_{\mathsf{CORE}} a = b$ is never derivable. □

The following technical corollary will be useful later (in the proof of Lemma 4.3.9) and has a simple proof using Corollary 2.5.4:

Corollary 2.5.6. *If* $\Delta \vdash_{\mathsf{CORE}} t = u$ *then* $\Delta \vdash a\#t$ *if and only if* $\Delta \vdash a\#u$.

Proof. By induction on t using the syntactic criteria of Corollary 2.5.4. □

2.6 Conclusions

Nominal terms embrace the difference between the object-level and the meta-level. There are two classes of variables, atoms a and unknowns X. Substitution for X can capture abstractions by a. For instance the syntactic equality

$$(\lambda[a]X)[a/X] \equiv \lambda[a]a$$

formally reflects informal practice where instantiation in $\lambda x.t$ of t to x yields $\lambda x.x$. Freshness side-conditions of the form $a\#X$ manage the interaction between the two levels and 'recover' capture-avoidance where this is required.

Nominal algebra stakes a claim to nominal terms as a *logical* system, and the support for binding allows us to reflect binding in logic directly.

2.6.1 Related Work

A number of solutions have been proposed to formalise equations with binding and meta-variables.

Nominal techniques

The derivation rules for freshness and equality (Figures 2.1 and 2.2) are inspired by the rules of freshness and α-equivalence from nominal unification [UPG04, Fig. 2 on page 480] and nominal rewriting [FG07, Definition 6 on page 926]. Our system *generalises* this theory of α-equivalence to arbitrary theories with binding by extending the rules of equality with the possibility to instantiate axioms and to introduce fresh atoms into a derivation.

Nominal algebra has similarities to nominal logic [Pit03] and is closely related to the recently conceived nominal equational logic [CP07].[2] For instance our rules (**perm**) and (**fr**) correspond to axioms (**F1**) and (**F4**) from [Pit03, page 191], and our rules ($\#$**X**) and (**fr**) correspond to the rules ($\#$-EQUIVAR) and (ATM-ELIM) from [CP07, Fig. 5 on page 238]. However, there are some significant differences:

- These approaches do not use nominal terms. In nominal logic, atoms and abstractions are modelled using the sorts of the ambient first-order logic framework. In nominal equational logic, atoms and abstractions can be modelled using families of term-formers, indexed by an infinite collection of atoms.

[2]In fact, existing work on nominal algebra has influenced the development of nominal equational logic (as the authors acknowledge in [CP07, page 226]).

- The notion of freshness is slightly different: it does not correspond precisely to $x \notin \mathit{fv}(t)$, like the notion of freshness in nominal algebra. In nominal logic and nominal equational logic more atoms are fresh for a term than there are in nominal algebra.

  Consider for example the informal derivation

  $$\lambda x.(((\lambda x.y)x)x) =_\beta \lambda x.(yx) =_\eta y$$

  from Subsection 2.3.2. Both nominal algebra and nominal equational logic can formally represent this derivation. But now consider the following 'incorrect' informal derivation

  $$\lambda x.(((\lambda x.y)x)x) =_\eta (\lambda x.y)x =_\beta y.$$

  This derivation is incorrect because the side-condition $x \notin \mathit{fv}((\lambda x.y)x)$ of the η-equality is not satisfied. Nominal algebra cannot represent this derivation, but nominal equational logic can: the representation of x is fresh for the representation of $(\lambda x.y)x$. In Chapter 3 we treat this notion of freshness in more detail.

- Regarding nominal logic it is important to mention that this is a *first-order* logic (with equality) while nominal algebra is an *equational* logic. Comparing these logics is like comparing standard first-order logic with equality to equational logic.

Higher-order techniques

The theory of contexts [Mic01] can be used to axiomatise systems with binding. So, differently, can higher-order algebra [Mei92]. So indeed can simply-typed λ-calculus [Bar00]. These systems are different and intended for different purposes but they share a core which is in essence simply-typed λ-terms up to $\alpha\beta\eta$-equivalence. Just like nominal terms, this richer term-language gives more expressivity which can be used to give stronger axioms. However, meta-variables are represented by *function* variables, and this inherits some distinctive features from their intended functional semantics:

- You have to choose the arity of your unknown in advance. A function variable

  $$F : \overbrace{\mathbb{T} \to \cdots \to \mathbb{T}}^{n} \to \mathbb{T}$$

  can be interpreted as an unknown n-ary predicate — but *which* n? Thus these logics distribute meta-variables across many types.

- Perhaps more importantly, instantiation of these variables avoids capture. This has a side-effect that it is not possible to represent a meta-variable uniformly across contexts. For example to represent a meta-variable ϕ in the context we can intuitively express as $\forall x.\phi$, it suffices to write $\forall\lambda x.F(x)$ where $F : \mathbb{T} \to \mathbb{T}$; but to represent ϕ in the context $\forall x.\forall y.\phi$ we must write $\forall\lambda x.\forall\lambda y.F(x)(y)$ — F must take a higher type. Since for any given F we must choose a type for it, it is not possible to directly represent the context we might write as $\mathsf{Qs}\phi$, where Qs represents an *unknown* context (of quantifiers).

Instantation of unknowns in nominal algebra does not avoid capture so that $\forall[a]X$ accurately reflects our intention when we write $\forall x.\phi$ where ϕ may be instantiated in a capturing manner. Furthermore X *still* represents ϕ in $\forall[a]\forall[b]X$.

- Moving to higher orders engenders certain computational difficulties. For example unification up to $\alpha\beta$-equivalence is not decidable (although restrictions of it are in [Mil91]), while unification up to CORE is decidable [UPG04].

  Still, quotienting terms by $\alpha\beta\eta$ is convenient, as exploited by theorem-provers such as Isabelle [Pau89]. It remains to be seen whether nominal algebra or something like it can enjoy the success and usefulness enjoyed by Isabelle and similar systems.

In [GJ02, Joj04], Geuvers and Jojgov tackle the first of the above issues by extending higher-order logic with explicit meta-variables. Although the approach of their oHOL language is similar to ours, there are a number of fundamental differences:

- The default notion of instantiation of meta-variables in oHOL is still capture-avoiding; capturing instantiation can be achieved by parameterising the meta-variable, just like in the function variable approach.
- Meta-variables are equipped with pending substitutions of object-variables in oHOL. In nominal algebra meta-variables are equipped with α-renamings of object-variables. Note that we could record these substitutions by using the explicit substitutions from theory SUB.

Besides these fundamental differences, we have not investigated how well nominal algebra can serve as a basis for implementation. From that point of view, Geuvers and Jojgov are far ahead of us.

Other approaches

A host of 'cylindric' algebraic techniques exist. These embrace meta-variables and reject object-level variables, preferring to encode their expressive power in the term-formers. Examples are lambda-abstraction algebras [Sal00] for the λ-calculus and cylindric algebras [BS81, ANS01] for first-order logic. Combinators [Bar84] reject object-level variables altogether. These systems are effective for their applications, but they cannot *naturally* represent equalities with binding and meta-variables from the simple fact that there are no object-variables.

We should also mention Sun's binding algebras [Sun99]. This work is based on a functional semantics for binding, whereas we work according to the relatively newer nominal semantics which is decidedly non-functional (see Chapter 3); currently the two strands are essentially independent and it remains to see what ideas might flow between them. Fiore, Plotkin and Turi's binding algebras [FPT99] are discussed in the Related Work of Chapter 3.

2.6.2 Future Work

We have seen examples of three fundamental systems axiomatised in nominal algebra: (capture-avoiding) substitution, the λ-calculus, and first-order logic. In Chapters 4 and 6 we study the axiomatisations of substitution and first-order logic. We would like to study other systems with binding; we have in mind particular process calculi which often feature quite complex binding side-conditions and for which algebraic reasoning principles are frequently being developed [AG97, Lut02, KD02].

We are also interested in developing logics with *hierarchies* of 'increasingly meta-'variables. Since nominal algebra offers two levels of variable, why not extend this to allow an infinite hierarchy of variables, by analogy with type hierarchies in the λ-calculus [Bar84]? Work already started in this direction by extending nominal terms with a hierarchy of variables [Gab05, Gab07b, GL07]. We would like to extend these hierarchical nominal terms to a logic framework of equations.

Finally, as mentioned before, we are interested in exploring how well nominal algebra can be used as a basis for an *implementation* of an interactive proof assistent.

Chapter 3

A Semantics

3.1 Introduction

In this chapter we give a denotation to the freshness and equality judgements defined in Chapter 2. We consider a denotation in so-called *nominal sets*.

Nominal sets were introduced by Gabbay and Pitts in [GP02].[1] They have proved to be an effective model for syntax with names and binding (see for example [Pit03]). In fact, nominal sets have inspired the design of nominal terms, which form the basis of nominal algebra. For this reason, nominal sets permit a natural semantic interpretation of atoms a, abstractions $[a]t$, permutations of atoms π, and freshness $a\#t$, which are not conveniently definable on 'ordinary' sets.

Overview In Section 3.2 we give a brief overview of nominal sets. For full treatments we refer the reader to [GP02] or [Pit03] ([Pit03] contains a simplified presentation of [GP02]).

In Section 3.3 we show how we can interpret freshness and equality from nominal algebra in nominal sets. We define what constitutes a model of a theory in nominal algebra, and when freshnesses and equalities are valid.

In Section 3.4 we show that derivability of equality (in some theory) is complete with respect to its models. We also show that derivability of freshness is not complete, but that validity of freshness can be expressed in terms of derivability of certain equalities.

We conclude in Section 3.5.

[1]In [GP02], nominal sets are called FM-sets, named after the Fraenkel and Mostowski who devised a permutation model of set theory in order to prove the independence of the axiom of choice in Zermelo-Fraenkel set theory with atoms.

3.2 Nominal Sets

Recall from Definitions 2.2.3 and 2.2.4 that we write $\mathbb{A}$ for the set of all atoms, $\mathbb{P}$ for the set of all permutations, ι and $\circ$ for the identity and composition of permutations, and recall from Definition 2.4.7 that we write $\mathrm{ds}(\pi, \pi')$ for the difference set of π and π'.

Definition 3.2.1. A **$\mathbb{P}$-action** $\cdot$ on a set $\mathbb{X}$ is a function $\cdot : \mathbb{P} \times \mathbb{X} \to \mathbb{X}$, write it infix as $\pi \cdot x$, such that $\iota \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$ for all $x \in \mathbb{X}$. Say that a *finite* set of atoms A **supports** x when for any permutation π:

$$\text{if } \pi(a) = a \text{ for each } a \in A\text{, then } \pi \cdot x = x.$$

Say that x has **finite support** when there exists such a set of atoms.

A **nominal set** is a set $\mathbb{X}$ equipped with a $\mathbb{P}$-action on $\mathbb{X}$ such that each $x \in \mathbb{X}$ has finite support.

In [GP02, Proposition 3.4] it is shown that if an element $x \in \mathbb{X}$ has finite support, then there is a unique least finite set of atoms that supports x.

Definition 3.2.2. When $x \in \mathbb{X}$ has finite support, call the least set of atoms that supports x the **support** of x, and write it as $\mathrm{supp}(x)$.

Write $a\#x$ when $a \notin \mathrm{supp}(x)$, and say that a **is fresh for** x.

Lemma 3.2.3. *Basic results on nominal sets are:*

1. $\mathrm{supp}(x) = \{a \in \mathbb{A} \mid \{b \in \mathbb{A} \mid (a\ b) \cdot x \neq x\}$ *is not finite*$\}$.
2. *If* $x = y$ *then* $a\#x$ *if and only* $a\#y$.
3. *If* $a\#x$ *for every* $a \in \mathrm{ds}(\pi, \pi')$ *then* $\pi \cdot x = \pi' \cdot x$.
4. *If* $a\#x$ *then* $\pi(a)\#\pi \cdot x$.
5. *If* $x = y$ *then* $\pi \cdot x = \pi \cdot y$.

Proof. Elsewhere [GP02, Proposition 3.4] and by calculations. □

Example 3.2.4.

1. The set $\mathbb{A}$ of all atoms with action $\pi \cdot a = \pi(a)$ is a nominal set; the support of $a \in \mathbb{A}$ is $\{a\}$. Note that for $x, y \in \mathbb{A}$, $x\#y$ when $x \neq y$.
2. The powerset $\mathcal{P}(\mathbb{A}) = \{U \mid U \subseteq \mathbb{A}\}$ of $\mathbb{A}$ with action $\pi \cdot U = \{\pi \cdot u \mid u \in U\}$, is *not* a nominal set; $\{a_1, a_3, a_5, \ldots\} \in \mathcal{P}(\mathbb{A})$ does not have finite support, since for no finite set of atoms it is the case that all permutations fixing that set map $\{a_1, a_3, a_5, \ldots\}$ to itself. Note that the support of $\mathbb{A} \in \mathcal{P}(\mathbb{A})$ is $\emptyset$, so $a\#\mathbb{A}$ for any a.
3. Call $U \subseteq \mathbb{A}$ **cofinite** when $\mathbb{A}\backslash U$ is finite. The set $\mathcal{P}_{fs}(\mathbb{A})$ of finite and cofinite subsets of $\mathbb{A}$, which can be defined as $\{U \mid U \subseteq \mathbb{A},\ U \text{ finite or cofinite}\}$, with the pointwise action inherited from $\mathcal{P}(\mathbb{A})$, is a nominal set (*fs* stands for finite support); the support of $\mathbb{A}\backslash\{a\} \in \mathcal{P}_{fs}(\mathbb{A})$ is a, so $b\#\mathbb{A}\backslash\{a\}$ but not $a\#\mathbb{A}\backslash\{a\}$.

4. The empty set $\emptyset$ with the trivial action is a nominal set.
5. If $\mathbb{X}$ and $\mathbb{Y}$ are nominal sets write $\mathbb{X} \times \mathbb{Y}$ for $\{(x, y) \mid x \in \mathbb{X},\ y \in \mathbb{Y}\}$ with action $\pi \cdot (x, y) = (\pi \cdot x, \pi \cdot y)$. This is also a nominal set; the support of $(x, y) \in \mathbb{X} \times \mathbb{Y}$ is the union of the supports of x and y.
6. If $\mathbb{X}$ is a nominal set write $\mathbb{X}^n$ for $\{(x_1, \dots, x_n) \mid x_i \in \mathbb{X}, 1 \leq i \leq n\}$ with action $\pi \cdot (x_1, \dots, x_n) = (\pi \cdot x_1, \dots, \pi \cdot x_n)$. Again, this is a nominal set; the support of an element $(x_1, \dots, x_n)$ is the union of the supports of the x_i.
7. The set of infinitary λ-terms [KKSdV97] with the pointwise action is *not* a nominal set: terms might mention an infinite number of different atoms, so they do not adhere to the finite support property. This problem can be overcome by moving to FMG (Fraenkel Mostowski Generalised). This generalises the countable set of atoms to any large cardinality, and finite sets of atoms to any strictly smaller cardinality (well-orderable sets to be precise); see elsewhere [Gab07a] for further details.

We can think of $a\#x$ as an abstract notion of 'does not occur in x in any *distinguished* manner'. We say 'distinguished', because the example of $\mathcal{P}_{fs}(\mathbb{A})$ shows that $\#$ is not the same as $\notin$: for example $a \notin (\mathbb{A} \setminus \{a\})$ but *not* $a\#(\mathbb{A} \setminus \{a\})$.

Definition 3.2.5. We assume the permutation actions on the sets from Example 3.2.4 henceforth.

Definition 3.2.6. For any nominal sets $\mathbb{X}, \mathbb{Y}$, call a function $f \in \mathbb{X} \to \mathbb{Y}$ (on the underlying sets) **equivariant** when $\pi \cdot f(x) = f(\pi \cdot x)$ for any $x \in \mathbb{X}$.

Lemma 3.2.7. *For any nominal sets $\mathbb{X}, \mathbb{Y}$, equivariant function $f \in \mathbb{X} \to \mathbb{Y}$ and $x \in \mathbb{X}$, $\mathrm{supp}(f(x)) \subseteq \mathrm{supp}(x)$.*

As a corollary, $a\#x$ implies $a\#f(x)$.

Proof. By Definition 3.2.6 $\pi \cdot f(x) = f(\pi \cdot x)$, so if $\pi \cdot x = x$ then $\pi \cdot f(x) = f(x)$.

The corollary follows by Definition 3.2.2. ☐

Subsets of (the underlying set of) a nominal set will be important later when we build free term algebras.

Definition 3.2.8. $\mathcal{X} \subseteq \mathbb{X}$ inherits a pointwise action $\pi \cdot \mathcal{X} = \{\pi \cdot x \mid x \in \mathcal{X}\}$.

We will always use this action on $\mathcal{X} \subseteq \mathbb{X}$.

$a\#\mathcal{X}$ does *not* imply that $a\#x$ for every $x \in \mathcal{X}$. For example $\mathbb{A} \subseteq \mathbb{A}$ and it is a fact that $a\#\mathbb{A}$ — but $a \in \mathbb{A}$ and not $a\#a$. Furthermore $\mathcal{X} \subseteq \mathbb{X}$ does *not* imply that $\mathcal{X}$ is finitely supported. For example $\{a_1, a_2, a_3, \dots\} \subseteq \mathbb{A}$ but $\{a_1, a_3, a_5, \dots\}$ is not finitely supported. However, the finitely-supported subsets of $\mathbb{X}$ form a nominal set — they have a permutation action, and are finitely supported.

Lemma 3.2.9. *Suppose $\mathbb{X}$ is a nominal set and $\mathcal{X} \subseteq \mathbb{X}$ is finitely-supported. Then if $a_1\#\mathcal{X}, \dots, a_n\#\mathcal{X}$ there exists some $x \in \mathcal{X}$ such that $a_1\#x, \dots, a_n\#x$.*

Proof. Choose any $y \in \mathcal{X}$. Let $b_1, \ldots, b_n$ be fresh (so $b_i \# \mathcal{X}$ and $b_i \# y$ for $1 \leq i \leq n$). Then by part 3 of Lemma 3.2.3 $(b_1\ a_1) \cdots (b_n\ a_n) \cdot \mathcal{X} = \mathcal{X}$.

Write $x = (b_1\ a_1) \cdots (b_n\ a_n) \cdot y$. Then $x \in \mathcal{X}$ by Definition 3.2.8 and we conclude $a_i \# x$ for $1 \leq i \leq n$ by part 4 of Lemma 3.2.3 and the assumption $b_i \# y$. □

3.3 Interpretations, Models and Validity

We now give a semantics in nominal sets to nominal algebra theories.

Definition 3.3.1. An **interpretation of a signature** Σ, which we write as $[\![\_]\!]$, is a nominal set $\mathbb{T}$ with *equivariant* functions

- $[\![\_]\!] \in \mathbb{A} \to \mathbb{T}$ to interpret atoms;
- $[\_]\_ \in \mathbb{A} \times \mathbb{T} \to \mathbb{T}$ such that $a \# [a]x$ always, to interpret abstraction;
- $[\![\mathsf{f}]\!] \in \mathbb{T}^n \to \mathbb{T}$ for each term-former $\mathsf{f} : n$ in Σ, to interpret term-formers.

We extend the notion of interpretation to terms, where we map unknowns to elements of the nominal set $\mathbb{T}$:

Definition 3.3.2. A **valuation** ς maps unknowns X to elements $\varsigma(X) \in \mathbb{T}$. We write $[\![t]\!]_\varsigma$ for the **interpretation of a term** t under a valuation ς, inductively defined by:

$$[\![a]\!]_\varsigma = [\![a]\!] \qquad [\![\pi \cdot X]\!]_\varsigma = \pi \cdot \varsigma(X) \qquad [\![[a]t]\!]_\varsigma = [a][\![t]\!]_\varsigma$$
$$[\![\mathsf{f}(t_1, \ldots, t_n)]\!]_\varsigma = [\![\mathsf{f}]\!]([\![t_1]\!]_\varsigma, \ldots, [\![t_n]\!]_\varsigma)$$

Interpretations are equivariant.

Lemma 3.3.3. *For any* π, $\pi \cdot [\![t]\!]_\varsigma = [\![\pi \cdot t]\!]_\varsigma$.

Proof. By induction on the structure of t, using Lemma 3.2.7 for the cases of a, $[a]t$ and $\mathsf{f}(t_1, \ldots, t_n)$. □

Using the interpretations of signatures and terms, we define the notion of *validity* on judgement forms as follows:

Definition 3.3.4. For any interpretation $[\![\_]\!]$, say that:

$$[\![\Delta]\!]_\varsigma \text{ (\textbf{is valid}) when } a\#\varsigma(X) \text{ for each } a\#X \in \Delta$$
$$[\![\Delta \vdash a\#t]\!]_\varsigma \text{ when } [\![\Delta]\!]_\varsigma \text{ implies } a\#[\![t]\!]_\varsigma$$
$$[\![\Delta \vdash t = u]\!]_\varsigma \text{ when } [\![\Delta]\!]_\varsigma \text{ implies } [\![t]\!]_\varsigma = [\![u]\!]_\varsigma$$
$$[\![\Delta \vdash a\#t]\!] \text{ when } [\![\Delta \vdash a\#t]\!]_\varsigma \text{ for all valuations } \varsigma$$
$$[\![\Delta \vdash t = u]\!] \text{ when } [\![\Delta \vdash t = u]\!]_\varsigma \text{ for all valuations } \varsigma$$

Then a model of a theory is an interpretation that validates its axioms:

Definition 3.3.5. A **model** of a theory T is an interpretation $[\![\_]\!]$ of its signature such that $[\![\nabla \vdash t = u]\!]$ for all axioms $\nabla \vdash t = u$ of T.

So a model of a nominal algebra theory is just like a model of any other algebraic theory, but we must interpret permutations by permutations, and atoms, abstractions and term-formers by *equivariant* functions on the underlying sets.

Definition 3.3.6. For any theory T, define **validity with respect to** T for judgement forms as follows:

- Write $\Delta \models_{\mathsf{T}} a\#t$ when $[\![\Delta \vdash a\#t]\!]$ for all models $[\![\_]\!]$ of T.
- Write $\Delta \models_{\mathsf{T}} t = u$ when $[\![\Delta \vdash t = u]\!]$ for all models $[\![\_]\!]$ of T.

Note the T subscript in $\models_{\mathsf{T}}$, which indicates that freshness validity is not a purely syntactic affair as in freshness derivability, but also depends on the axioms of theory T. More on this in Subsection 3.4.3.

Derivability of freshness and equality is sound for the semantics:

Theorem 3.3.7 (Soundness)**.** *For any* T*,* Δ*,* a*,* t, u*:*

1. *If* $\Delta \vdash a\#t$ *then* $\Delta \models_{\mathsf{T}} a\#t$*.*
2. *If* $\Delta \vdash_{\mathsf{T}} t = u$ *then* $\Delta \models_{\mathsf{T}} t = u$*.*

Proof. Let $[\![\_]\!]$ be a model of T. We must show that if $a\#t$ (or $t = u$) is derived from Δ then $[\![\Delta]\!]_\varsigma$ implies $a\#[\![t]\!]_\varsigma$ (or $[\![t]\!]_\varsigma = [\![u]\!]_\varsigma$) for any valuation ς. We work by induction on derivations built using the rules in Figures 2.1 and 2.2.

- $(\#\mathbf{ab})$. We must show $a\#[\![b]\!]$. By Lemma 3.2.7 this follows from $a\#b$, which is a standard property of freshness (see part 1 of Example 3.2.4).
- $(\#\mathbf{X})$. By the inductive hypothesis we know $\pi^{-1}(a)\#\varsigma(X)$. By part 4 of Lemma 3.2.3 we conclude $a\#\pi\cdot\varsigma(X)$.
- $(\#[]\mathbf{a})$. $a\#[a][\![t]\!]_\varsigma$ holds by construction.
- $(\#[]\mathbf{b})$. $a\#[\![t]\!]_\varsigma$ implies $a\#[b][\![t]\!]_\varsigma$, by Lemma 3.2.7.
- $(\#\mathsf{f})$. If $a\#[\![t_i]\!]_\varsigma$ for $1 \leq i \leq n$ then $a\#[\![\mathsf{f}]\!]([\![t_1]\!]_\varsigma, \dots, [\![t_n]\!]_\varsigma)$ follows using Lemma 3.2.7.
- $(\mathbf{refl})$, $(\mathbf{symm})$, $(\mathbf{tran})$, $(\mathbf{cong}[])$, $(\mathbf{congf})$. By properties of equality.
- $(\mathbf{perm})$. We know that $a\#[\![t]\!]_\varsigma$ and $b\#[\![t]\!]_\varsigma$ imply $(a\ b)\cdot[\![t]\!]_\varsigma = [\![t]\!]_\varsigma$ by part 3 of Lemma 3.2.3. We conclude $[\![(a\ b)\cdot t]\!]_\varsigma = [\![t]\!]_\varsigma$ by Lemma 3.3.3.
- $(\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}})$. Suppose $[\![\nabla^\pi\sigma]\!]_\varsigma$ for any ς. Then $\pi(a)\#\sigma(X)\varsigma$ holds for all $a\#X \in \nabla$. By part 4 of Lemma 3.2.3 also $a\#\pi^{-1}\cdot\sigma(X)\varsigma$ for all $a\#X \in \nabla$. Let ς' be defined as $\varsigma'(X) = \pi^{-1}\cdot\sigma(X)\varsigma$ for any X. Then $a\#\varsigma'(X)$ for all $a\#X \in \nabla$, so $[\![\nabla]\!]_{\varsigma'}$ holds. But then also $[\![t]\!]_{\varsigma'} = [\![u]\!]_{\varsigma'}$ since $\nabla \vdash t = u$ is an axiom of T. By part 5 of Lemma 3.2.3 $\pi\cdot[\![t]\!]_{\varsigma'} = \pi\cdot[\![u]\!]_{\varsigma'}$, and by Lemma 3.3.3 we obtain $[\![\pi\cdot t]\!]_{\varsigma'} = [\![\pi\cdot u]\!]_{\varsigma'}$. Now by a straightforward induction on syntax we can verify that $[\![\pi\cdot t]\!]_{\varsigma'} = [\![t^\pi\sigma]\!]_\varsigma$ and $[\![\pi\cdot u]\!]_{\varsigma'} = [\![u^\pi\sigma]\!]_\varsigma$, and we conclude $[\![t^\pi\sigma]\!]_\varsigma = [\![u^\pi\sigma]\!]_\varsigma$.

- **(fr)**. So suppose $\Delta \vdash_{\mathsf{T}} t = u$ is derived from $\Delta, a\#X_1, \ldots, a\#X_n \vdash_{\mathsf{T}} t = u$, where $a \notin t, u, \Delta$. By ZFA equivariance (Theorem A.2.5) then also
$$\Delta, a'\#X_1, \ldots, a'\#X_n \vdash_{\mathsf{T}} t = u$$
for all other a' not occurring in Δ, t or u. We also retain the inductive hypothesis for $\Delta, a'\#X_1, \ldots, a'\#X_n \vdash_{\mathsf{T}} t = u$ by ZFA equivariance.
We must show that $[\![\Delta \vdash_{\mathsf{T}} t = u]\!]_{\varsigma}$ for any ς. Now pick an $a' \notin \Delta, t, u$ such that $a'\#\varsigma(X_i)$ for $1 \leq i \leq n$. Then by the inductive hypothesis we obtain $[\![\Delta, a'\#X_1, \ldots, a'\#X_n \vdash_{\mathsf{T}} t = u]\!]_{\varsigma}$. But this is equivalent to $[\![\Delta \vdash_{\mathsf{T}} t = u]\!]_{\varsigma}$, since $[\![a'\#X_1, \ldots, a'\#X_n]\!]_{\varsigma}$. The result follows.

□

3.4 Completeness

In this section we show that derivability of equality is complete with respect to the semantics. We also show that completeness does not hold for derivability of freshness, and that this is precisely what we want. For this we use the notion of a free term model, which we introduce first.

3.4.1 Free Term Models

Definition 3.4.1. In this subsection fix a signature Σ, a theory $\mathsf{T} = (\Sigma, Ax)$, and fix a set of term-formers D disjoint from Σ.

The usual technique to obtain models for a theory T is to add constant symbols to the language (to ensure a supply of 'arbitrary elements') and quotient by provable equality. But in nominal algebra constants have empty support; if d has arity 0 then $\vdash a\#\mathsf{d}$ is derivable for any a. Adding constants only ensures a supply of elements with empty support.

To reflect in syntax that an element of a nominal set can have support, we use n-ary term-formers d applied to n distinct atoms. This idea goes back to [Gab06]. We now give the construction in detail.

Definition 3.4.2. Let **free terms** be inductively generated by the following grammar:
$$g \quad ::= \quad a \mid [a]g \mid \mathsf{f}(g_1, \ldots, g_n) \mid \mathsf{d}(a_1, \ldots, a_n)$$
Here $\mathsf{f} : n$ ranges over elements of Σ, and $\mathsf{d} : n$ ranges over elements of D.

Recall the notation $\pi \cdot t$ for the object-level permutation action on a term t from Definition 2.3.1.

Lemma 3.4.3. *The set of free terms with action $\pi \cdot g$ is a nominal set; the support of g is $\{a \in \mathbb{A} \mid a \notin g\}$.*
As a corollary, $a \notin g$ if and only if $a\#g$.

Proof. $\mathrm{supp}(g) = \{a \in \mathbb{A} \mid a \notin g\}$ follows by an induction on the structure of g, using part 1 of Lemma 3.2.3. The corollary follows by Definition 3.2.2. $\square$

We need a technical lemma:

Lemma 3.4.4. *For any free term g, if $a \notin g$ then $\vdash a\#g$.*

Proof. By an induction on syntax using the rules in Figure 2.1. $\square$

Definition 3.4.5. Write (**cong**d) for an instance of the (**cong**f) rule when $\mathsf{f} \in \mathsf{D}$. Write $[g]_{\mathsf{T}}$ for the set of free terms g' such that a derivation of $\vdash_{\mathsf{T}} g = g'$ exists that does not mention (**cong**d) for any $\mathsf{d} \in \mathsf{D}$.

Let the **set of free terms up to** T be the set $\{[g]_{\mathsf{T}} \mid g \text{ a free term}\}$.

Lemma 3.4.6. *The set of free terms up to T with action $\pi \cdot [g]_{\mathsf{T}} = [\pi \cdot g]_{\mathsf{T}}$ is a nominal set; the set $[g]_{\mathsf{T}}$ is supported by $\{a \in \mathbb{A} \mid \nvdash a\#g\}$.*

As a corollary, if $\vdash a\#g$ then $a\#[g]_{\mathsf{T}}$.

Proof. Let A be the set $\{a \mid \nvdash a\#g\}$; this is finite by Lemma 3.4.4. It suffices to show that A supports $[g]_{\mathsf{T}}$.

So let π be a permutation such that $\pi(a) = a$ for all $a \in A$. We must show $\pi \cdot [g]_{\mathsf{T}} = [g]_{\mathsf{T}}$. By assumption $\pi \cdot [g]_{\mathsf{T}} = [\pi \cdot g]_{\mathsf{T}}$, so it suffices to show $[\pi \cdot g]_{\mathsf{T}} = [g]_{\mathsf{T}}$. By definition of $[\_]_{\mathsf{T}}$, this is when $\vdash_{\mathsf{T}} \pi \cdot g = g$. By Lemma 2.4.8, this follows from $\vdash \mathrm{ds}(\pi, \iota)\#g$. But this follows from the assumption on π since $\mathrm{ds}(\pi, \iota)$ and A are disjoint.

The corollary follows by Definition 3.2.2. $\square$

The following technical lemma will be useful later:

Lemma 3.4.7. $a_1\#[g]_{\mathsf{T}}, \dots, a_n\#[g]_{\mathsf{T}}$ *if and only if there exists some $g' \in [g]_{\mathsf{T}}$ such that $\vdash a_1\#g', \dots, \vdash a_n\#g'$ are all derivable.*

Proof. For the left-right implication we use Lemma 3.2.9 to pick some $g' \in [g]_{\mathsf{T}}$ such that $a_1\#g', \dots, a_n\#g'$. We conclude $\vdash a_1\#g', \dots, \vdash a_n\#g'$ by Lemmas 3.4.3 and 3.4.4.

For the right-to-left implication, we observe that if $g' \in [g]_{\mathsf{T}}$ then $[g]_{\mathsf{T}} = [g']_{\mathsf{T}}$. The result follows by Lemma 3.4.6. $\square$

The following example shows why Lemma 3.4.7 is non-trivial.

Example 3.4.8. Consider a theory ATOM with one axiom $\vdash a = b$. It is easy to verify that $a\#[a]_{\mathsf{ATOM}}$ (since $[a]_{\mathsf{ATOM}} = \mathbb{A}$) but $a\#a$ is not derivable. Of course $a = b$ and $a\#b$ *are* derivable. Similarly in LAM it is a fact that $a\#[(\lambda[a]b)a]_{\mathsf{LAM}}$ but $a\#(\lambda[a]b)a$ is not derivable; of course $(\lambda[a]b)a = b$ and $a\#b$ are derivable.

Definition 3.4.9. We construct the **free term model** $[\![\_]\!]^{\mathcal{T}}$ of T over D as follows:

- Take as underlying nominal set the set of free terms up to T with action $\pi \cdot [g]_{\mathsf{T}} = [\pi \cdot g]_{\mathsf{T}}$.

- Take as interpretations of the equivariant functions:
  - $[\![a]\!]^{\mathcal{T}} = [a]_{\mathsf{T}}$.
  - $[a]x = [[a]g]_{\mathsf{T}}$ for some $g \in x$.
  - $[\![\mathsf{f}]\!]^{\mathcal{T}}(x_1, \dots, x_n) = [\mathsf{f}(g_1, \dots, g_n)]_{\mathsf{T}}$ for some $g_1 \in x_1, \dots, g_n \in x_n$ (for each term-former $\mathsf{f} : n$ in Σ).

It is usual to build free term models by quotienting by provable equality; we exclude (**congd**) to avoid the following degenerate case: if we allow (**congd**) and T contains an axiom such as $\vdash a = b$, then $\mathrm{supp}[\mathsf{d}(a_1, \dots, a_n)]_{\mathsf{T}} = \emptyset$. This is not the behaviour we want. Similarly our syntax of free terms does not allow terms of the form $\mathsf{d}(g_1, \dots, g_n)$ in general. The only purpose of $[\mathsf{d}(a_1, \dots, a_n)]_{\mathsf{T}}$ is to 'be an unknown element with support $a_1, \dots, a_n$'.

Lemma 3.4.10. *$[\![\_]\!]^{\mathcal{T}}$ is an interpretation of Σ.*

Proof. The underlying set of $[\![\_]\!]^{\mathcal{T}}$ is a nominal set by Lemma 3.4.6. For the interpretation functions we must show that they are well-defined (for $[\_]\_$ and $[\![\mathsf{f}]\!]^{\mathcal{T}}$ the choices of $g \in x$ and $g_1 \in x_1, \dots, g_n \in x_n$ do not matter) and equivariant. This is easy using the definitions of $[\_]_{\mathsf{T}}$ and the permutation action. Finally, for the $[\_]\_$ function, we must show that $a\#[a]x$ holds, i.e. $a\#[[a]g]_{\mathsf{T}}$, which we will do now.

Choose b fresh (so $b \notin g$ and $b\#[[a]g]_{\mathsf{T}}$). Since $b\#[[a]g]_{\mathsf{T}}$, also $a\#(b\ a) \cdot [[a]g]_{\mathsf{T}}$ by part 4 of Lemma 3.2.3. By definition of the permutation action, then also $a\#[[b](b\ a) \cdot g]_{\mathsf{T}}$. Now since $\vdash a\#[b](b\ a) \cdot g$ and $\vdash b\#[b](b\ a) \cdot g$ by Lemma 3.4.4 and the derivation rules of Figure 2.1, we know $\vdash_{\mathsf{T}} [b](b\ a) \cdot g = [a]g$ by (**perm**). Then $[[b](b\ a) \cdot g]_{\mathsf{T}} = [[a]g]_{\mathsf{T}}$, and we obtain $a\#[[a]g]_{\mathsf{T}}$ as required. □

Theorem 3.4.11. *$[\![\_]\!]^{\mathcal{T}}$ is a model of T.*

Proof. Suppose $\nabla \vdash t = u$ is an axiom of T. Suppose that ς is a valuation to the underlying set of $[\![\_]\!]^{\mathcal{T}}$ and that $a\#\varsigma(X)$ for every $a\#X \in \nabla$. We must show that $[\![t]\!]_\varsigma = [\![u]\!]_\varsigma$.

Let $\mathcal{X}$ be the set of all unknowns mentioned in ∇, t, or u. By Lemma 3.4.7, for every $X \in \mathcal{X}$ there is an element $g_X \in \varsigma(X)$ such that $\vdash a\#g_X$ for every $a\#X \in \nabla$. Let σ be the substitution such that $\sigma(X) = g_X$ when $X \in \mathcal{X}$ and $\sigma(X) = X$ when $X \notin \mathcal{X}$. Then $\vdash \nabla\sigma$, and $\vdash_{\mathsf{T}} t\sigma = u\sigma$ by ($\mathbf{ax}_{\nabla \vdash \mathbf{t}=\mathbf{u}}$). Since this derivation does not mention (**congd**), we know $[t\sigma]_{\mathsf{T}} = [u\sigma]_{\mathsf{T}}$ by Definition 3.4.5. By an induction on syntax we verify that $[t\sigma]_{\mathsf{T}} = [\![t]\!]_\varsigma$ and $[u\sigma]_{\mathsf{T}} = [\![u]\!]_\varsigma$, and the result follows. □

3.4.2 Completeness for Equality Derivations

Definition 3.4.12. For this subsection, fix a signature Σ, a theory $\mathsf{T} = (\Sigma, Ax)$, and terms t, u and a freshness context Δ in signature Σ.

We will show that derivability of $\Delta \vdash_{\mathsf{T}} t = u$ is complete. That is, we will prove:

Theorem 3.4.13 (Completeness)**.** *If $\Delta \models_{\mathsf{T}} t = u$ then $\Delta \vdash_{\mathsf{T}} t = u$.*

The proof takes up the rest of this subsection.

We shall consider a specific free term model and a specific valuation to this model that preserves sufficient information to allow us to reconstruct a derivation of $\Delta \vdash_{\mathsf{T}} t = u$.

Definition 3.4.14. Let $\mathcal{X}$ be the unknowns mentioned in Δ, t, u, and let $\mathcal{A}$ be the atoms mentioned in Δ, t, u. For each $X \in \mathcal{X}$:

- let $a_{X1}, \dots, a_{Xk_X}$ be the atoms in $\mathcal{A}$ (in some arbitary but fixed order) such that $a_{Xi}\#X \notin \Delta$;
- let $\mathsf{d}_X : k_X$ be a term-former.

For each unknown $X \notin \mathcal{X}$, let $\mathsf{d}_X : 0$ be a term-former.

Let D be the set of all d_X's (so $\mathsf{d}_X \in \mathsf{D}$ for each X).

Definition 3.4.15. Now we consider the free term model $[\![\_]\!]^{\mathcal{T}}$ of T over D, and the following valuation ς to the underlying set:

$$\begin{array}{ll} \varsigma(X) = [\mathsf{d}_X(a_{X1}, \dots, a_{Xk_X})]_{\mathsf{T}} & (X \in \mathcal{X}) \\ \varsigma(X) = [\mathsf{d}_X()]_{\mathsf{T}} & (X \notin \mathcal{X}) \end{array}$$

Lemma 3.4.16. $[\![\Delta]\!]_\varsigma$ *holds.*

Proof. Suppose $a\#X \in \Delta$. We must show that $a\#\varsigma(X)$. By construction $X \in \mathcal{X}$ so $\varsigma(X) = [\mathsf{d}_X(a_{X1}, \dots, a_{Xk_X})]_{\mathsf{T}}$. But also $a \notin \{a_{X1}, \dots, a_{Xk_X}\}$ by construction so $a \notin \mathsf{d}_X(a_{X1}, \dots, a_{Xk_X})$. The result follows by Lemmas 3.4.4 and 3.4.6. □

We define a substitution to talk about a specific element each $\varsigma(X)$.

Definition 3.4.17. Let σ be the following substitution:

$$\begin{array}{ll} \sigma(X) = \mathsf{d}_X(a_{X1}, \dots, a_{Xk_X}) & (X \in \mathcal{X}) \\ \sigma(X) = X & (X \notin \mathcal{X}) \end{array}$$

Lemma 3.4.18. $[t\sigma]_{\mathsf{T}} = [\![t]\!]_\varsigma$ *and* $[u\sigma]_{\mathsf{T}} = [\![u]\!]_\varsigma$.

Proof. By an induction on syntax, using the fact that t and u only mention unknowns from $\mathcal{X}$. □

Definition 3.4.19. Let Π be a derivation of $\vdash_{\mathsf{T}} t\sigma = u\sigma$ without using (**cong**d). By Corollary 2.4.11 we assume that Π does not contain unknowns or instances of (**#X**) and (**fr**).

Let $\mathcal{A}^+$ be $\mathcal{A}$ extended with:

- atoms mentioned anywhere in Π (that were not already in $\mathcal{A}$);
- a set $\mathcal{B} = \{b_{Xi} \mid X, i \text{ such that } a_{Xi} \in \mathcal{A}\}$ of fresh atoms in bijection with $\mathcal{A}$;
- one fresh atom c (so c does not occur in $\mathcal{A}$, Π, or $\mathcal{B}$).

Let Δ^+ be Δ extended with freshness assumptions $a'\#X$ for every $X \in \mathcal{X}$ and every $a' \in \mathcal{A}^+ \setminus \mathcal{A}$.

Definition 3.4.20. For the rest of this subsection let g and h range over free terms in $\Sigma \cup \mathsf{D}$ that mention only atoms from $\mathcal{A}^+ \setminus (\mathcal{B} \cup \{c\})$. Define an **inverse mapping** $\text{-}^{\text{-}1}$ from such free terms to terms in Σ inductively as follows:

$$\begin{array}{ll} a^{\text{-}1} \equiv a \quad ([a](g))^{\text{-}1} \equiv [a]g^{\text{-}1} \quad \mathsf{f}(g_1,\dots,g_n)^{\text{-}1} \equiv \mathsf{f}(g_1^{\text{-}1},\dots,g_n^{\text{-}1}) & \\ \mathsf{d}_X(a'_{X1},\dots,a'_{Xk_X})^{\text{-}1} \equiv \pi_X(a'_{X1},\dots,a'_{Xk_X}) \cdot X & (X \in \mathcal{X}) \\ \mathsf{d}_X()^{\text{-}1} \equiv c & (X \notin \mathcal{X}) \end{array}$$

Here we use the abbreviation

$$\pi_X(a'_{X1},\dots,a'_{Xk_X}) \text{ for } (a'_{X1}\ b_{X1}) \cdots (a'_{Xk_X}\ b_{Xk_X})(b_{X1}\ a_{X1}) \cdots (b_{Xk_X}\ a_{Xk_X}).$$

We extend the notation $\text{-}^{\text{-}1}$ to freshnesses and freshness contexts by acting on the terms they mention.

The inverse mapping is equivariant (for the terms we care about):

Lemma 3.4.21. $\Delta^+ \vdash_{\mathsf{CORE}} (\pi \cdot g)^{\text{-}1} = \pi \cdot g^{\text{-}1}$ *when π mentions only atoms from $\mathcal{A}^+ \setminus (\mathcal{B} \cup \{c\})$.*

Proof. By induction on the structure of g. The only non-trivial case is when $g \equiv \mathsf{d}_X(a'_{X1},\dots,a'_{Xk_X})$ with $X \in \mathcal{X}$. Then we must show

$$\Delta^+ \vdash_{\mathsf{CORE}} \pi_X(\pi(a'_{X1}),\dots,\pi(a'_{Xk_X})) \cdot X = (\pi \circ \pi_X(a'_{X1},\dots,a'_{Xk_X})) \cdot X.$$

By the syntactic criteria for CORE derivability of Corollary 2.5.4 it suffices to show

$$\Delta^+ \vdash \mathrm{ds}(\pi_X(\pi(a'_{X1}),\dots,\pi(a'_{Xk_X})), (\pi \circ \pi_X(a'_{X1},\dots,a'_{Xk_X})))\#X.$$

This follows by a case analysis on the atoms in the difference set, using the fact that π does not mention any of the b_{Xi}. □

Lemma 3.4.22. $\Delta^+ \vdash_{\mathsf{CORE}} t\sigma^{\text{-}1} = t$ *and* $\Delta^+ \vdash_{\mathsf{CORE}} u\sigma^{\text{-}1} = u$.

Proof. We show $\Delta^+ \vdash_{\mathsf{CORE}} v\sigma^{\text{-}1} = v$ for each subterm v of t and u. We do this by induction on the structure of v. The proof of the case of $v \equiv \pi \cdot X$ is analogous to the $\mathsf{d}_X(a'_{X1},\dots,a'_{Xk_X})$ case in the proof of Lemma 3.4.21. □

Lemma 3.4.23. *If* $\vdash_{\mathsf{T}} t\sigma = u\sigma$ *without using* (**congd**) *then* $\Delta \vdash_{\mathsf{T}} t = u$.

Proof. Suppose we could transform the derivation Π of $\vdash_{\mathsf{T}} t\sigma = u\sigma$ into a derivation of $\Delta^+ \vdash_{\mathsf{T}} t\sigma^{\text{-}1} = u\sigma^{\text{-}1}$. Given that, the result follows because by Lemma 3.4.22 we deduce $\Delta^+ \vdash_{\mathsf{T}} t = u$ and we obtain $\Delta \vdash_{\mathsf{T}} t = u$ as required by strengthening (Theorem 2.4.14).

Our transformation of $\vdash_{\mathsf{T}} t\sigma = u\sigma$ into $\Delta^+ \vdash_{\mathsf{T}} t\sigma^{\text{-}1} = u\sigma^{\text{-}1}$ is inductive on Π. Suppose Π concludes with an instance of …

- (**#ab**), (**#[a]**), (**#[b]**), (**refl**), (**symm**), (**tran**) or (**cong[]**). Then the result trivially follows by an instance of the same rule, possibly using the inductive hypothesis.
- (**#X**) or (**fr**). This is impossible by assumption (see Definition 3.4.19).
- (**#f**). There are three cases to consider:
  - The case of $\vdash a\#\mathsf{f}(g_1,\ldots,g_n)$ for $\mathsf{f}\in\Sigma$.
    Then by assumption $\vdash a\#g_i$ for $1\leq i\leq n$, and $\Delta^+\vdash a\#g_i^{-1}$ by the inductive hypothesis. We conclude $\Delta^+\vdash a\#\mathsf{f}(g_1^{-1},\ldots,g_n^{-1})$ using (**#f**).
  - The case of $\vdash a\#\mathsf{d}_X(a'_{X1},\ldots,a'_{Xk_X})$ for $\mathsf{d}_X\in\mathsf{D}$ and $X\in\mathcal{X}$.
    Then by assumption $\vdash a\#a'_{Xi}$ for $1\leq i\leq k_X$, and we obtain $a\neq a'_{Xi}$ by (**#ab**). We must show $\Delta^+\vdash a\#\pi_X(a'_{X1},\ldots,a'_{Xk_X})\cdot X$. By (**#X**), this follows from $\Delta^+\vdash\pi_X(a'_{X1},\ldots,a'_{Xk_X})^{-1}(a)\#X$. Since $a\neq a'_{Xi}$ and also $a\neq b_{Xi}$ for all i, we have
    $$\pi_X(a'_{X1},\ldots,a'_{Xk_X})^{-1}(a)=(b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})(a).$$
    We proceed by a case distinction on a:
    - If $a=a_{Xi}$ for some i, then $(b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})(a)=b_{Xi}$, and the result follows since $b_{Xi}\#X\in\Delta^+$ by construction.
    - If $a\neq a_{Xi}$ for all i, then $(b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})(a)=a$ since also $a\neq b_{Xj}$ for any j. Then by construction $a\#X\in\Delta$ since the a_{Xi} are the only atoms in $\mathcal{A}$ for which $a_{Xi}\#X\notin\Delta$. The result follows.
  - The case of $\vdash a\#\mathsf{d}_X()$ for $\mathsf{d}_X\in\mathsf{D}$ and $X\notin\mathcal{X}$.
    It is immediate by (**#ab**) that $\vdash a\#c$.
- (**congf**). We consider two cases:
  - The case of $\mathsf{f}\in\Sigma$ follows using the inductive hypothesis.
  - The case of $\mathsf{d}\in\mathsf{D}$ is impossible, since we assumed that Π does not mention (**congd**).
- (**perm**). By the inductive hypothesis we have $\Delta^+\vdash a\#g^{-1}$ and $\Delta^+\vdash b\#g^{-1}$. Then $\Delta^+\vdash_{\mathsf{T}}(a\ b)\cdot g^{-1}=g^{-1}$ by (**perm**). Using Lemma 3.4.21, we conclude $\Delta^+\vdash_{\mathsf{T}}((a\ b)\cdot g)^{-1}=g^{-1}$.
- (**ax$_{\nabla\vdash v=w}$**). Then $\vdash\nabla^\pi\tau$ and $\vdash_{\mathsf{T}} v^\pi\tau=w^\pi\tau$ for some permutation π and substitution τ such that $\nabla\tau$, $v\tau$ and $w\tau$ do not mention any unknowns. We must show $\Delta^+\vdash_{\mathsf{T}}(v^\pi\tau)^{-1}=(w^\pi\tau)^{-1}$.
  Now let τ' be the substitution such that $\tau'(X)=\tau(X)^{-1}$ when $\tau(X)\neq X$ and $\tau'(X)=X$ when $\tau(X)=X$. Then $(v^\pi\tau)^{-1}\equiv v^\pi\tau'$, $(w^\pi\tau)^{-1}\equiv w^\pi\tau'$ and $(\nabla^\pi\tau)^{-1}\equiv\nabla^\pi\tau'$, so it suffices to show $\Delta^+\vdash_{\mathsf{T}} v^\pi\tau'=w^\pi\tau'$. By (**ax$_{\nabla\vdash v=w}$**), this follows from $\Delta^+\vdash\nabla^\pi\tau'$, i.e. $\Delta^+\vdash(\nabla^\pi\tau)^{-1}$. By the inductive hypothesis, this follows from the assumption $\vdash\nabla^\pi\tau$.

□

We are now ready for the main result of this subsection:

Proof of Theorem 3.4.13. Suppose $\Delta \models_{\mathsf{T}} t = u$, so $[\![\Delta \vdash t = u]\!]_{\varsigma}$ for the free term model $[\![\_]\!]^{\mathcal{T}}$ and valuation ς constructed above. Now $[\![\Delta]\!]_{\varsigma}$ by Lemma 3.4.16 so $[\![t]\!]_{\varsigma} = [\![u]\!]_{\varsigma}$. By Lemma 3.4.18 $[t\sigma]_{\mathsf{T}} = [\![t]\!]_{\varsigma}$ and $[u\sigma]_{\mathsf{T}} = [\![u]\!]_{\varsigma}$. Therefore by construction $\vdash_{\mathsf{T}} t\sigma = u\sigma$ without using (**congd**). It follows by Lemma 3.4.23 that $\Delta \vdash_{\mathsf{T}} t = u$. □

3.4.3 The Status of Freshness Derivations

Definition 3.3.6 states that validity of freshness depends on the axioms of a theory. Derivability does not depend on axioms, it only inspects syntax. For this reason, we do not have completeness for freshnesses. That is, $\Delta \models_{\mathsf{T}} b\#t$ does *not* imply $\Delta \vdash b\#t$ necessarily.

For examples of incompleteness of freshness, we recall Example 3.4.8. We showed that in theory ATOM with one axiom $\vdash a = b$, $a\#[\![a]\!]^{\mathcal{T}}_{\varsigma}$ is valid for the term model $[\![\_]\!]^{\mathcal{T}}$ (and any valuation ς). It is not hard to verify that this property holds for *all* models $[\![\_]\!]$ of ATOM. So we have $\models_{\mathsf{ATOM}} a\#a$, but also $\nvdash a\#a$. Similarly, we can show that $[\![(\lambda[a]b)a]\!]_{\varsigma} = [\![b]\!]_{\varsigma}$ for any model $[\![\_]\!]$ of theory LAM, so $\models_{\mathsf{LAM}} a\#(\lambda[a]b)a$, but also $\nvdash a\#(\lambda[a]b)a$.

To understand why this is desirable we must draw a distinction between the intension and the extension of a term.

'$\vdash a\#(\lambda[a]b)a$' has the status of '$x \notin fv((\lambda x.y)x)$'; both are false. Yet $(\lambda x.y)x$ is β-convertible to y and $x \notin fv(y)$ holds. A freshness judgement $\Delta \vdash a\#t$ is an intensional judgement on concrete syntax.[2] All the capture-avoidance side-conditions we know of are in accordance with the slogan 'ϵ away from informal practice', this is what $\Delta \vdash a\#t$ models.

Nominal sets is unusual amongst semantics in that it has an extensional *semantic* notion of freshness $a\#x$. In fact, semantic freshness is hiding in nominal algebra in the theory of *equality*.

Theorem 3.4.24. *Suppose $\{X_1, \ldots, X_n\}$ and $\{a_1, \ldots, a_m\}$ are the unknowns and atoms mentioned in Δ and t, and suppose that $b \notin \{a_1, \ldots, a_m\}$. Then:*

$$\Delta \models_{\mathsf{T}} a\#t \quad \textit{if and only if} \quad \Delta, b\#X_1, \ldots, b\#X_n \vdash_{\mathsf{T}} (b\ a) \cdot t = t.$$

Proof. Choose a model $[\![\_]\!]$ and any valuation ς such that $b\#\varsigma(X_1), \ldots, b\#\varsigma(X_n)$; it follows by an induction on syntax that $b\#[\![t]\!]_{\varsigma}$. It is a fact that $a\#[\![t]\!]_{\varsigma}$ if and only if $(b\ a) \cdot [\![t]\!]_{\varsigma} = [\![t]\!]_{\varsigma}$ (see [GP02] for details). By Lemma 3.3.3, the last part is equivalent to $[\![(b\ a) \cdot t]\!]_{\varsigma} = [\![t]\!]_{\varsigma}$.

Now suppose $\Delta \models_{\mathsf{T}} a\#t$. By definition $a\#[\![t]\!]_{\varsigma}$ for any $[\![\_]\!]$ and ς such that $[\![\Delta]\!]_{\varsigma}$. By the arguments above $[\![(b\ a) \cdot t]\!]_{\varsigma} = [\![t]\!]_{\varsigma}$ if $b\#\varsigma(X_1), \ldots, b\#\varsigma(X_n)$. By completeness Theorem 3.4.13 it follows that $\Delta, b\#X_1, \ldots, b\#X_n \vdash_{\mathsf{T}} (b\ a) \cdot t = t$. The reverse implication is similar. □

[2]The reader familiar with a theorem-prover such as Isabelle [Pau89] might like to imagine that # maps to *Prop* and = maps to *o*.

Theorem 3.4.24 tells us for instance that $\models_{\mathsf{ATOM}} a\#a$ if and only if $\vdash_{\mathsf{ATOM}} b = a$ (which follows by the axiom $\vdash a = b$), and that $\models_{\mathsf{LAM}} a\#(\lambda[a]b)a$ if and only if $\vdash_{\mathsf{LAM}} (\lambda[a]b)a = (\lambda[c]b)c$ (which follows since both sides are derivably equal to b).

So semantic freshness $\Delta \models_{\mathsf{T}} a\#t$ can be expressed as an equality axiom. Any undecidability or algorithmic complexity is isolated in the equality judgement form.

3.5 Conclusions

We have given a semantics to nominal algebra in nominal sets. In this semantics, object-level variables are first-class entities in the denotation; so an atom a represents an object-level variable symbol in the syntax *and* in the semantics.

3.5.1 Related Work

To put things into perspective we discuss some related work.

Functions

Nominal sets provide a model of α-conversion that is pretty close to informal practice. Semantic notions based on functions are models of α-conversion *and* substitution of terms for variables. These are less close to informal practice because they do not have the ability to manipulate names of bound variables *explicitly.* See [GP02, Subsection 1.1] and [Pit03, Section 9] for excellent discussions on this.

However, in addition to α-conversion, we often do need substitution behaviour on nominal sets. We can do this by imposing *axioms* such as theory SUB from Example 2.2.17. If we can substitute for an atom a then it has the flavour of a variable 'ranging over' denotational elements; so SUB is a theory of 'variables in denotation'. A detailed account of the proof-theory of SUB is in Chapter 4.

Binding algebras

Just like us, Fiore, Plotkin and Turi investigate a general framework for binding [FPT99]. They use categories of presheaves, whereas we use nominal sets. Categories of presheaves do not have a notion of "least supporting set" like nominal sets do [GP02, Related Work]. So in its current form, the freshness judgement of $a\#x$ cannot be expressed in their framework. For this reason, it is not clear how an easy and direct connection can be made between the two frameworks.

Freshness

Finally we briefly return to our discussion in the Conclusions of Chapter 2 on the difference in freshness derivability between nominal algebra on the one hand, and nominal logic [Pit03] and nominal equational logic [CP07] on the other hand. Using terminology from Subsection 3.4.3, nominal algebra uses *syntactic* freshness while nominal logic and nominal equational logic use *semantic* freshness. For this reason, derivability of freshness in nominal equational logic is complete with respect to its semantics in nominal sets [CP07, Theorem 10.10], while derivability of freshness in nominal algebra is not. However, nominal algebra is capable of expressing semantic freshness using the notion of equality (Theorem 3.4.24).

Chapter 4

Capture-Avoiding Substitution

4.1 Introduction

Substitution is intuitively the operation $v[x \mapsto t]$ meaning:

"Replace the variable x by t in v."

Can we give an algebraic characterisation of the properties of $v[x \mapsto t]$ *independently* of what v and t are (λ-terms, formulae of a logic, terms of some process calculus, or any mixture or variation thereof)?

Consider by way of analogy the notion of 'a field'. This has an algebraic characterisation which tells us what properties 'a field' must have, independently of *which* field it is, or *how* it may be implemented (if we are programming). This is useful; for example the definition of 'vector space' is parametric over fields, and this step requires a characterisation of what fields are [BS81].

When we begin to axiomatise substitution, unusual difficulties present themselves. Consider the following informally expressed candidate property of substitution:

$$v[x \mapsto t][y \mapsto u] = v[y \mapsto u][x \mapsto t[y \mapsto u]] \qquad \text{if } x \notin \mathit{fv}(u)$$

This is *not* algebraic, because of the side-condition $x \notin \mathit{fv}(u)$ (recall that $\mathit{fv}(u)$ represents the free variables of u, which is a property of the syntax of u).

In this chapter we use the framework of nominal algebra from Chapter 2 to propose a solution to this problem. We consider theory SUB from Example 2.2.17, and we will show that the axiomatisation satisfies the following properties:

- The treatment is *general* in the sense that it is independent of the choice of term-formers.
- The axioms are *natural* in the sense that they are close to informal practice.

- It is *decidable* whether two terms are equal.
- The axioms are *sound and complete* for a canonical model.

Overview In Section 4.2 we provide a simple sort system to keep terms well-formed. We recall the definition of SUB in this sorted setting and indicate why the axiomatisation is both *general* and *natural.*

To show that the axiomatisation is *decidable* and *sound and complete* takes up the rest of the chapter. Section 4.3 creates a concrete model out of syntax and defines a standard capture-avoiding substitution action on the model — this is what the axioms need to express. Section 4.4 defines a relatively weak subset of SUB which we call SIMP (for 'simple'); this subset is *sound* for the concrete model, but *not complete.* Section 4.5 proves useful properties of this simple axiomatisation; here we make heavy use of techniques from (nominal) rewriting. Section 4.6 explores the stronger axioms of SUB by relating them to the axioms of SIMP. This enables us to show *decidability* (Subsection 4.6.2) and *completeness* with respect to the concrete model from Section 4.3 (Subsection 4.6.3). Finally, Subsection 4.6.4 shows how the general axiomatisation of SUB relates to a more concrete axiomatisation.

4.2 A Sort System

Recall from Example 2.2.17 that sub is a binary term-former for explicit substitution, and that we usually write $t[a \mapsto u]$ for $\mathsf{sub}([a]t, u)$. Using the sortless system of Chapter 2 it is possible to write 'silly' terms like $\mathsf{sub}(a, t)$ and $t[a \mapsto \mathsf{sub}(u, v)]$. Looking at the axioms of theory SUB from Example 2.2.17, we can see that in the term $\mathsf{sub}(a, t)$, t will never replace anything in a since a is not an abstraction. Also in $t[a \mapsto \mathsf{sub}(u, v)]$, free occurrences of a in t will be replaced by the term $\mathsf{sub}(u, v)$, which intuitively lives at a different level.

To exclude such terms, we provide a simple sort system. The sort system we use here is tailored to our application of axiomatising substitution.

Definition 4.2.1. Fix a **sort of terms** $\mathbb{T}$. Define **sorts** τ by the following grammar:

$$\tau \quad ::= \quad \mathbb{T} \ \mid \ [\mathbb{A}]\mathbb{T} \ \mid \ [\mathbb{A}][\mathbb{A}]\mathbb{T}$$

We call $[\mathbb{A}]\mathbb{T}$ and $[\mathbb{A}][\mathbb{A}]\mathbb{T}$ **abstraction sorts**.

This particularly simple grammar consisting of only three sorts is sufficient for our needs. We only use sorts to make sure that terms are well-formed; the reader should not view it as a 'type system'.

Intuitively, the sorts $[\mathbb{A}]\mathbb{T}$ and $[\mathbb{A}][\mathbb{A}]\mathbb{T}$ contain terms that are abstractions $[a]t$ and $[a][b]t$, respectively. However, we will see that the sort $[\mathbb{A}]\mathbb{T}$ also contains some unknowns X and some terms of the form $\mathsf{sub}(t, u)$.

Definition 4.2.2. We assume the infinite collection of unknowns $X, Y, Z, \ldots$ is partitioned into two infinite collections $X_{\mathbb{T}}, Y_{\mathbb{T}}, Z_{\mathbb{T}}, \ldots$ of **unknowns of sort** $\mathbb{T}$ and $X_{[\mathbb{A}]\mathbb{T}}, Y_{[\mathbb{A}]\mathbb{T}}, Z_{[\mathbb{A}]\mathbb{T}}, \ldots$ of **unknowns of sort** $[\mathbb{A}]\mathbb{T}$.

We usually leave out the sort subscripts of the unknowns, as these sorts are usually clear from the context. We will also write $X : \tau$ for X **of sort** τ, where $\tau \in \{\mathbb{T}, [\mathbb{A}]\mathbb{T}\}$. Typically we let T and U be unknowns of sort $\mathbb{T}$.

Definition 4.2.3. We associate a **sorting arity** $(\tau_1, \ldots, \tau_n)\tau$ with each term-former f as follows:

- There are two term-formers sub, one with arity $([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}$, and one with arity $([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$.
- For any other term-former f, sorting arities are of the form $(\tau_1, \ldots, \tau_n)\mathbb{T}$, where $\tau_i \in \{\mathbb{T}, [\mathbb{A}]\mathbb{T}\}$, $1 \leq i \leq n$.

We write $\mathsf{f} : (\tau_1, \ldots, \tau_n)\tau$ for f **of arity** $(\tau_1, \ldots, \tau_n)\tau$.

The term-formers f excluding sub constitute the term-language in which atoms are going to be substituted. The sorting arities explicitly allow the use of abstraction sorts $[\mathbb{A}]\mathbb{T}$ for the arguments.

Example 4.2.4. The signature of theory LAM has the following term-formers:

$$\mathsf{lam} : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \mathsf{app} : (\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}.$$

So there are two term-formers sub that should take care of substitution, and there are two term-formers lam and app that constitute the term-language.

Remark 4.2.5. Term-former $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$ involves higher abstraction sorts than any of the unknowns and other term-formers. In combination with unknowns of sort $[\mathbb{A}]\mathbb{T}$, this term-former is convenient because it allows us to talk *about* the term-language of substitution at a general level. For instance, we can phrase the distributivity axiom

$$(\mathsf{f}{\mapsto}) \quad \vdash \quad \mathsf{f}(X_1, \ldots, X_n)[a \mapsto T] = \mathsf{f}(X_1[a \mapsto T], \ldots, X_n[a \mapsto T])$$

as a *schema* of axioms (one for each term-former f).

It is also possible to present the theory of substitution without term-former $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$ and unknowns of sort $[\mathbb{A}]\mathbb{T}$. This is made formal in Subsection 4.6.4. In fact, it is the approach we take in Chapter 5.

We will not use this more compact sort system here because it makes the presentation less general. Most importantly, in the presentation of the $(\mathsf{f}{\mapsto})$ axioms, we would need to *instantiate* the schema for each term-formers f (see Subsection 4.6.4 and Figure 5.1 for examples).

Definition 4.2.6. Let **(valid) sorting assertions** $t : \tau$, read 't has sort τ' be inductively defined by:

$$\frac{}{a : \mathbb{T}} \qquad \frac{}{\pi \cdot X_\tau : \tau} \qquad \frac{t : \tau}{[a]t : [\mathbb{A}]\tau} \qquad \frac{t_1 : \tau_1 \quad \cdots \quad t_n : \tau_n}{\mathsf{f}(t_1, \ldots, t_n) : \tau}\ (\mathsf{f} : (\tau_1, \ldots, \tau_n)\tau)$$

From now on, we will only consider terms that adhere to the valid sorting assertions. Furthermore, we require equalities $t = u$ and meta-level substitutions σ to be **sort-respecting**, i.e. $t = u$ is an equality when both $t : \tau$ and $u : \tau$. For σ we have the requirement that $\sigma(X) : \tau$ when $X : \tau$.

Note that in the above rules for moderated unknowns $\pi \cdot X_\tau$ and abstractions $[a]t$, τ is restricted to $\mathbb{T}$ and $[\mathbb{A}]\mathbb{T}$ by construction: for the case of $\pi \cdot X_\tau$, the τ subscript of X_τ is restricted to $\mathbb{T}$ or $[\mathbb{A}]\mathbb{T}$ by assumption; for the case of $[a]t$, τ can never be $[\mathbb{A}][\mathbb{A}]\mathbb{T}$ because then $[a]t$ would have sort $[\mathbb{A}][\mathbb{A}][\mathbb{A}]\mathbb{T}$ which is undefined.

Remark 4.2.7. The sorting discipline used here resembles the one from nominal unification [UPG04]. There are a number of important differences, though. On the one hand our setting is a bit more general because we allow unknowns of non-base sorts (i.e. $X : [\mathbb{A}]\mathbb{T}$), and one term-former of which the result sort is a non-base sort (i.e. $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$). On the other hand it is much more restricted because we only consider a finite hierarchy of abstractions sorts (i.e. only $[\mathbb{A}]\mathbb{T}$ and $[\mathbb{A}][\mathbb{A}]\mathbb{T}$) as opposed to the infinite hierarchy of [UPG04]. We are interested in lifting this restriction; this is current work.

We give some intuition of terms and their sorting assertions:

- An atom a represents a variable symbol of sort $\mathbb{T}$.
- A moderated unknown $\pi \cdot X : \tau$ represents an unknown term of sort τ on which a permutation of atoms is performed.
- An abstraction $[a]t : [\mathbb{A}]\tau$ represents a term of sort τ in which an atom a is abstracted.
- *Binders* are represented by term-formers that take abstractions as one or more of their arguments. So for example $\mathsf{lam} : ([\mathbb{A}]\mathbb{T})\mathbb{T}$ is a binder, and both term-formers sub are binders as well.
  The sort system is such that a well-sorted term of the form $\mathsf{lam}(t)$ must be of the form $\mathsf{lam}(\pi \cdot X)$, $\mathsf{lam}(\mathsf{sub}(t', u'))$ or $\mathsf{lam}([a]t')$ (so $t \equiv \pi \cdot X$, $t \equiv \mathsf{sub}(t', u')$ or $t \equiv [a]t'$). Similarly a well-sorted term of the form $\mathsf{sub}(t, u)$ must be of the form $\mathsf{sub}(\pi \cdot X, u)$, $\mathsf{sub}(\mathsf{sub}(t', u'), u)$, or $\mathsf{sub}([a]t', u)$, i.e. $t'[a \mapsto u]$.

We repeat the definition of theory SUB, cast into our sort system.

Definition 4.2.8. Call a signature Σ **suitable for** SUB when it includes term-formers $\mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}$ and $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$. Then SUB is a theory (Σ, Ax) where Σ is suitable for SUB and Ax is given by the axioms in Figure 4.1, in which:

- ($\mathsf{f}\mapsto$) represents a schema of axioms: there is one ($\mathsf{f}\mapsto$) axiom for each f in Σ excluding $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$ (but including $\mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}$).
- The ($\#\mapsto$) and (**ren**$\mapsto$) axioms each represent two axioms: one for $X : \mathbb{T}$ and one for $X : [\mathbb{A}]\mathbb{T}$.

For the rest of this chapter, we will only consider signatures suitable for SUB.

$$\begin{array}{rll}
(\mathbf{var}{\mapsto}) & \vdash & a[a\mapsto T] = T \\
(\#{\mapsto}) & a\#X \vdash & X[a\mapsto T] = X \\
(\mathsf{f}{\mapsto}) & \vdash & \mathsf{f}(X_1,\ldots,X_n)[a\mapsto T] = \mathsf{f}(X_1[a\mapsto T],\ldots,X_n[a\mapsto T]) \\
(\mathbf{abs}{\mapsto}) & b\#T \vdash & ([b]U)[a\mapsto T] = [b](U[a\mapsto T]) \\
(\mathbf{ren}{\mapsto}) & b\#X \vdash & X[a\mapsto b] = (b\ a)\cdot X \\
(\eta{\mapsto}) & a\#X \vdash & [a]\mathsf{sub}(X,a) = X
\end{array}$$

Figure 4.1 Axioms of theory SUB

Informally, the axioms of Figure 4.1 express the following:

($\mathbf{var}{\mapsto}$): If a is a variable then a with a replaced by T, is T.

($\#{\mapsto}$): If a is fresh for X then X with a replaced by T is X.

($\mathsf{f}{\mapsto}$): Substitution distributes through term-formers.

($\mathbf{abs}{\mapsto}$): Substitution of a distributes under an abstraction $[b]U$, provided a capture-avoidance condition holds (b is fresh for T).

($\mathbf{ren}{\mapsto}$): If b is fresh for X then X with a replaced by b is identical to X with a replaced by b and *simultaneously* b replaced by a.

($\eta{\mapsto}$): This axiom generalises the property that

$$\Delta \vdash a\#t \quad \text{implies} \quad \Delta \vdash_{\mathsf{SUB}} [a](t[b\mapsto a]) = [b]t$$

from abstractions $[b]t : [\mathbb{A}]\mathbb{T}$ to all terms of sort $[\mathbb{A}]\mathbb{T}$, i.e. the property also holds for unknowns X and substitutions $\mathsf{sub}(t', u')$.
The reader can think of ($\eta{\mapsto}$) as related to *η-equality* from the world of the λ-calculus, though X is not a function. The reader can also think of ($\eta{\mapsto}$) as related to a known property of the *atoms-concretion* operation of Gabbay-Pitts abstraction [GP02], though sub is not atoms-concretion.

Remark 4.2.9. We require the presence of a term-former taking at least two arguments (like app from theory LAM) to exclude degenerate models. This is only needed for so-called ω-completeness (Theorem 4.6.28), which is a *very* strong notion of completeness with respect to a concrete model.[1] The other major results, including a weaker notion of completeness (Theorem 4.6.19), are in no danger and remain valid even in a signature with just sub.

4.3 Substitution on Ground Terms

In this section we build a concrete model of syntax from the syntax of SUB, on which we define the usual notion of capture-avoiding substitution. We will use this model later to express the *meaning* of explicit substitution.

[1]The required term-former is needed in its proof to construct a translation -* from terms in an extended signature to terms in the original syntax (see Subsection 4.6.3).

Definition 4.3.1. Call a term **ground** when it does not mention any unknowns or explicit substitutions. Ground terms are inductively characterised by:

$$g \quad ::= \quad a \ \mid \ [a]g \ \mid \ \mathsf{f}(g, \dots, g)$$

where f ranges over all term-formers except for sub.

We can easily define a 'free variables' function on ground terms. We call it the 'free atoms' function, since variables are represented by atoms in nominal algebra.

Definition 4.3.2. Define a '**free atoms**' function $fa(g)$ on ground terms inductively as follows:

$$fa(a) = \{a\} \qquad fa([a]g) = fa(g) \setminus \{a\} \qquad fa(\mathsf{f}(g_1, \dots, g_n)) = \bigcup_{1 \le i \le n} fa(g_i)$$

On ground terms, 'not in the free atoms of' coincides with derivability of freshness:

Lemma 4.3.3 (Freshness on ground terms)**.** *For ground terms g:*

$$\vdash a\#g \quad \textit{if and only if} \quad a \notin fa(g).$$

Proof. By induction on the structure of g. □

Definition 4.3.4. Let the **size** of a ground term be inductively defined by:

$$|a| = 1 \qquad |[a]g| = |g| + 1 \qquad |\mathsf{f}(g_1, \dots, g_n)| = |g_1| + \dots + |g_n| + 1$$

Definition 4.3.5. For each finite set of atoms arbitrarily choose some canonical 'fresh' atom not in that finite set. Then define a **ground substitution action** $g[h/a]$ on ground terms of sort $\mathbb{T}$ and $[\mathbb{A}]\mathbb{T}$ inductively on the *size* of g by:

$$\begin{gathered}
a[h/a] \equiv h \qquad b[h/a] \equiv b \\
([a]g)[h/a] \equiv [a]g \qquad ([b]g)[h/a] \equiv [b](g[h/a]) \quad (b \notin fa(h)) \\
([b]g)[h/a] \equiv [c](g[c/b][h/a]) \quad (b \in fa(h),\ c \text{ fresh}) \\
\mathsf{f}(g_1, \dots, g_n)[h/a] \equiv \mathsf{f}(g_1[h/a], \dots, g_n[h/a]),
\end{gathered}$$

where f ranges over all term-formers excluding sub. 'c fresh' means c is chosen such that $c \notin \{a, b\} \cup fa(g) \cup fa(h)$ according to our arbitrary choice. We will not mention $c \notin \{a, b\}$ anymore in the remainder of this chapter, since this is enforced by our permutative convention on atoms (see Definition 2.3.12).

Note that the ground substitution action is well-defined, since $|g[c/b]| = |g|$ in the penultimate case of Definition 4.3.5, as can be shown by induction on the size of g. We will often use this fact that capture-avoiding substitution of atoms for atoms preserves size.

We will also use the following simple property relating freshness and capture-avoiding substitution on ground terms:

Lemma 4.3.6. *For ground terms g, h,* $\quad fa(g[h/a]) \quad\subseteq\quad (fa(g)\backslash\{a\}) \cup fa(h)$.

Proof. By induction on the size of g. In the calculations we indicate uses of the inductive hypothesis with a superscript ^{IH}. We consider the three more interesting cases in turn:

- $g \equiv [a]g'$: Then $([a]g')[h/a] \equiv [a]g'$. We must show
$$fa(g') \setminus \{a\} \quad\subseteq\quad (fa(g') \setminus \{a\}) \cup fa(h),$$
which is trivial.
- $g \equiv [b]g'$, $b \notin fa(h)$: Then $([b]g')[h/a] \equiv [b](g'[h/a])$. We calculate as follows:
$$fa(g'[h/a])\backslash\{b\} \overset{IH}{\subseteq} ((fa(g')\backslash\{a\}) \cup fa(h))\backslash\{b\} \;=\; ((fa(g')\backslash\{b\})\backslash\{a\}) \cup fa(h).$$
- $g \equiv [b]g'$, $b \in fa(h)$: Then $([b]g')[h/a] \equiv [c](g'[c/b][h/a])$, where c is our choice of fresh atom such that $c \notin fa(g') \cup fa(h)$. We must show
$$(fa(g'[c/b][h/a]) \setminus \{c\}) \subseteq ((fa(g') \setminus \{b\}) \setminus \{a\}) \cup fa(h),$$
which we calculate as follows:
$$\begin{aligned} fa(g'[c/b][h/a]) \setminus \{c\} &\overset{IH}{\subseteq} ((fa(g'[c/b]) \setminus \{a\}) \cup fa(h)) \setminus \{c\} \\ &= ((fa(g'[c/b]) \setminus \{c\}) \setminus \{a\}) \cup fa(h) \\ &\overset{IH}{\subseteq} ((((fa(g') \setminus \{b\}) \cup fa(c)) \setminus \{c\}) \setminus \{a\}) \cup fa(h) \\ &= ((fa(g') \setminus \{b\}) \setminus \{a\}) \cup fa(h). \end{aligned}$$
In the first application of the inductive hypothesis above, we use the fact that $|g'[c/b]| < |[b]g'|$.

$\square$

Lemma 4.3.7. *For ground terms g, h:*

1. *If $\vdash a\#h$ then $\vdash a\#g[h/a]$.*
2. *If $\vdash a\#g$ and $\vdash a\#h$ then $\vdash a\#g[h/b]$.*

Proof. By Lemma 4.3.3 it suffices to show that, using the contrapositive:

1. If $a \in fa(g[h/a])$ then $a \in fa(h)$.
2. If $a \in fa(g[h/b])$ then $a \in fa(g)$ and $a \in fa(h)$.

This is easy using Lemma 4.3.6. $\square$

Lemma 4.3.8 states familiar properties of ground terms — but are they true? Lemma 4.3.8 makes the vital connection between 'substitution as we know it' and the nominal technology we bring to bear on it. We give proofs in some detail. They run smoothly, and this fact is encouraging evidence that our definitions are appropriate:

Lemma 4.3.8. *For ground terms g, h, k:*

1. ***Identity.*** $\vdash_{\mathsf{CORE}} g[a/a] = g$.
2. ***Swapping.*** *If* $\vdash b\#g$ *then* $\vdash_{\mathsf{CORE}} g[b/a] = (b\ a)\cdot g$.
3. ***Garbage collection.*** *If* $\vdash a\#g$ *then* $\vdash_{\mathsf{CORE}} g[h/a] = g$.
4. ***Distributivity.*** *If* $\vdash a\#k$ *then* $\vdash_{\mathsf{CORE}} g[h/a][k/b] = g[k/b][h[k/b]/a]$.

Proof of parts 1, 2 and 3. Part 1 follows from the stronger property $g[a/a] \equiv g$. We prove that by an induction on the size of g which we omit.

We show part 2 by induction on the size of g. Most cases are easy; the interesting ones are:

- $g \equiv [a]g'$. Then $([a]g')[b/a] \equiv [a]g'$, so we must show $\vdash_{\mathsf{CORE}} [a]g' = (b\ a)\cdot[a]g'$. By (**symm**) and (**perm**), this follows from assumption $b\#[a]g'$ and from $a\#[a]g'$, which follows by ($\#[]$**a**).
- $g \equiv [b]g'$. Then $([b]g')[b/a] \equiv [c](g'[c/b][b/a])$ where $c \notin fa(g')$ is our choice of fresh atom. We must now show that

$$\vdash_{\mathsf{CORE}} [c](g'[c/b][b/a]) = [a](b\ a)\cdot g'.$$

By (**symm**) and the syntactic criteria of Corollary 2.5.4, this happens when

$$\vdash c\#(b\ a)\cdot g' \quad\text{and}\quad \vdash_{\mathsf{CORE}} g'[c/b][b/a] = (c\ a)\cdot(b\ a)\cdot g'.$$

We consider each part in turn.
Since $c \notin fa(g)$, we know $\vdash c\#(b\ a)\cdot g'$ by Lemma 4.3.3 and object-level equivariance (Theorem 2.4.6).
To prove $\vdash_{\mathsf{CORE}} g'[c/b][b/a] = (c\ a)\cdot(b\ a)\cdot g'$ it suffices to show that

$$\vdash_{\mathsf{CORE}} g'[c/b][b/a] = (b\ a)\cdot g'[c/b] \quad\text{and}\quad \vdash_{\mathsf{CORE}} (b\ a)\cdot g'[c/b] = (b\ a)\cdot(c\ b)\cdot g',$$

by (**tran**).
Now $|g'[c/b]| < |[b]g'|$ so we use the inductive hypothesis to deduce

$$\vdash_{\mathsf{CORE}} g'[c/b][b/a] = (b\ a)\cdot g'[c/b]$$

from $\vdash b\#g'[c/b]$, which follows by Lemma 4.3.7.
We deduce $\vdash_{\mathsf{CORE}} (b\ a)\cdot g'[c/b] = (b\ a)\cdot(c\ b)\cdot g'$ from $\vdash_{\mathsf{CORE}} g'[c/b] = (c\ b)\cdot g'$ using object-level equivariance (Theorem 2.4.6). By the inductive hypothesis and Lemma 4.3.3, this follows from the assumption $c \notin fa(g')$.
- $g \equiv [c]g'$. Then $([c]g')[b/a] \equiv [c](g'[b/a])$, so we need to show

$$\vdash_{\mathsf{CORE}} [c](g'[b/a]) = [c](b\ a)\cdot g'.$$

This follows directly from the inductive hypothesis using (**cong**[]).

We show part 3 by induction on the size of g using the syntactic criteria of Corollary 2.5.4. The only interesting case is when $g \equiv [b]g'$ and $b \in \mathit{fa}(h)$. Then $([b]g')[h/a] \equiv [c](g'[c/b][h/a])$ where $c \notin \mathit{fa}(g') \cup \mathit{fa}(h)$, so we must show

$$\vdash_{\mathsf{CORE}} [c](g'[c/b][h/a]) = [b]g'.$$

By (**tran**) this follows from

$$\vdash_{\mathsf{CORE}} [c](g'[c/b][h/a]) = [c](c\ b) \cdot g' \quad \text{and} \quad \vdash_{\mathsf{CORE}} [c](c\ b) \cdot g' = [b]g'.$$

By (**perm**), $\vdash_{\mathsf{CORE}} [c](c\ b) \cdot g' = [b]g'$ follows from $\vdash b\#[b]g'$ and $\vdash c\#[b]g'$, which follow from assumption $c \notin \mathit{fa}(g')$ by the rules for freshness and Lemma 4.3.3. By (**cong**[]) and (**tran**), $\vdash_{\mathsf{CORE}} [c](g'[c/b][h/a]) = [c](c\ b) \cdot g'$ follows from

$$\vdash_{\mathsf{CORE}} g'[c/b][h/a] = g'[c/b] \quad \text{and} \quad \vdash_{\mathsf{CORE}} g'[c/b] = (c\ b) \cdot g'.$$

By the inductive hypothesis, $\vdash_{\mathsf{CORE}} g'[c/b][h/a] = g'[c/b]$ follows from $\vdash a\#g'[c/b]$, which follows from assumption $\vdash a\#[b]g'$ by Lemmas 4.3.7 and 2.4.2. Finally, $\vdash_{\mathsf{CORE}} g'[c/b] = (c\ b) \cdot g'$ is an instance of part 2 of this lemma, since $\vdash c\#g'$. □

We can prove part 4 of Lemma 4.3.8 by induction on the size of g, using parts 2 and 3 and congruence of capture-avoiding substitution on ground terms. We will only provide the congruence properties here.

Lemma 4.3.9. *For ground terms g, h, k:*

1. *If $\vdash_{\mathsf{CORE}} g = h$ then $\vdash_{\mathsf{CORE}} g[k/a] = h[k/a]$.*
2. *If $\vdash_{\mathsf{CORE}} h = k$ then $\vdash_{\mathsf{CORE}} g[h/a] = g[k/a]$.*

Proof. The proof of the first part is by induction on the size of g, using the syntactic criteria of Corollary 2.5.4. We consider the two most interesting cases:

- $g \equiv [a]g'$. Then there are two possibilities:
  - $h \equiv [a]h'$ and $\vdash_{\mathsf{CORE}} g' = h'$. Then we must show $\vdash_{\mathsf{CORE}} [a]g' = [a]h'$ since $([a]g')[k/a] \equiv [a]g'$ and $([a]h')[k/a] \equiv [a]h'$. The result follows from the assumption $\vdash_{\mathsf{CORE}} g' = h'$ by (**cong**[]).
  - $h \equiv [b]h'$, $\vdash b\#g'$ and $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$. Then since $\vdash b\#g'$, we also have $\vdash a\#(b\ a) \cdot g'$ by object-level equivariance (Theorem 2.4.6). Using Corollary 2.5.6 and assumption $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$ we obtain $\vdash a\#h'$. Then $\vdash a\#[b]h'$ by (#[]**b**), so by part 3 of Lemma 4.3.8 we obtain $\vdash_{\mathsf{CORE}} ([b]h')[k/a] = [b]h'$.
    Now by (**cong**[]), (**symm**) and assumption $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$, we have $\vdash_{\mathsf{CORE}} [b]h' = (b\ a) \cdot [a]g'$. Also $\vdash_{\mathsf{CORE}} (b\ a) \cdot [a]g' = [a]g'$ by (**perm**) and assumption $\vdash b\#g'$. Since $[a]g' \equiv ([a]g')[k/a]$, we may use (**tran**) and (**symm**) to conclude $\vdash_{\mathsf{CORE}} ([a]g')[k/a] = ([b]h')[k/a]$, as required.
- $g \equiv [b]g'$. Then again there are two possibilities:

$$\frac{}{g =_\alpha g} \qquad \frac{g =_\alpha h}{h =_\alpha g} \qquad \frac{g_1 =_\alpha g_2 \quad g_2 =_\alpha g_3}{g_1 =_\alpha g_3}$$

$$\frac{g =_\alpha h}{[a]g =_\alpha [a]h} \qquad \frac{g_1 =_\alpha h_1 \ \cdots \ g_n =_\alpha h_n}{\mathsf{f}(g_1,\ldots,g_n) =_\alpha \mathsf{f}(h_1,\ldots,h_n)}$$

$$\frac{g[c/a] =_\alpha h[c/b]}{[a]g =_\alpha [b]h}\ (c \text{ fresh})$$

Figure 4.2 α-equivalence on ground terms

- $h \equiv [a]h'$, $\vdash a\#g'$ and $\vdash_{\mathsf{CORE}} (a\ b)\cdot g' = h'$. Completely analogous to the previous part we can show that $\vdash_{\mathsf{CORE}} ([b]g')[k/a] = ([a]h')[k/a]$.
- $g \equiv [b]h'$ and $\vdash_{\mathsf{CORE}} g' = h'$.
  If $b \notin fa(k)$ we must show $\vdash_{\mathsf{CORE}} [b](g'[k/a]) = [b](h'[k/a])$, which follows from the assumption $\vdash_{\mathsf{CORE}} g' = h'$ by (**cong**[]) and the inductive hypothesis.
  If $b \in fa(k)$ we must show $\vdash_{\mathsf{CORE}} [c](g'[c/b][k/a]) = [c](h'[c/b][k/a])$ where $c \notin fa(g') \cup fa(h') \cup fa(k)$.[2] By an instance of (**cong**[]) this follows from $\vdash_{\mathsf{CORE}} g'[c/b][k/a] = h'[c/b][k/a]$. By inductive hypothesis, this follows from $\vdash_{\mathsf{CORE}} g'[c/b] = h'[c/b]$. Since $\vdash c\#g'$ and $\vdash c\#h'$, this is equivalent to $\vdash_{\mathsf{CORE}} (c\ b)\cdot g' = (c\ b)\cdot h'$ by part 2 of Lemma 4.3.8. By object-level equivariance (Theorem 2.4.6) this follows from $\vdash_{\mathsf{CORE}} g' = h'$, which we assumed.

The proof of the second part is similar, but simpler. □

We will now show how equality on ground terms in theory CORE coincides with a straightforward definition of α-equivalence on ground terms: syntactic equality extended with a rule to rename bound variables.

Definition 4.3.10. Define **α-equivalence** $g =_\alpha h$ on ground terms g and h inductively by the rules in Figure 4.2. Here 'c fresh' means *any* c such that $c \notin \{a, b\} \cup fa(g) \cup fa(h)$.

Before we can relate CORE equality and $=_\alpha$, we need to establish a few technical properties of $=_\alpha$.

Lemma 4.3.11. *If $b \notin g$ then $[b](g[b/a]) =_\alpha [a]g$.*

[2]Both $([b]g')[k/a]$ and $([b]h')[k/a]$ introduce the same fresh atom c because $fa(g') = fa(h')$. We can see this as follows: $fa(g') = fa(h')$ is equivalent to $a' \notin fa(g')$ if and only if $a' \notin fa(h')$ for all atoms a'. By Lemma 4.3.3 this is equivalent to $\vdash a'\#g'$ if and only if $\vdash a'\#h'$, and by Corollary 2.5.6, this follows from the assumption $\vdash_{\mathsf{CORE}} g' = h'$.

Proof. Let $c \notin g$. By the α-conversion rule of Figure 4.2, $[b](g[b/a]) =_\alpha [a]g$ follows from $g[b/a][c/b] =_\alpha g[c/a]$. This follows by reflexivity since we can show $g[b/a][c/b] \equiv g[c/a]$ by an easy induction on the structure of g: the cases of a and d are trivial, the cases of $\mathsf{f}(g_1, \dots, g_n)$ and $[d]g'$ follow using the inductive hypothesis, the case of $[a]g'$ follows from the fact that $g'[c/b] \equiv g'$, and the remaining cases b, c, $[b]g'$ and $[c]g'$ are vacuously true. □

Lemma 4.3.11 enables us to prove that $=_\alpha$ can simulate the (**perm**) rule.

Lemma 4.3.12. *If $a, b \notin fa(g)$ then $(a\ b) \cdot g =_\alpha g$.*

Proof. Since $a, b \notin fa(g)$, all instances of a and b in g occur in the scope of abstractors $[a]$ and $[b]$. Traverse the structure of g bottom-up, and use Lemma 4.3.11 to rename those abstractors so they are not a or b anymore, but some completely fresh set of atoms — a different atom for each instance of $[a]$ and $[b]$. Call the new term g', then it is easy to show $(a\ b) \cdot g' \equiv g'$, since $a, b \notin g'$. Now equality is symmetric, so we reverse the process to get g back again. □

We now have all the ingredients to show the main result of this section.

Theorem 4.3.13 (CORE on ground terms)**.** *For ground terms g, h:*

$$\vdash_{\mathsf{CORE}} g = h \quad \textit{if and only if} \quad g =_\alpha h.$$

Proof. We prove the left-to-right implication by induction on the structure of g, using the syntactic criteria for CORE-equality (Corollary 2.5.4). The case of $g \equiv a$ follows by reflexivity, and the case of $g \equiv \mathsf{f}(g_1, \dots, g_n)$ follows by congruence using the inductive hypothesis. Now suppose $g \equiv [a]g'$, then there are two possibilities:

1. $h \equiv [a]h'$ and $\vdash_{\mathsf{CORE}} g' = h'$. Then $g' =_\alpha h'$ by the inductive hypothesis, and we conclude $[a]g' =_\alpha [a]h'$ by congruence.
2. $h \equiv [b]h'$, $\vdash b\#g'$ and $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$. By Lemma 4.3.3 and some easy calculations we know $a, b \notin fa([a]g')$, so $[a]g' =_\alpha [b](b\ a) \cdot g'$ by Lemma 4.3.12 and symmetry. Also $[b](b\ a) \cdot g' =_\alpha [b]h'$ by congruence and the inductive hypothesis. We conclude $[a]g' =_\alpha [b]h'$ by transitivity.

Conversely suppose that $g =_\alpha h$. It suffices to show that equality in CORE can simulate every derivation rule of $=_\alpha$. We treat the only non-trivial case.

Suppose we have deduced $[a]g =_\alpha [b]h$ from $g[c/a] =_\alpha h[c/b]$, where c is chosen fresh. Since $c \notin fa(g) \cup fa(h)$, we know $\vdash c\#g$ and $\vdash c\#h$, and we obtain

$$\vdash_{\mathsf{CORE}} g[c/a] = (c\ a) \cdot g \qquad \text{and} \qquad \vdash_{\mathsf{CORE}} h[c/b] = (c\ b) \cdot h$$

by part 2 of Lemma 4.3.8.

Now also $\vdash_{\mathsf{CORE}} g[c/a] = h[c/b]$ by the inductive hypothesis. Then we obtain

$$\vdash_{\mathsf{CORE}} [c](c\ a) \cdot g = [c](c\ b) \cdot h.$$

by (**symm**), (**tran**) and (**cong[]**).

By (**perm**) $\vdash_{\mathsf{CORE}} [c](c\ a) \cdot g = [a]g$ and $\vdash_{\mathsf{CORE}} [c](c\ b) \cdot h = [b]h$, since $\vdash c\#g$ and $\vdash c\#h$. Using (**symm**) and (**tran**) we conclude that $\vdash_{\mathsf{CORE}} [a]g = [b]h$. □

$$
\begin{array}{llrcll}
(\mathbf{var}\mapsto) & \vdash & a[a\mapsto T] & = & T & \\
(\mathbf{b}\mapsto) & \vdash & b[a\mapsto T] & = & b & \\
(\mathsf{f}\mapsto) & \vdash & \mathsf{f}(X_1,\dots,X_n)[a\mapsto T] & = & \mathsf{f}(X_1[a\mapsto T],\dots,X_n[a\mapsto T]) & (\mathsf{f}\neq\mathsf{sub}) \\
(\mathbf{abs}\mapsto) & c\#T\vdash & ([c]U)[a\mapsto T] & = & [c](U[a\mapsto T]) &
\end{array}
$$

Figure 4.3 Axioms of theory SIMP

4.4 Theory SIMP: Substitution on Closed Terms

We define a nominal algebra theory SIMP which is sound but not complete with respect to the ground term model from Section 4.3. This is an important technical step towards SUB because we shall prove properties of SUB by reducing them to properties of SIMP.

Definition 4.4.1. Let SIMP be the nominal algebra theory with axioms as in Figure 4.3.

Here, f ranges over all term-formers except for sub. Recall that T and U are unknowns of $\mathbb{T}$, and that the X_i are unknowns of sort $\mathbb{T}$ or $[\mathbb{A}]\mathbb{T}$ (which one applies depends on the instance of f).

Lemma 4.4.2. *For ground terms g, h, if $\vdash a\#g$ then $\vdash_{\mathsf{SIMP}} g[a\mapsto h] = g$.*

Proof. We work by induction on the size of g. We consider the cases in turn:

- $g \equiv a$. Then $\nvdash a\#a$ and there is nothing to prove.
- $g \equiv b$. Then $\vdash a\#b$. $\vdash_{\mathsf{SIMP}} b[a\mapsto h] = b$ by axiom ($\mathbf{b}\mapsto$).
- $g \equiv [a]g'$. Take c fresh for g' and h. Then $\vdash_{\mathsf{CORE}} [c](c\ a)\cdot g' = [a]g'$ by (**perm**) since $\vdash c\#g'$ and $\vdash a\#g'$. Using (**symm**), (**cong**[]) and (**congf**) we obtain

  $$\vdash_{\mathsf{CORE}} ([a]g')[a\mapsto h] = ([c](c\ a)\cdot g')[a\mapsto h].$$

  Now by (**abs**$\mapsto$) also

  $$\vdash_{\mathsf{SIMP}} ([c](c\ a)\cdot g')[a\mapsto h] = [c](((c\ a)\cdot g')[a\mapsto h])$$

  since $\vdash c\#h$. Since $\vdash a\#(c\ a)\cdot g'$ we know $\vdash_{\mathsf{SIMP}} ((c\ a)\cdot g')[a\mapsto h] = (c\ a)\cdot g'$ by the inductive hypothesis ($(c\ a)\cdot g'$ has the same size as g', and so is smaller than $[a]g'$). Using (**cong**[]) and $\vdash_{\mathsf{CORE}} [c](c\ a)\cdot g' = [a]g'$ we obtain

  $$\vdash_{\mathsf{SIMP}} [c]((c\ a)\cdot g')[a\mapsto h] = [a]g'.$$

  We use (**tran**) to obtain $\vdash_{\mathsf{SIMP}} ([a]g')[a\mapsto h] = [a]g'$, as required.
- $g \equiv [b]g'$. Analogous to the previous part.

- $g \equiv \mathsf{f}(g_1, \ldots, g_n)[a \mapsto h]$, $\mathsf{f} \neq \mathsf{sub}$. By assumption $\vdash a\#\mathsf{f}(g_1, \ldots, g_n)$. Then $\vdash a\#g_i$ by Theorem 2.4.2 and $\vdash_{\mathsf{SIMP}} g_i[a \mapsto h] = g_i$ by the inductive hypothesis for $1 \leq i \leq n$. Using (**congf**) and (**tran**) we obtain

  $$\vdash_{\mathsf{SIMP}} \mathsf{f}(g_1[a \mapsto h], \ldots, g_n[a \mapsto h]) = \mathsf{f}(g_1, \ldots, g_n).$$

  Now by axiom ($\mathsf{f}\mapsto$)

  $$\vdash_{\mathsf{SIMP}} \mathsf{f}(g_1, \ldots, g_n)[a \mapsto h] = \mathsf{f}(g_1[a \mapsto h], \ldots, g_n[a \mapsto h]).$$

  We obtain $\vdash_{\mathsf{SIMP}} \mathsf{f}(g_1, \ldots, g_n)[a \mapsto h] = \mathsf{f}(g_1, \ldots, g_n)$ by (**tran**), as required.

□

Theorem 4.4.3. *For ground terms g, h, $\vdash_{\mathsf{SIMP}} g[a \mapsto h] = g[h/a]$.*

Proof. By induction on the size of g:

- $g \equiv a$. Then $\vdash_{\mathsf{SIMP}} a[a \mapsto h] = h$ by axiom (**var**$\mapsto$), and $a[h/a] \equiv h$.
- $g \equiv b$. Then $\vdash_{\mathsf{SIMP}} b[a \mapsto h] = b$ by axiom (**b**$\mapsto$), and $b[h/a] \equiv b$.
- $g \equiv [a]g'$. Then $\vdash_{\mathsf{SIMP}} ([a]g')[a \mapsto t] = [a]g'$ by Lemma 4.4.2 since $\vdash a\#[a]g'$. We are done since $([a]g')[h/a] \equiv [a]g'$.
- $g \equiv [b]g'$, $b \notin fa(h)$. Then $\vdash_{\mathsf{SIMP}} ([b]g')[a \mapsto h] = [b](g'[a \mapsto h])$ is an instance of axiom (**abs**$\mapsto$), since $\vdash b\#h$ by Lemma 4.3.3. By the inductive hypothesis also $\vdash_{\mathsf{SIMP}} g'[a \mapsto h] = g'[h/a]$, so we conclude $([b]g')[a \mapsto h] = [b](g'[h/a])$ using (**cong**[]) and (**tran**). We are done, since $([b]g')[h/a] \equiv [b](g'[h/a])$.
- $g \equiv [b]g'$, $b \in fa(h)$. Then $([b]g')[h/a] \equiv [c](g[c/b][h/a])$ where c is a fresh atom according to our arbitary choice, i.e. $c \notin fa(g') \cup fa(h)$. Then also $\vdash c\#g'$ and $\vdash c\#h$ by Lemma 4.3.3 . Now $\vdash_{\mathsf{CORE}} [c](c\ b) \cdot g' = [b]g'$ by (**perm**) since $\vdash b\#[b]g'$ and $\vdash c\#[b]g'$. By (**symm**), (**cong**[]) and (**congf**) we obtain

  $$\vdash_{\mathsf{CORE}} ([b]g')[a \mapsto h] = ([c](c\ b) \cdot g')[a \mapsto h].$$

  By axiom (**abs** $\mapsto$) also

  $$\vdash_{\mathsf{SIMP}} ([c](c\ b) \cdot g')[a \mapsto h] = [c](((c\ b) \cdot g')[a \mapsto h])$$

  since $\vdash c\#h$. By the inductive hypothesis and (**cong**[])

  $$\vdash_{\mathsf{SIMP}} [c](((c\ b) \cdot g')[a \mapsto h]) = [c](((c\ b) \cdot g')[h/a]).$$

  Since $\vdash_{\mathsf{CORE}} g'[c/b] = (c\ b) \cdot g'$ by part 2 of Lemma 4.3.8, we deduce

  $$\vdash_{\mathsf{SIMP}} [c](((c\ b) \cdot g')[h/a]) = [c](g'[c/b][h/a])$$

  using the rules of equality and Lemma 4.3.9. Using (**tran**) we put the pieces together to conclude

  $$\vdash_{\mathsf{SIMP}} ([b]g')[a \mapsto h] = [c](g'[c/b][h/a]),$$

  as required.

- $g \equiv \mathsf{f}(g_1, \ldots, g_n)$, $\mathsf{f} \neq \mathsf{sub}$. Then

  $$\vdash_{\mathsf{SIMP}} \mathsf{f}(g_1, \ldots, g_n)[a \mapsto h] = \mathsf{f}(g_1[a \mapsto h], \ldots, g_n[a \mapsto h])$$

  by ($\mathsf{f}\mapsto$). By the inductive hypothesis $\vdash_{\mathsf{SIMP}} g_i[a \mapsto h] = g_i[h/a]$ for $1 \leq i \leq n$. Then using (**congf**) and (**tran**) we obtain

  $$\vdash_{\mathsf{SIMP}} \mathsf{f}(g_1, \ldots, g_n)[a \mapsto h] = \mathsf{f}(g_1[h/a], \ldots, g_n[h/a])$$

  Since $\mathsf{f}(g_1, \ldots, g_n)[h/a] \equiv \mathsf{f}(g_1[h/a], \ldots, g_n[h/a])$ we are done.

□

On closed terms we can interpret all occurrences of term-former sub by capture-avoiding substitution.

Definition 4.4.4. Define the **translation** $t^{\mathcal{I}}$ of closed terms t to ground terms inductively on closed terms by:

$$a^{\mathcal{I}} \equiv a \qquad ([a]t)^{\mathcal{I}} \equiv [a](t^{\mathcal{I}}) \qquad \mathsf{sub}(t,u)^{\mathcal{I}} \equiv g[u^{\mathcal{I}}/a] \quad ([a]g \equiv t^{\mathcal{I}})$$
$$\mathsf{f}(t_1, \ldots, t_n)^{\mathcal{I}} \equiv \mathsf{f}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}}) \quad (\mathsf{f} \neq \mathsf{sub}).$$

Note that $(t[a \mapsto u])^{\mathcal{I}} \equiv t^{\mathcal{I}}[u^{\mathcal{I}}/a]$ follows from the sub case, since $[a](t^{\mathcal{I}}) \equiv ([a]t)^{\mathcal{I}}$ by the abstraction case.

Lemma 4.4.5 (Freshness on closed terms)**.** *For any closed term t:*

$$\textit{if } \vdash a\#t \textit{ then } \vdash a\#t^{\mathcal{I}}.$$

Proof. By induction on the structure of t. The only interesting case is when $t \equiv \mathsf{sub}(u, v)$. By assumption we know $\vdash a\#\mathsf{sub}(u, v)$. By Lemma 2.4.2 also $\vdash a\#u$ and $\vdash a\#v$. By the inductive hypothesis, $\vdash a\#u^{\mathcal{I}}$ and $\vdash a\#v^{\mathcal{I}}$.

We proceed by case distinction on $u^{\mathcal{I}}$:

- $u^{\mathcal{I}} \equiv [b]g$ for some atom b. Then $\mathsf{sub}(u, v)^{\mathcal{I}} \equiv g[v^{\mathcal{I}}/b]$ and we must show $\vdash a\#g[v^{\mathcal{I}}/b]$. This follows from $\vdash a\#v^{\mathcal{I}}$ and $\vdash a\#g$ by Lemma 4.3.7. $\vdash a\#v^{\mathcal{I}}$ is an assumption, and $\vdash a\#g$ follows from assumption $\vdash a\#u^{\mathcal{I}}$ (recall that $u^{\mathcal{I}} \equiv [b]g$) by Lemma 2.4.2.
- $u^{\mathcal{I}} \equiv [a]g$. Then $\mathsf{sub}(u, v)^{\mathcal{I}} \equiv g[v^{\mathcal{I}}/a]$, and $\vdash a\#g[v^{\mathcal{I}}/a]$ follows from assumption $\vdash a\#v^{\mathcal{I}}$ by Lemma 4.3.7.

□

Note that the converse of Lemma 4.4.5 does not hold. Consider for instance the term $b[c \mapsto a]$. Then $\vdash a\#(b[c \mapsto a])^{\mathcal{I}}$ since $(b[c \mapsto a])^{\mathcal{I}} \equiv b$, but $\nvdash a\#b[c \mapsto a]$ since $\nvdash a\#a$.

Theorem 4.4.6 (SIMP on closed terms)**.** *For any closed term t:*

$$\vdash_{\mathsf{SIMP}} t = t^{\downarrow}.$$

Proof. By induction on the structure of t:

- $t \equiv a$. Then $a^{\downarrow} \equiv a$, and we conclude $\vdash_{\mathsf{SIMP}} a = a$ by (**refl**).
- $t \equiv [a]u$. Then $([a]u)^{\downarrow} \equiv [a](u^{\downarrow})$. By (**cong**[]) $\vdash_{\mathsf{SIMP}} [a]u = [a](u^{\downarrow})$ follows from $\vdash_{\mathsf{SIMP}} u = u^{\downarrow}$, which holds by the inductive hypothesis.
- $t \equiv \mathsf{f}(t_1, \ldots, t_n)$, $\mathsf{f} \neq \mathsf{sub}$. Then $\mathsf{f}(t_1, \ldots, t_n)^{\downarrow} \equiv \mathsf{f}(t_1^{\downarrow}, \ldots, t_n^{\downarrow})$. By (**congf**) and (**tran**) $\vdash_{\mathsf{SIMP}} \mathsf{f}(t_1, \ldots, t_n) = \mathsf{f}(t_1^{\downarrow}, \ldots, t_n^{\downarrow})$ follows from $\vdash_{\mathsf{SIMP}} t_i = t_i^{\downarrow}$ for $1 \leq i \leq n$, which hold by the inductive hypothesis.
- $t \equiv \mathsf{sub}(u, v)$. Then $\mathsf{sub}(u, v)^{\downarrow} \equiv g[v^{\downarrow}/a]$ where $[a]g \equiv u^{\downarrow}$. By the inductive hypothesis, $\vdash_{\mathsf{SIMP}} u = u^{\downarrow}$ and $\vdash_{\mathsf{SIMP}} v = v^{\downarrow}$. By (**congf**) and (**tran**), we obtain

  $$\vdash_{\mathsf{SIMP}} \mathsf{sub}(u, v) = \mathsf{sub}(u^{\downarrow}, v^{\downarrow}).$$

  By Theorem 4.4.3 we know $\vdash_{\mathsf{SIMP}} g[a \mapsto v^{\downarrow}] = g[v^{\downarrow}/a]$. Since $[a]g \equiv u^{\downarrow}$ and $g[v^{\downarrow}/a] \equiv \mathsf{sub}(u, v)^{\downarrow}$, this is

  $$\vdash_{\mathsf{SIMP}} \mathsf{sub}(u^{\downarrow}, v^{\downarrow}) = \mathsf{sub}(u, v)^{\downarrow}.$$

  Using (**tran**) we conclude $\vdash_{\mathsf{SIMP}} \mathsf{sub}(u, v) = \mathsf{sub}(u, v)^{\downarrow}$ as required.

□

As a corollary of Theorem 4.4.6 the standard properties of capture-avoiding substitution on ground terms from Lemma 4.3.8 carry over to closed terms:

Corollary 4.4.7. *For closed terms t, u, v:*

1. $\vdash_{\mathsf{SIMP}} t[a \mapsto a] = t$.
2. *If* $\vdash b\#t$ *then* $\vdash_{\mathsf{SIMP}} t[a \mapsto b] = (b\ a) \cdot t$.
3. *If* $\vdash a\#t$ *then* $\vdash_{\mathsf{SIMP}} t[a \mapsto u] = t$.
4. *If* $\vdash a\#v$ *then* $\vdash_{\mathsf{SIMP}} t[a \mapsto u][b \mapsto v] = t[b \mapsto v][a \mapsto u[b \mapsto v]]$.

Proof. Using Theorem 4.4.6 and Lemma 4.3.8. For the second case we also use the property that $\vdash_{\mathsf{CORE}} (\pi \cdot t)^{\downarrow} = \pi \cdot t^{\downarrow}$ for closed terms t, which we can show by an induction on t. □

In this section we have established that in the presence of the equalities of SIMP, t and $t^{\downarrow}$ are provably equal, for *closed* terms t. Yet many questions related to *open* terms remain:

1. Is SIMP conservative over CORE? That is, if two terms that do not mention sub are equated in SIMP, are they necessarily equated in CORE?

   (Answer: Yes.)

2. Is equality in SIMP decidable?

   (Answer: Yes.)

3. Is SIMP sound for the ground term model? That is, if two terms are equated in SIMP, are all their closed instances α-equivalent, if we interpret every occurrence of sub by capture-avoiding substitution?

   (Answer: Yes.)

4. Is SIMP complete for the ground term model? That is, if all closed instances of two terms are α-equivalent where we interpret sub by capture-avoiding substitution, are the terms themselves provably equal in SIMP?

   (Answer: No, but a more powerful theory SUB *exists which* is *complete, and which retains properties 1 to 3 of this list.)*

The next two sections provide answers to these questions together with detailed proofs.

4.5 Substitution on Open Terms using SIMP

In this section we recall a notion of rewriting called *nominal rewriting*, which is tailored to nominal terms [FG07]. We will use nominal rewriting to prove properties on open terms of theory SIMP.

4.5.1 Nominal Rewriting

Definition 4.5.1. A **nominal rewrite rule** $\nabla \vdash l \to r$ is a tuple of a freshness context ∇ and terms l and r of the same sort such that ∇ and r mention only unknowns appearing in l.

A **nominal rewrite system** R is a set of nominal rewrite rules. It determines a set of **(nominal) rewrites** $\Delta \vdash_{\mathsf{R}} t \to u$ inductively defined by the rules in Figure 4.4.

Write $\Delta \nvdash_{\mathsf{R}} t \to u$ when the rewrite $\Delta \vdash_{\mathsf{R}} t \to u$ is *not* derivable.

In Figure 4.4, f ranges over *all* term-formers of the signature of R (whatever that signature is). The ($\to$**rew**) rule is closely related to the ($\mathbf{ax}_{\nabla\vdash\mathbf{t}=\mathbf{u}}$) rule from Figure 2.2. We discuss the aspects of this rule in more detail:

- The permutation π allows us to permutatively rename atoms. Consider for instance the signature of theory LAM (Example 4.2.4) extended with a term-former const : $()\mathbb{T}$, and write const() as const. Then the rewrite rule $ab \to \mathsf{const}$ generates, for example, rewrites

$$ba \to \mathsf{const} \qquad \text{and} \qquad ac \to \mathsf{const}$$

$$\frac{\Delta \vdash_{\mathsf{CORE}} t = l^\pi\sigma \quad \Delta \vdash_{\mathsf{CORE}} u = r^\pi\sigma \quad \Delta \vdash \nabla^\pi\sigma}{\Delta \vdash_{\mathsf{R}} t \to u}(\to\mathbf{rew}) \quad (\nabla \vdash l \to r \in \mathsf{R})$$

$$\frac{\Delta \vdash_{\mathsf{R}} t \to u}{\Delta \vdash_{\mathsf{R}} [a]t \to [a]u}(\to[]) \qquad \frac{\Delta \vdash_{\mathsf{R}} t \to u}{\Delta \vdash_{\mathsf{R}} \mathsf{f}(t_1,\dots,t,\dots,t_n) \to \mathsf{f}(t_1,\dots,u,\dots,t_n)}(\to\mathsf{f})$$

Figure 4.4 Derivation rules for nominal rewriting

$$\frac{\Delta \vdash_{\mathsf{R}} t \to u}{\Delta \vdash_{\mathsf{R}} t \to^* u}(\to^*\to) \qquad \frac{\Delta \vdash_{\mathsf{CORE}} t = u}{\Delta \vdash_{\mathsf{R}} t \to^* u}(\to^*\mathbf{refl})$$

$$\frac{\Delta \vdash_{\mathsf{R}} t \to^* u \quad \Delta \vdash_{\mathsf{R}} u \to^* v}{\Delta \vdash_{\mathsf{R}} t \to^* v}(\to^*\mathbf{tran})$$

Figure 4.5 Derivation rules for the transitive reflexive closure of R

but *not*

$$aa \to \mathsf{const}$$

because no π can identify a with b.

- The substitution σ gives unknowns X in rules the character of 'unknown terms', and is subject to the freshness conditions formulated by $\Delta \vdash \nabla^\pi\sigma$.
- The use of equality in CORE gives abstractions $[a]t$ the character of *real* abstractions. For example the rule $\lambda[a]\lambda[b]b \to \mathsf{const}$ generates rewrites

$$\lambda[a]\lambda[a]a \to \mathsf{const} \qquad \lambda[b]\lambda[b]b \to \mathsf{const} \qquad \lambda[b]\lambda[c]c \to \mathsf{const}.$$

The following result is easy to prove from the definition of rewriting:

Lemma 4.5.2. *If* $\Delta \vdash_{\mathsf{R}} t \to u$ *and* $\Delta' \vdash_{\mathsf{R}} \Delta\sigma$*, then* $\Delta' \vdash_{\mathsf{R}} t\sigma \to u\sigma$*.*

Proof. By induction on the derivation of $\Delta \vdash_{\mathsf{R}} t \to u$ we construct a derivation of $\Delta' \vdash_{\mathsf{R}} t\sigma \to u\sigma$. □

Definition 4.5.3. Write $\Delta \vdash_{\mathsf{R}} t \to^* u$ for the **transitive reflexive closure** of R defined inductively by the rules in Figure 4.5.

Write $\Delta \nvdash_{\mathsf{R}} t \to^* u$ when $\Delta \vdash_{\mathsf{R}} t \to^* u$ is *not* derivable.

If a term does not have any rewrites, the transitive reflexive closure of R is precisely CORE-equality:

Lemma 4.5.4. *If there is no t' such that $\Delta \vdash_{\mathsf{R}} t \to t'$ then*

$$\Delta \vdash_{\mathsf{R}} t \to^* u \quad \textit{if and only if} \quad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. Suppose there is no t' such that $\Delta \vdash_{\mathsf{R}} t \to t'$. If $\Delta \vdash_{\mathsf{CORE}} t = u$ then by ($\to^*$**refl**) also $\Delta \vdash_{\mathsf{R}} t \to^* u$. Conversely if $\Delta \vdash_{\mathsf{R}} t \to^* u$ then this rewrite must be derived using ($\to^*$**refl**) by an inductive argument. □

Definition 4.5.5. Call a nominal rewrite system R **confluent** when if $\Delta \vdash_{\mathsf{R}} t \to^* u$ and $\Delta \vdash_{\mathsf{R}} t \to^* v$ then there is some w such that $\Delta \vdash_{\mathsf{R}} u \to^* w$ and $\Delta \vdash_{\mathsf{R}} v \to^* w$.

Definition 4.5.6. Call a nominal rewrite system R **strongly normalising** when there is no infinite sequence $t_1, t_2, t_3, \ldots$ such that $\Delta \vdash_{\mathsf{R}} t_i \to t_{i+1}$ for all $1 \leq i$.

Lemma 4.5.7. *If a derivation exists of $\Delta \vdash_{\mathsf{R}} t \to u$ then that derivation mentions* ($\to$**rew**) *exactly once.*

Proof. By the structure of the derivation rules from Figure 4.4. □

Definition 4.5.8. If a rewrite $\Delta \vdash_{\mathsf{R}} t \to u$ occurs, it must occur *at* some subterm t' of t (the subterm t' where we actually use ($\to$**rew**) and prove $\Delta \vdash \nabla^\pi\sigma$ and $\Delta \vdash_{\mathsf{CORE}} t' = l^\pi\sigma$). We say that the rewrite **occurs at** t' **in** t. If the derivation tree is just an instance of ($\to$**rew**), then we say the rewrite **occurs at top level**.

Call a pair of nominal rewrites $\Delta \vdash_{\mathsf{R}} t \to u$ and $\Delta \vdash_{\mathsf{R}} t \to v$ a **critical pair** when at least one of the rewrites occurs at top level. If any of the two rewrites occurs at a moderated unknown in t, call the critical pair **trivial**. Otherwise call it **non-trivial**.

Definition 4.5.9. Call a rewrite rule $\nabla \vdash l \to r$ **uniform** when $\Delta \vdash a\#l$ and $\Delta \vdash \nabla$ imply $\Delta \vdash a\#r$ for all Δ and a. A rewrite rule $\nabla \vdash l \to r$ is **left-linear** when l does not mention the same unknown more than once. A uniform nominal rewrite system with only left-linear rules and no non-trivial critical pairs is **orthogonal**.

Theorem 4.5.10. *An orthogonal uniform nominal rewrite system is confluent.*

Proof. See [FG07, Theorem 65]. □

Remark 4.5.11. Nominal rewriting is the default notion of rewriting for nominal terms, like higher-order rewriting (such as CRS's [KvOvR93] and HRS's [MN98]) is for higher-order terms. The major differences between the two frameworks can be summarised as follows:

- The default notion of instantiation of meta-variables is capturing for nominal rewriting whereas it is capture-avoiding for higher-order rewriting.
- Nominal rewriting uses unification up to α whereas higher-order rewriting uses unification up $\alpha\beta(\eta)$.

A more detailed comparison of these frameworks can be found in [FG07].

$$\boxed{\begin{array}{rlrcl}
(\mathbf{Rvar}) & \vdash & a[a \mapsto T] & \to & T \\
(\mathbf{Rb}) & \vdash & b[a \mapsto T] & \to & b \\
(\mathbf{Rf}) & \vdash & \mathsf{f}(X_1, \ldots, X_n)[a \mapsto T] & \to & \mathsf{f}(X_1[a \mapsto T], \ldots, X_n[a \mapsto T]) \ \ (\mathsf{f} \neq \mathsf{sub}) \\
(\mathbf{Rabs}) & c\#T \vdash & ([c]U)[a \mapsto T] & \to & [c](U[a \mapsto T])
\end{array}}$$

Figure 4.6 Substitution as a rewrite system SIMPr

4.5.2 SIMPr: Explicit Substitution Rewritten

Definition 4.5.12. The nominal rewrite system SIMPr is defined by the rules in Figure 4.6.

Remark 4.5.13. The (**Rb**) rule cannot be represented by a rule in a higher-order rewrite system, since in such a system object-variables only exist when they are bound by a meta-level abstraction. That is, the rule $\mathsf{sub}(\lambda x.y, T) \to y$ does *not* represent (**Rb**) since y represents a meta-variable instead of an object-variable. It represents the more general rule $a\#X \vdash X[a \mapsto T] \to X$.

A basic correctness result is this:

Theorem 4.5.14 (SIMPr implies SIMP)**.** *For possibly open terms t, u:*

$$\textit{if} \quad \Delta \vdash_{\mathsf{SIMPr}} t \to u \quad \textit{then} \quad \Delta \vdash_{\mathsf{SIMP}} t = u.$$

Proof. We work by induction on the derivation of $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ to show that $\Delta \vdash_{\mathsf{SIMP}} t = u$ is derivable.

If the derivation concludes in ($\to$[]) or ($\to$f) we may use the inductive hypothesis and extend the derivation with (**cong**[]) or (**cong**f) respectively.

Suppose the derivation concludes in ($\to$**rew**). Then there are various cases depending on which rewrite rule is used:

- (**Rvar**). $\Delta \vdash_{\mathsf{SIMP}} a[a \mapsto u] = u$ is derivable using axiom (**var**$\mapsto$).
- (**Rb**). $\Delta \vdash_{\mathsf{SIMP}} b[a \mapsto t] = b$ is derivable using axiom (**b**$\mapsto$).
- (**Rf**). $\Delta \vdash_{\mathsf{SIMP}} \mathsf{f}(u_1, \ldots, u_n)[a \mapsto t] = \mathsf{f}(u_1[a \mapsto u], \ldots, u_n[a \mapsto u])$ is derivable using axiom (f$\mapsto$).
- (**Rabs**). $\Delta \vdash_{\mathsf{SIMP}} ([b]u)[a \mapsto t] = [b](u[a \mapsto t])$ is derivable using (**abs**$\mapsto$); the freshness side-condition of (**Rabs**) is guaranteed by the identical condition on (**abs**$\to$).

The result follows. □

Corollary 4.5.15. *For possibly open terms t, u:*

$$\textit{if} \quad \Delta \vdash_{\mathsf{SIMPr}} t \to^* u \quad \textit{then} \quad \Delta \vdash_{\mathsf{SIMP}} t = u.$$

Proof. By induction on the structure of derivations of $\Delta \vdash_{\mathsf{SIMPr}} t \to^* u$. The case of ($\to^*\to$) uses Theorem 4.5.14, ($\to^*$**refl**) uses (**refl**), and ($\to^*$**tran**) uses (**tran**). □

4.5.3 Confluence, Conservativity and Consistency

Lemma 4.5.16. *All rewrite rules of* SIMPr *are uniform.*

Proof. For each rule, use appropriate instances of Lemma 2.4.2. For example, for the (**Rabs**) rule, we need to show that $\Delta \vdash a'\#([c]U)[a \mapsto T]$ and $\Delta \vdash c\#T$ imply $\Delta \vdash a'\#[c](U[a \mapsto T])$ for any a' and Δ. From the first assumption we know, by Lemma 2.4.2:

- if $a' = a$ or $a' = c$ then $a'\#T \in \Delta$;
- if $a' \neq a$ and $a' \neq c$ then $a'\#T \in \Delta$ and $a'\#U \in \Delta$.

Using these assumptions it is easy to show $\Delta \vdash a'\#[c](U[a \mapsto T])$ by case distinction on a'. □

Theorem 4.5.17 (Confluence of SIMPr)**.** SIMPr *is confluent.*

Proof. By Lemma 4.5.16 all rewrite rules of SIMPr are uniform. Also, SIMPr has no non-trivial critical pairs and every rule is left-linear (each unknown is mentioned on the left at most once). Then SIMPr is orthogonal and uniform by Definition 4.5.9. We conclude that it is confluent by Theorem 4.5.10. □

Confluence of SIMPr has a number of nice corollaries, which comprise the remainder of this subsection.

Definition 4.5.18. Call a term t a **SIMPr-normal form with respect to Δ** when there is no u such that $\Delta \vdash_{\mathsf{SIMPr}} t \rightarrow u$.

Theorem 4.5.19. *Suppose that t and u are* SIMPr*-normal forms with respect to Δ. Then*

$$\Delta \vdash_{\mathsf{SIMP}} t = u \qquad \textit{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. The (empty) set of axioms of CORE is a subset of the axioms of SIMP so a derivation in CORE is also a derivation in SIMP and it follows that $\Delta \vdash_{\mathsf{CORE}} t = u$ implies $\Delta \vdash_{\mathsf{SIMP}} t = u$.

Conversely suppose $\Delta \vdash_{\mathsf{SIMP}} t = u$. By Theorem 4.5.17 there is some term v such that $\Delta \vdash_{\mathsf{SIMPr}} t \rightarrow^* v$ and $\Delta \vdash_{\mathsf{SIMPr}} u \rightarrow^* v$. By assumption there can be no t' and u' such that $\Delta \vdash_{\mathsf{SIMPr}} t \rightarrow t'$ and $\Delta \vdash_{\mathsf{SIMPr}} u \rightarrow u'$. Then $\Delta \vdash_{\mathsf{CORE}} t = v$ and $\Delta \vdash_{\mathsf{CORE}} u = v$ by Lemma 4.5.4, and we conclude $\Delta \vdash_{\mathsf{CORE}} t = u$ by (**symm**) and (**tran**). □

Corollary 4.5.20 (Conservativity of SIMP over CORE)**.** *Suppose that t and u do* not *mention the term-former* sub*. Then*

$$\Delta \vdash_{\mathsf{SIMP}} t = u \qquad \textit{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. This is an instance of Theorem 4.5.19, since if t and u do not mention sub, then t and u are SIMPr-normal forms with respect to any Δ. □

Recall the notation $t^{\mathbb{I}}$ for closed terms t from Definition 4.4.4.

Corollary 4.5.21. *For closed terms t and u,*

$$\vdash_{\mathsf{SIMP}} t = u \quad \textit{if and only if} \quad \vdash_{\mathsf{CORE}} t^{\mathbb{I}} = u^{\mathbb{I}}.$$

Proof. $\vdash_{\mathsf{SIMP}} t = u$ is equivalent to $\vdash_{\mathsf{SIMP}} t^{\mathbb{I}} = u^{\mathbb{I}}$ by Theorem 4.4.6 and (**tran**). By Corollary 4.5.20 this is equivalent to $\vdash_{\mathsf{CORE}} t^{\mathbb{I}} = u^{\mathbb{I}}$. □

Corollary 4.5.22 (Consistency of SIMP)**.** *For all Δ there are t and u such that $\Delta \nvdash_{\mathsf{SIMP}} t = u$.*

Proof. By the syntactic criteria for CORE-equality (Corollary 2.5.4), we know $\Delta \nvdash_{\mathsf{CORE}} a = b$. This is equivalent to $\Delta \nvdash_{\mathsf{SIMP}} a = b$ by conservativity of SIMP over CORE (Corollary 4.5.20). □

For the final theorem of this subsection we need a few definitions.

Definition 4.5.23. Call a substitution σ **closing for an unknown** X when $\sigma(X)$ is a closed term. Call σ **closing for a term** t or **closing for a freshness context** Δ when $\sigma(X)$ is closed for every $X \in t$ or $X \in \Delta$.

Say a closing substitution σ for Δ is **Δ-consistent** when $\vdash \Delta\sigma$, i.e. when $\vdash a\#\sigma(X)$ for all $a\#X \in \Delta$.

Theorem 4.5.24 (Soundness of SIMP)**.** *Theory* SIMP *is sound for the ground term model. That is: if $\Delta \vdash_{\mathsf{SIMP}} t = u$ then $t\sigma^{\mathbb{I}} =_\alpha u\sigma^{\mathbb{I}}$ for all Δ-consistent closing substitutions σ (for Δ, t and u).*

Proof. $\vdash_{\mathsf{SIMP}} t\sigma = u\sigma$ by meta-level substitution (Theorem 2.4.10) and our assumption that $\vdash \Delta\sigma$. We obtain $\vdash_{\mathsf{CORE}} t\sigma^{\mathbb{I}} = u\sigma^{\mathbb{I}}$ by Corollary 4.5.21, since $t\sigma$ and $u\sigma$ are closed. We conclude $t\sigma^{\mathbb{I}} =_\alpha u\sigma^{\mathbb{I}}$ by Theorem 4.3.13, since $t\sigma^{\mathbb{I}}$ and $u\sigma^{\mathbb{I}}$ are ground. □

4.5.4 Failure of Completeness for SIMP

SIMP is not a *complete* theory of substitution on ground terms. Unknowns in nominal algebra represent unknown terms, and here are some examples of statements that are true for every closing σ (for the unknowns in the statements) but *not* derivable in SIMP:

Theorem 4.5.25 (Incompleteness of SIMP)**.**

1. $\nvdash_{\mathsf{SIMP}} X[a \mapsto a] = X$.
2. $b\#X \nvdash_{\mathsf{SIMP}} X[a \mapsto b] = (b\ a) \cdot X$.
3. $a\#X \nvdash_{\mathsf{SIMP}} X[a \mapsto T] = X$.
4. $a\#U \nvdash_{\mathsf{SIMP}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]]$.

Proof. We consider part 4. We can easily check using the syntactic criteria of CORE-equality (Corollary 2.5.4) that

$$a\#U \nvdash_{\mathsf{CORE}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]].$$

Since $X[a \mapsto T][b \mapsto U]$ and $X[b \mapsto U][a \mapsto T[b \mapsto U]]$ have no SIMPr rewrites with respect to $a\#U$, we conclude

$$a\#U \nvdash_{\mathsf{SIMP}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]]$$

by Theorem 4.5.19.

The proofs of the other parts are similar. □

The fact that above assertions are derivable for closing substitutions follows by Corollary 4.4.7.

So SIMP defines substitution, but it does not express all of the properties which emerge from that definition. To do that we must strengthen the theory. Before that however, it is useful to consider the computational content of SIMP.

4.5.5 Strong Normalisation and Decidability

We expect the part of a λ-calculus that handles substitution to be terminating [BR95, Les94] — so is SIMP terminating? After all our syntax contains sub as an explicit term-former, and it contains unknowns so that sub cannot always be completely eliminated. Also, we consider single substitutions and not *simultaneous* substitutions, so that the order of substitutions matters. Perhaps that all makes enough of a difference that reductions could cycle or diverge in some way?

In fact reductions in SIMPr are extremely well-behaved. We show strong normalisation by a standard method: define a well-founded measure on terms and show that rewrites reduce it.

Definition 4.5.26. Let the measure $|t|_m$ on terms t be inductively defined by:

$$|a|_m = 1 \qquad |\pi \cdot X|_m = 1 \qquad |[a]t|_m = |t|_m + 1$$
$$|\mathsf{sub}(t,u)|_m = |t|_m * (|u|_m + 1)$$
$$|\mathsf{f}(t_1,\ldots,t_n)|_m = |t_1|_m + \ldots + |t_n|_m + n + 1 \qquad (\mathsf{f} \neq \mathsf{sub}).$$

For terms $u[a \mapsto t]$ we have

$$|u[a \mapsto t]|_m \equiv |\mathsf{sub}([a]u,t)|_m = |[a]u|_m * (|t|_m + 1) = (|u|_m + 1) * (|t|_m + 1).$$

Lemma 4.5.27. *For all terms t and permutations π:*

1. $|t|_m > 0$.
2. $|t|_m = |\pi \cdot t|_m$. *As a corollary, if $\Delta \vdash_{\mathsf{CORE}} t = u$ then $|t|_m = |u|_m$.*

Proof. Both parts can be proven by a simple induction on the structure of t. The corollary follows by the syntactic criteria for CORE-equality (Corollary 2.5.4) which states that t and u are renamed versions of each other by means of permutations. □

Lemma 4.5.28. *For terms $t, u, t_1, \dots, t_n$ and term-formers $\mathsf{f} \neq \mathsf{sub}$:*

1. $|a[a \mapsto t]|_m > |t|_m$.
2. $|b[a \mapsto t]|_m > |b|_m$.
3. $|\mathsf{f}(t_1, \dots, t_n)[a \mapsto t]|_m > |\mathsf{f}(t_1[a \mapsto t], \dots, t_n[a \mapsto t])|_m$.
4. $|([c]u)[a \mapsto t]|_m > |[c](u[a \mapsto t])|_m$.

Proof. By straightforward calculations. The last part uses the fact that $|t|_m > 0$ (Lemma 4.5.27 above). □

Lemma 4.5.29. *For terms $t, u, t_1, \dots, t_n$ and term-formers f:*

1. *If* $|t|_m > |u|_m$ *then* $|[a]t|_m > |[a]u|_m$.
2. *If* $|t|_m > |u|_m$ *then* $|\mathsf{f}(t_1, \dots, t, \dots, t_n)|_m > |\mathsf{f}(t_1, \dots, u, \dots, t_n)|_m$.

Proof. Again by straightforward calculations. The last part uses the fact that $|t|_m > 0$ when $\mathsf{f} = \mathsf{sub}$. □

Theorem 4.5.30 (Strong normalisation of SIMPr)**.** SIMPr *is strongly normalising.*

Proof. It suffices to show that if $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ then $|t|_m > |u|_m$. We proceed by induction on the rules from Figure 4.4, using the rewrite rules from Figure 4.6.

Suppose $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ is derived using ($\to$[]) and ($\to$f); then the result follows by Lemma 4.5.29 and the inductive hypothesis.

Suppose $\Delta \vdash_{\mathsf{SIMPr}} t \to u$ is derived using ($\to$**rew**); then for each SIMPr rewrite rule of the form $\nabla \vdash l \to r$ from Figure 4.6 we have, for some π and σ,

$$\Delta \vdash_{\mathsf{CORE}} t = \pi \cdot l\sigma \qquad \text{and} \qquad \Delta \vdash_{\mathsf{CORE}} u = \pi \cdot r\sigma \qquad \text{and} \qquad \Delta \vdash \pi \cdot \nabla\sigma.$$

By part 2 of Lemma 4.5.27, $|t|_m = |\pi \cdot l\sigma|_m = |l\sigma|_m$ and $|u|_m = |\pi \cdot r\sigma|_m = |r\sigma|_m$. So in order to show $|t|_m > |u|_m$, it is equivalent to show $|l\sigma|_m > |r\sigma|_m$. For each rule of SIMPr this is an instance of Lemma 4.5.28. □

We have already given an algorithm to compute normal forms on closed terms in Definition 4.4.4:

Lemma 4.5.31. *If t is closed then $t^{\mathbb{I}}$ is a* SIMPr*-normal form of t.*

Proof. By an induction on t. □

Theorem 4.5.32 (Unique normal forms for SIMPr)**.** SIMPr*-normal forms are unique up to equality in* CORE.

Proof. Let Δ be a freshness context and t be a term. By Theorem 4.5.30, t has a normal form, say u, with respect to Δ. Now suppose v is also a normal form of t with respect to Δ. Then $\Delta \vdash_{\mathsf{SIMPr}} t \to^* u$ and $\Delta \vdash_{\mathsf{SIMPr}} t \to^* v$. By confluence (Theorem 4.5.17) there exists a term w such that $\Delta \vdash_{\mathsf{SIMPr}} u \to^* w$ and $\Delta \vdash_{\mathsf{SIMPr}} v \to^* w$. Since u and v do not have any rewrites $\Delta \vdash_{\mathsf{CORE}} u = w$ and $\Delta \vdash_{\mathsf{CORE}} v = w$ by Lemma 4.5.4. Using (**symm**) and (**tran**) we conclude $\Delta \vdash_{\mathsf{CORE}} u = v$. $\square$

As a corollary we obtain decidability of theory SIMP:

Corollary 4.5.33 (Decidability of SIMP)**.** *It is decidable whether* $\Delta \vdash_{\mathsf{SIMP}} t = u$.

Proof. Given Δ, t and u the following procedure decides whether $t = u$ is derivable from Δ in SIMP:

1. Rewrite t and u to SIMPr-normal forms t' and u' with respect to Δ; by Theorem 4.5.30 these exist and by Theorem 4.5.32, they are unique up to equality in CORE.
2. Check whether $\Delta \vdash_{\mathsf{CORE}} t' = u'$ using the syntactic criteria of Corollary 2.5.4.
3. If $\Delta \vdash_{\mathsf{CORE}} t' = u'$ then return ‘true’, otherwise return ‘false’.

$\square$

4.6 Substitution on Open Terms using SUB

In this section we focus on theory SUB from Definition 4.2.8 and Figure 4.1. We show that it is decidable and complete with respect to the ground term model by relating it to theory SIMP.

As a first example, we show that our standard properties of capture-avoiding substitution are all derivable in SUB.

Lemma 4.6.1. *The following judgements are derivable in* SUB*:*

1. $\vdash_{\mathsf{SUB}} X[a \mapsto a] = X$.
2. $b\#X \vdash_{\mathsf{SUB}} X[a \mapsto b] = (b\ a) \cdot X$.
3. $a\#X \vdash_{\mathsf{SUB}} X[a \mapsto T] = X$.
4. $a\#U \vdash_{\mathsf{SUB}} X[a \mapsto T][b \mapsto U] = X[b \mapsto U][a \mapsto T[b \mapsto U]]$.

Proof. Part 1 is Lemma 2.3.19. Parts 2 and 3 are direct from (**ren**$\mapsto$) and ($\#\mapsto$). For part 4 we give the derivation in full, writing $\mathfrak{s}$ for $[b \mapsto U]$ and using the unsugared syntax for the other substitutions:

$$\dfrac{\dfrac{}{\mathsf{sub}([a]X,T)\mathfrak{s} = \mathsf{sub}(([a]X)\mathfrak{s}, T\mathfrak{s})}(\mathbf{ax}_{\mathsf{f}\mapsto}) \qquad \dfrac{\dfrac{a\#U}{([a]X)\mathfrak{s} = [a](X\mathfrak{s})}(\mathbf{ax}_{\mathbf{abs}\mapsto})}{\mathsf{sub}(([a]X)\mathfrak{s}, T\mathfrak{s}) = \mathsf{sub}([a](X\mathfrak{s}), T\mathfrak{s})}(\mathbf{congf})}{\mathsf{sub}([a]X,T)\mathfrak{s} = \mathsf{sub}([a](X\mathfrak{s}), T\mathfrak{s})}(\mathbf{tran})$$

$\square$

4.6.1 Soundness and the Relation to SIMP

SUB can do everything that SIMP can:

Lemma 4.6.2. *If* $\Delta \vdash_{\mathsf{SIMP}} t = u$ *then* $\Delta \vdash_{\mathsf{SUB}} t = u$.

Proof. All the axioms of SIMP are also axioms of SUB, except for $(\mathbf{b}{\mapsto})$. However an instance of $(\mathbf{b}{\mapsto})$ is *also* an instance of $(\#{\mapsto})$. Therefore any SIMP derivation is also a SUB derivation. The result follows. $\square$

We now show that SIMP can do everything that SUB can — provided that the terms are closed. We need a technical lemma:

Lemma 4.6.3. *If* t, u, v *are closed, then:*

1. $\vdash_{\mathsf{SIMP}} \mathsf{sub}(t,u)[a \mapsto v] = \mathsf{sub}(t[a \mapsto v], u[a \mapsto v])$.
2. *If* $\vdash a\#t$ *then* $\vdash_{\mathsf{SIMP}} [a]\mathsf{sub}(t,a) = t$.

Proof. We consider the parts in turn:

1. Since $\vdash_{\mathsf{SIMP}} t = t^{\mathbb{I}}$ by Theorem 4.4.6, the proof obligation is equivalent to

$$\vdash_{\mathsf{SIMP}} \mathsf{sub}(t^{\mathbb{I}},u)[a \mapsto v] = \mathsf{sub}(t^{\mathbb{I}}[a \mapsto v], u[a \mapsto v]).$$

   Now $t^{\mathbb{I}}$ is of the form $[a]g$ or $[b]g$, where g is a ground term. We only consider the case of b, the case of a is completely analogous. Take c fresh such that $\vdash c\#g$ and $\vdash c\#v$. Then $\vdash_{\mathsf{CORE}} [b]g = [c](c\ b)\cdot g$ using (**perm**), since $\vdash b\#[b]g$ and $\vdash c\#[b]g$. Then using the rules for equality, the proof obligation is equivalent to

$$\vdash_{\mathsf{SIMP}} \mathsf{sub}([c](c\ b)\cdot g, u)[a \mapsto v] = \mathsf{sub}(([c](c\ b)\cdot g)[a \mapsto v], u[a \mapsto v]).$$

   Also $\vdash_{\mathsf{SIMP}} ([c](c\ b)\cdot g)[a \mapsto v] = [c](((c\ b)\cdot g)[a \mapsto v])$ by axiom $(\mathbf{abs}{\mapsto})$ and $\vdash c\#v$, so the proof obligation is equivalent to

$$\vdash_{\mathsf{SIMP}} \mathsf{sub}([c](c\ b)\cdot g, u)[a \mapsto v] = \mathsf{sub}([c](((c\ b)\cdot g)[a \mapsto v]), u[a \mapsto v]).$$

   Or, using sugar:

$$\vdash_{\mathsf{SIMP}} ((c\ b)\cdot g)[c \mapsto u][a \mapsto v] = ((c\ b)\cdot g)[a \mapsto v][c \mapsto u[a \mapsto v]].$$

   Since $\vdash c\#v$, this is an instance of part 4 of Corollary 4.4.7, so we are done.

2. We must show $\vdash_{\mathsf{SIMP}} [a]\mathsf{sub}(t,a) = t$. By Theorem 4.4.6 this is equivalent to $\vdash_{\mathsf{SIMP}} [a]\mathsf{sub}(t^{\mathbb{I}},a) = t^{\mathbb{I}}$. We proceed by case distinction on the structure of $t^{\mathbb{I}}$; here we only consider the two most interesting cases:

   - $t^{\mathbb{I}} \equiv [a]g$: Then $\vdash_{\mathsf{SIMP}} [a](g[a \mapsto a]) = [a]g$ follows by $(\mathbf{cong}[])$ and part 1 of Corollary 4.4.7.

- $t^{\mathtt{I}} \equiv [b]g$: Then $\vdash_{\mathsf{SIMP}} [a](g[b \mapsto a]) = [b]g$ follows from

  $$\vdash_{\mathsf{SIMP}} [a](g[b \mapsto a]) = [a](a\ b) \cdot g \quad \text{and} \quad \vdash_{\mathsf{SIMP}} [a](a\ b) \cdot g = [b]g.$$

  by (**tran**). By (**cong**[]), $\vdash_{\mathsf{SIMP}} [a](g[b \mapsto a]) = [a](b\ a) \cdot g$ follows from $\vdash_{\mathsf{SIMP}} g[b \mapsto a] = (a\ b) \cdot g$. By Corollary 4.4.7, this is when $\vdash a\#g$. By (**perm**) and the rules for freshness also $\vdash_{\mathsf{SIMP}} [a](b\ a) \cdot g = [b]g$ when $\vdash a\#g$. Now by Lemma 2.4.2 $\vdash a\#g$ when $\vdash a\#[b]g$. Since $[b]g \equiv t^{\mathtt{I}}$, this follows from the assumption $\vdash a\#t$ by Lemma 4.4.5.

□

On closed terms, SUB is equivalent to SIMP:

Lemma 4.6.4 (SUB on closed terms)**.** *For closed terms t, u:*

$$\vdash_{\mathsf{SUB}} t = u \quad \textit{if and only if} \quad \vdash_{\mathsf{SIMP}} t = u.$$

Proof. The right-to-left part is Lemma 4.6.2.

For the left-to-right part, we show that SIMP can simulate the axioms of SUB on closed terms. Axioms (**var**$\mapsto$), (**abs**$\mapsto$) and ($\mathsf{f}\mapsto$), for $\mathsf{f} \neq \mathsf{sub}$, are also present in SIMP. Each instance of axiom ($\mathsf{f}\mapsto$), where $\mathsf{f} = \mathsf{sub}$, is an instance of part 1 of Lemma 4.6.3. Instances of axioms (**ren**$\mapsto$) and ($\#\mapsto$) are instances of parts 2 and 3 of Corollary 4.4.7. Finally, each instance of ($\eta\mapsto$) is an instance of part 2 of Lemma 4.6.3. □

To recap, SIMP is sound for a ground term model by Theorem 4.4.3, equality in SIMP is decidable by Corollary 4.5.33, but not complete by the Theorem 4.5.25.

We now build the tools to prove that SUB is sound, decidable — and also complete for the ground term model. For soundness we have already done all the hard work:

Theorem 4.6.5 (Soundness of SUB)**.** SUB *is sound for the ground term model. That is: If $\Delta \vdash_{\mathsf{SUB}} t = u$ then $t\sigma^{\mathtt{I}} =_\alpha u\sigma^{\mathtt{I}}$ for all Δ-consistent closing substitutions σ (for Δ, t and u).*

Proof. $\vdash_{\mathsf{SUB}} t\sigma = u\sigma$ by meta-level substitution (Theorem 2.4.10) and our assumptions. $t\sigma$ and $u\sigma$ are closed so by Lemma 4.6.4 $\vdash_{\mathsf{SIMP}} t\sigma = u\sigma$. By Corollary 4.5.21 we obtain $\vdash_{\mathsf{CORE}} t\sigma^{\mathtt{I}} = u\sigma^{\mathtt{I}}$. We conclude $t\sigma^{\mathtt{I}} =_\alpha u\sigma^{\mathtt{I}}$ by Theorem 4.3.13, since $t\sigma^{\mathtt{I}}$ and $u\sigma^{\mathtt{I}}$ are ground. □

4.6.2 Decidability

In this section we will establish that equality in SUB is decidable. We do this by transforming the problem of deciding whether a derivation of $\Delta \vdash_{\mathsf{SUB}} t = u$ exists, into the problem of deciding whether a derivation of $\vdash_{\mathsf{SIMP}} t' = u'$ exists, for some carefully-chosen closed terms t' and u' in an extended signature (to be precise: in a correspondingly extended theory with extra ($\mathsf{f}\mapsto$) axioms for the extra term-formers). We can then exploit decidability of SIMP (Corollary 4.5.33), and conclude that SUB is decidable. The rest of this subsection makes this formal.

Remark 4.6.6. The constructions are similar to those in the completeness proof of Chapter 3 (Subsection 3.4.2). However, the situation is a bit more complicated: in the completeness proof of Chapter 3 we only needed to take care of permutations of atoms, but here we also have to deal with capture-avoiding substitution of terms for atoms.

First we introduce some notation.

Definition 4.6.7. Fix some signature Σ suitable for SUB, and fix Δ, t, and u where t and u are terms in the signature Σ. Let $\mathcal{A}$ be the atoms mentioned anywhere in Δ, t, or u, and let $\mathcal{X}$ be the unknowns mentioned anywhere in Δ, t, or u. For each $X \in \mathcal{X}$:

- let $a_{X1}, \dots, a_{Xk_X}$ be the atoms in $\mathcal{A}$ (in some arbitrary but fixed order) such that $a_{Xi}\#X \notin \Delta$;
- pick some fresh term-former $\mathsf{d}_X : (\mathbb{T}, \dots, \mathbb{T})\mathbb{T}$ (so d_X does not occur in Σ) with k_X arguments.

Write $\mathcal{D} = \{\mathsf{d}_X \mid X \in \mathcal{X}\}$, and let Σ' be the signature $\Sigma \cup \mathcal{D}$.

Definition 4.6.8. Define a substitution ς by:

$$\begin{array}{rcll} \varsigma(X) & = & \mathsf{d}_X(a_{X1}, \dots, a_{Xk_X}) & (X \in \mathcal{X},\ X : \mathbb{T}) \\ \varsigma(X) & = & [a'_X]\mathsf{d}_X(a_{X1}, \dots, a_{Xk_X}) & (X \in \mathcal{X},\ X : [\mathbb{A}]\mathbb{T}) \\ \varsigma(X) & = & X & (X \notin \mathcal{X}) \end{array}$$

Here for each $X \in \mathcal{X}$, a'_X is an arbitrarily chosen fresh atom.

Note that ς maps unknowns to terms in Σ'. Furthermore, if v is term that mentions only unknowns from $\mathcal{X}$, then v is *closed.*
ς is Δ-consistent:

Lemma 4.6.9. $\vdash \Delta\varsigma$ *is derivable.*

Proof. For each $a\#X \in \Delta$, we need to show $\vdash a\#\varsigma(X)$.
Suppose $a\#X \in \Delta$ and $X : \mathbb{T}$. We must prove $\vdash a\#\mathsf{d}_X(a_{X1}, \dots, a_{Xk_X})$. By construction $a_{Xi}\#X \notin \Delta$, so $a \notin \{a_{X1}, \dots, a_{Xk_X}\}$, and the result follows.
The case of $X : [\mathbb{A}]\mathbb{T}$ is similar. □

Now it is not hard to show that ς preserves derivability:

Theorem 4.6.10. *If* $\Delta \vdash_{\mathsf{SUB}} t = u$ *then* $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$.

Proof. Suppose $\Delta \vdash_{\mathsf{SUB}} t = u$ in the syntax of Σ. Then also $\Delta \vdash_{\mathsf{SUB}} t = u$ in the syntax of Σ', since $\Sigma \subseteq \Sigma'$. We obtain $\vdash_{\mathsf{SUB}} t\varsigma = u\varsigma$ by meta-level substitution (Theorem 2.4.10) and Lemma 4.6.9. We conclude $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ by Lemma 4.6.4. □

Since t and u mention only unknowns from $\mathcal{X}$, $t\varsigma$ and $u\varsigma$ are closed terms. Then by Lemma 4.5.31, $t\varsigma^{\downarrow}$ and $u\varsigma^{\downarrow}$ are the SIMPr-normal forms of $t\varsigma$ and $u\varsigma$.

Definition 4.6.11. Let $\mathcal{A}^+$ be the set of *all* atoms mentioned anywhere in the chain of SIMPr-reductions

$$t\varsigma \equiv t'_1 \to t'_2 \to \ldots \to t'_m \equiv t\varsigma^{\downarrow} \quad \text{and} \quad u\varsigma \equiv u'_1 \to u'_2 \to \ldots \to u'_n \equiv u\varsigma^{\downarrow},$$

extended with

- a set $\mathcal{B} = \{b_{Xi} \mid X, i \text{ such that } a_{Xi} \in \mathcal{A}\}$ of fresh atoms in bijection with $\mathcal{A}$;
- a set $\mathcal{C} = \{c_X \mid X \in \mathcal{X}\}$ of fresh atoms.

Let Δ^+ be Δ enriched with freshness assumptions $a'\#X$ for every $a' \in \mathcal{A}^+ \setminus \mathcal{A}$ and every $X \in \mathcal{X}$.

Definition 4.6.12. Let t' and u' range over closed terms in Σ' mentioning only atoms in $\mathcal{A}^+ \setminus (\mathcal{B} \cup \mathcal{C})$.

Define an **inverse translation** from these closed terms in Σ' to terms in Σ, inductively as follows, where $X : \mathbb{T}$ and $Y : [\mathbb{A}]\mathbb{T}$:

$$a'^{-1} \equiv a' \qquad ([a']t')^{-1} \equiv [a'](t'^{-1}) \qquad \mathsf{f}(t'_1, \ldots, t'_n)^{-1} \equiv \mathsf{f}(t'^{-1}_1, \ldots, t'^{-1}_n) \quad (\mathsf{f} \notin \mathcal{D})$$
$$\mathsf{d}_X(t'_1, \ldots, t'_{k_X})^{-1} \equiv ((b_{Xk_X}\ a_{Xk_X}) \cdots (b_{X1}\ a_{X1}) \cdot X)[b_{X1} \mapsto t'^{-1}_1] \cdots [b_{Xk_X} \mapsto t'^{-1}_{k_X}]$$
$$\mathsf{d}_Y(t'_1, \ldots, t'_{k_Y})^{-1} \equiv \mathsf{sub}(((b_{Yk_Y}\ a_{Yk_Y}) \cdots (b_{Y1}\ a_{Y1}) \cdot Y)[b_{Y1} \mapsto t'^{-1}_1] \cdots [b_{Yk_Y} \mapsto t'^{-1}_{k_Y}], c_Y)$$

The importance of t' and u' is that they include all of the t'_i and u'_i in the two chains of SIMPr-reductions mentioned above.

The following lemma shows that -^{-1} is the inverse of ς (regarding t and u):

Lemma 4.6.13. $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} = t$*, and* $\Delta^+ \vdash_{\mathsf{SUB}} (u\varsigma)^{-1} = u$.

Proof. We prove $\Delta^+ \vdash_{\mathsf{SUB}} (v\varsigma)^{-1} = v$ for any subterm v of t or u, by induction on the structure of v.

The only interesting case is when $v \equiv \pi \cdot X$. When $X : \mathbb{T}$, we must show

$$\Delta^+ \vdash_{\mathsf{SUB}} \pi \cdot X = ((b_{X1}\ a_{X1}) \cdots (b_{Xk_X}\ a_{Xk_X}) \cdot X)[b_{X1} \mapsto \pi(a_{X1})] \cdots [b_{Xk_X} \mapsto \pi(a_{Xk_X})].$$

Take $\pi' = (b_{Xk_X}\ \pi(a_{Xk_X})) \cdots (b_{X1}\ \pi(a_{X1})) \circ (b_{X1}\ a_{X1}) \cdots (b_{Xk_X}\ a_{Xk_X})$. Then the proof obligation follows from $\Delta^+ \vdash_{\mathsf{SUB}} \pi \cdot X = \pi' \cdot X$ by $(\mathbf{ren}{\mapsto})$, since

$$\Delta^+ \vdash \pi(a_{Xi})\#((b_{X1}\ a_{X1}) \cdots (b_{Xk_X}\ a_{Xk_X}) \cdot X)[b_{X1} \mapsto \pi(a_{X1})] \cdots [b_{Xi-1} \mapsto \pi(a_{Xi-1})]$$

for all i. We can see this as follows: it suffices to show

$$\Delta^+ \vdash \pi(a_{Xi})\#(b_{X1}\ a_{X1}) \cdots (b_{Xk_X}\ a_{Xk_X}) \cdot X,$$

since the $\pi(a_{Xi})$ are pairwise disjoint and by using the rules for freshnesses. Then there are two possibilities:

- $\pi(a_{Xi}) \neq a_{Xj}$ for all j: then $\pi(a_{Xi})\#X \in \Delta^+$ since $\pi(a_{Xi})\#X \in \Delta$.
- $\pi(a_{Xi}) = a_{Xj}$ for some j: then $b_{Xj}\#X \in \Delta^+$ by definition.

The remaining proof obligation is

$$\Delta^+ \vdash_{\mathsf{SUB}} \pi \cdot X = \pi' \cdot X.$$

It is convenient to show the stronger property $\Delta^+ \vdash_{\mathsf{CORE}} \pi \cdot X = \pi' \cdot X$. By the syntactic criteria of Corollary 2.5.4 we need only show that $\Delta^+ \vdash \mathrm{ds}(\pi, \pi')\#X$. That is, we must show that $\Delta^+ \vdash \mathrm{ds}(\pi, \pi')\#X$ for every a such that $\pi(a) \neq \pi'(a)$. We consider every possible a (every $a \in \pi$ and $a \in \pi'$):

- $a = b_{Xi}$: then $b_{Xi}\#X \in \Delta^+$ by definition, and the result follows.
- $a = a_{Xi}$: then $\pi(a_{Xi}) = \pi'(a_{Xi})$ and there is nothing to prove.
- $a = \pi(a_{Xi})$: then we distinguish two cases:
  - if $\pi(a_{Xi}) = a_{Xj}$ for some j, the result follows by the case of a_{Xi};
  - if $\pi(a_{Xi}) \neq a_{Xj}$ for all j, then $\pi(a_{Xi})\#X \in \Delta^+$ by definition.
- $a \in \pi$, but $a \neq a_{Xj}$ for all j, then $a\#X \in \Delta^+$ by definition.

The case of $X : [\mathbb{A}]\mathbb{T}$ is similar except that we additionally need to prove that

$$\Delta^+ \vdash_{\mathsf{SUB}} [c_X]\mathsf{sub}(\pi \cdot X, c_X) = \pi \cdot X.$$

This follows by axiom ($\eta{\mapsto}$), since $\Delta^+ \vdash c_X\#\pi \cdot X$. □

Remark 4.6.14. The reader might wonder why the inverse mapping of the d_X needs to rename the atoms a_{Xi} to the fresh b_{Xi}. We give an example to make this clear. Consider $(a_{T1}\ a_{T2}) \cdot T$, where we do not know $a_{T1}\#T$ or $a_{T2}\#T$. Then

$$(((a_{T1}\ a_{T2}) \cdot T)\varsigma)^{-1} \equiv \mathsf{d}_T(a_{T2}\ a_{T1})^{-1} \equiv ((b_{T2}\ a_{T2})(b_{T1}\ a_{T1}) \cdot T)[b_{T1} \mapsto a_{T2}][b_{T2} \mapsto a_{T1}].$$

By calculations we can verify Lemma 4.6.13:

$$((b_{T2}\ a_{T2})(b_{T1}\ a_{T1}) \cdot T)[b_{T1} \mapsto a_{T2}][b_{T2} \mapsto a_{T1}] = (a_{T1}\ a_{T2}) \cdot T$$

is derivable in an appropriate freshness context. Had we left out the renaming to fresh atoms the result of $(((a_{T1}\ a_{T2}) \cdot T)\varsigma)^{-1}$ would be $T[a_{T1} \mapsto a_{T2}][a_{T2} \mapsto a_{T1}]$, which is not equal to $(a_{T1}\ a_{T2}) \cdot T$, since for example

$$((a_{T1}\ a_{T2}) \cdot T)[a_{T1}/T] = a_{T2} \quad \text{but} \quad (T[a_{T1} \mapsto a_{T2}][a_{T2} \mapsto a_{T1}])[a_{T1}/T] = a_{T1}.$$

We now build up towards Theorem 4.6.19, which is our main result. Recall from Definition 4.6.12 that t' and u' are closed terms in Σ' mentioning only atoms in $\mathcal{A}^+ \setminus (\mathcal{B} \cup \mathcal{C})$.

Lemma 4.6.15. *For any $a' \in \mathcal{A}^+$, if $\vdash a' \# t'$ then $\Delta^+ \vdash a' \# t'^{\text{-}1}$.*

Proof. By induction on the structure of t'. The only non-trivial case is when $t' \equiv \mathsf{d}_X(t'_1, \ldots, t'_{k_X})$. We treat the case of $X : \mathbb{T}$, the case of $X : [\mathbb{A}]\mathbb{T}$ is similar. Suppose $\vdash a' \# \mathsf{d}_X(t'_1, \ldots, t'_{k_x})$. Then $\vdash a' \# t'_i$ for $1 \leq i \leq k_X$ by Lemma 2.4.2. By the inductive hypothesis, $\Delta^+ \vdash a' \# t'^{\text{-}1}_i$. We must show

$$\Delta^+ \vdash a' \# (\pi \cdot X)[b_{X1} \mapsto t'^{\text{-}1}_1] \cdots [b_{Xk_x} \mapsto t'^{\text{-}1}_{k_x}],$$

where $\pi = (b_{Xk_x}\ a_{Xk_x}) \cdots (b_{X1}\ a_{X1})$. We distinguish two cases:

- $a' = b_{Xj}$ for some j: then $b_{Xj} \# [b_{Xj}]((\pi \cdot X)[b_{X1} \mapsto t'^{\text{-}1}_1] \cdots [b_{Xj-1} \mapsto t'^{\text{-}1}_{j-1}])$ by $(\#[]\mathbf{a})$, and the result follows by the rules of freshness using the inductive hypothesis.
- $a' \neq b_{Xj}$ for all j: then $\pi^{\text{-}1}(a') \neq a_{Xj}$ for all j, so $\pi^{\text{-}1}(a') \# X \in \Delta^+$ by definition. The result follows using the rules for freshness and the inductive hypothesis.

□

Lemma 4.6.16. $\Delta^+ \vdash_{\mathsf{CORE}} (\pi \cdot t')^{\text{-}1} = \pi \cdot t'^{\text{-}1}$ *when π mentions only atoms from $\mathcal{A}^+ \setminus (\mathcal{B} \cup \mathcal{C})$.*

Proof. By induction on the structure of t'. In the case of $t' \equiv \mathsf{d}_X(a'_{X1}, \ldots, a'_{Xk_X})$ with $X \in \mathcal{X}$, we use the fact that π does not mention atoms from $\mathcal{B}$ or $\mathcal{C}$. □

Lemma 4.6.17. *If $\vdash_{\mathsf{CORE}} t' = u'$ then $\Delta^+ \vdash_{\mathsf{CORE}} t'^{\text{-}1} = u'^{\text{-}1}$.*

Proof. By induction on the structure of t', using the syntactic criteria of Corollary 2.5.4. The only non-trivial case is when $t' \equiv [a']v'$, $u' \equiv [b']w'$, $\vdash b' \# v'$ and $\vdash_{\mathsf{CORE}} (b'\ a') \cdot v' = w'$. By Lemma 4.6.15 we obtain $\Delta^+ \vdash b' \# v'^{\text{-}1}$, and by the inductive hypothesis $\Delta^+ \vdash_{\mathsf{CORE}} ((b'\ a') \cdot v')^{\text{-}1} = w'^{\text{-}1}$. By Lemma 4.6.16 we also know $\Delta^+ \vdash_{\mathsf{CORE}} ((b'\ a') \cdot v')^{\text{-}1} = (b'\ a') \cdot v'^{\text{-}1}$, so by standard reasoning using the rules for freshness and equality we conclude $\Delta^+ \vdash_{\mathsf{CORE}} [a']v'^{\text{-}1} = [b']w'^{\text{-}1}$. □

Lemma 4.6.18. *If $\vdash_{\mathsf{SIMPr}} t' \to u'$ then $\Delta^+ \vdash_{\mathsf{SUB}} t'^{\text{-}1} = u'^{\text{-}1}$.*

Proof. We work by induction on the derivation of $\vdash_{\mathsf{SIMPr}} t' \to u'$ to show that $\Delta^+ \vdash_{\mathsf{SUB}} t'^{\text{-}1} = u'^{\text{-}1}$ is derivable.

If the derivation concludes in $(\to[])$ or $(\to\mathsf{f})$ we may use the inductive hypothesis and extend the derivation with (**cong**[]) or (**cong**f) respectively.

Suppose the derivation concludes in ($\to$**rew**). Then there are various cases depending on which rewrite rule is used:

- (**Rvar**). $\Delta^+ \vdash_{\mathsf{SUB}} a'[a' \mapsto v'^{\text{-}1}] = v'^{\text{-}1}$ is an instance of axiom (**var**$\mapsto$).
- (**Rb**). $\Delta^+ \vdash_{\mathsf{SUB}} b'[a' \mapsto v'^{\text{-}1}] = b'$ is an instance of $(\#\mapsto)$, since $\Delta^+ \vdash a' \# b'$.

- (**Rf**), $\mathsf{f} \notin \mathcal{D}$.

  $$\Delta^+ \vdash_{\mathsf{SUB}} \mathsf{f}(v_1'^{\text{-}1}, \ldots, v_n'^{\text{-}1})[a' \mapsto v'^{\text{-}1}] = \mathsf{f}(v_1'^{\text{-}1}[a' \mapsto v'^{\text{-}1}], \ldots, v_n'^{\text{-}1}[a' \mapsto v'^{\text{-}1}])$$

  is an instance of ($\mathsf{f}\mapsto$).

- (**Rf**), $\mathsf{f} \in \mathcal{D}$. In case $X : \mathbb{T}$, we must show

  $$\begin{aligned}&\Delta^+ \vdash_{\mathsf{SUB}} (\pi \cdot X)[b_{X1} \mapsto v_1'^{\text{-}1}] \cdots [b_{Xk_X} \mapsto v_{k_X}'^{\text{-}1}][a' \mapsto v'^{\text{-}1}] = \\ &(\pi \cdot X)[b_{X1} \mapsto v_1'^{\text{-}1}[a' \mapsto v'^{\text{-}1}]] \cdots [b_{Xk_X} \mapsto v_{k_X}'^{\text{-}1}[a' \mapsto v'^{\text{-}1}]],\end{aligned}$$

  where $\pi = (b_{Xk_X}\ a_{Xk_X}) \cdots (b_{X1}\ a_{X1})$.
  Since $b_{Xi} \notin v'$ for all i, we know $\vdash b_{Xi}\#v'$. Then also $\Delta^+ \vdash b_{Xi}\#v'^{\text{-}1}$ by Lemma 4.6.15. Now we apply part 4 of Lemma 4.6.1, such that the left-hand-side of the proof obligation is SUB-equal to

  $$(\pi \cdot X)[a' \mapsto v'^{\text{-}1}][b_{X1} \mapsto v_1'^{\text{-}1}[a' \mapsto v'^{\text{-}1}]] \cdots [b_{Xk_X} \mapsto v_{k_X}'^{\text{-}1}[a' \mapsto v'^{\text{-}1}]].$$

  By the rules for equality, this is equal to the right-hand-side of the proof obligation when $\Delta^+ \vdash_{\mathsf{SUB}} (\pi \cdot X)[a' \mapsto v'^{\text{-}1}] = \pi \cdot X$. By axiom ($\#\mapsto$) this is when $\Delta^+ \vdash a'\#\pi \cdot X$, i.e. when $\pi^{\text{-}1}(a')\#X \in \Delta^+$. There are two possibilities:

  – $a' = a_{Xi}$ for some i: then $\pi^{\text{-}1}(a') = b_{Xi}$, and $b_{Xi}\#X \in \Delta^+$ by definition.

  – $a' \neq a_{Xi}$ for all i: then $\pi^{\text{-}1}(a') = a'$, and $a'\#X \in \Delta^+$ by definition.

  The result follows.
  The case of $X : [\mathbb{A}]\mathbb{T}$ is similar.

- (**Rabs**). Suppose $\vdash c\#v'$. Then $\Delta^+ \vdash c\#v'^{\text{-}1}$ by Lemma 4.6.15. Then

  $$\Delta^+ \vdash_{\mathsf{SUB}} ([c]w'^{\text{-}1})[a' \mapsto v'^{\text{-}1}] = [c](w'^{\text{-}1}[a' \mapsto v'^{\text{-}1}])$$

  is an instance (**abs**$\mapsto$), and we are done.

The result follows. □

Theorem 4.6.19 (SUB on open terms)**.**

$$\Delta \vdash_{\mathsf{SUB}} t = u \quad \textit{if and only if} \quad \vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma.$$

Proof. The left-to-right part is Lemma 4.6.10.

For the right-to-left part, suppose that $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. We have observed that there are SIMPr rewrites

$$t\varsigma \equiv t_1' \to t_2' \to \ldots \to t_m' \equiv t\varsigma^{\downarrow}$$

and

$$u\varsigma \equiv u_1' \to u_2' \to \ldots \to u_n' \equiv u\varsigma^{\downarrow}.$$

Then by Lemma 4.6.18, we know

$$\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{\text{-}1} \equiv t_1'^{\text{-}1} = t_2'^{\text{-}1} = \ldots = t_m'^{\text{-}1} \equiv (t\varsigma^{\mathbb{I}})^{\text{-}1}$$

and

$$\Delta^+ \vdash_{\mathsf{SUB}} (u\varsigma)^{\text{-}1} \equiv u_1'^{\text{-}1} = u_2'^{\text{-}1} = \ldots = u_n'^{\text{-}1} \equiv (u\varsigma^{\mathbb{I}})^{\text{-}1},$$

so $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{\text{-}1} = (t\varsigma^{\mathbb{I}})^{\text{-}1}$ and $\Delta^+ \vdash_{\mathsf{SUB}} (u\varsigma)^{\text{-}1} = (u\varsigma^{\mathbb{I}})^{\text{-}1}$ by transitivity.

We supposed that $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ so by Corollary 4.5.21 it must be the case that $\Delta^+ \vdash_{\mathsf{CORE}} t\varsigma^{\mathbb{I}} = u\varsigma^{\mathbb{I}}$. By Lemma 4.6.17 also $\Delta^+ \vdash_{\mathsf{CORE}} (t\varsigma^{\mathbb{I}})^{\text{-}1} = (u\varsigma^{\mathbb{I}})^{\text{-}1}$.

Then $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{\text{-}1} = (u\varsigma)^{\text{-}1}$ using symmetry and transitivity. By Lemma 4.6.13 then also $\Delta^+ \vdash_{\mathsf{SUB}} t = u$. Since Δ^+ freshly extends Δ with atoms not in t and u, we conclude $\Delta \vdash_{\mathsf{SUB}} t = u$ by strengthening (Theorem 2.4.14). □

As a simple corollary of Theorem 4.6.19, we obtain decidability of SUB.

Corollary 4.6.20 (Decidability of SUB)**.**

$$\Delta \vdash_{\mathsf{SUB}} t = u \quad \textit{if and only if} \quad \vdash_{\mathsf{CORE}} t\varsigma^{\mathbb{I}} = u\varsigma^{\mathbb{I}}.$$

As a corollary, it is decidable whether $\Delta \vdash_{\mathsf{SUB}} t = u$*.*

Proof. By Theorem 4.6.19, $\Delta \vdash_{\mathsf{SUB}} t = u$ is equivalent to $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. By Corollary 4.5.21, this is equivalent to $t\varsigma^{\mathbb{I}} = u\varsigma^{\mathbb{I}}$.

The corollary then follows by the syntactic criteria of equality in CORE (Corollary 2.5.4). □

By a similar method to the one we used to prove Theorem 4.6.19 we can prove conservativity of SUB over CORE. We use the notation and machinery of this subsection in the following proof:

Theorem 4.6.21 (Conservativity of SUB over CORE)**.** *Suppose that t and u do* not *mention term-former* sub*. Then*

$$\Delta \vdash_{\mathsf{SUB}} t = u \qquad \textit{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

Proof. A derivation in CORE is also a derivation in SUB so the right-to-left implication is immediate.

Now suppose that $\Delta \vdash_{\mathsf{SUB}} t = u$. We take a suitably chosen enriched signature and suitably chosen ς, as in the proofs above. By Lemma 4.6.10, $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. By construction $t\varsigma$ and $u\varsigma$ do not mention sub, therefore by Corollary 4.5.20 also $\vdash_{\mathsf{CORE}} t\varsigma = u\varsigma$.

Given $\vdash_{\mathsf{CORE}} t\varsigma = u\varsigma$ we can show $\Delta \vdash_{\mathsf{CORE}} t = u$ by exploiting the syntactic criteria of Corollary 2.5.4. The proof is by induction on t using detailed but entirely routine calculations. We consider just one case, the hardest one:

Suppose $t \equiv \pi \cdot X$ and $X : [\mathbb{A}]\mathbb{T}$. Then

$$t\varsigma \equiv [c_X]\mathsf{d}_X(\pi(a_{X1}), \ldots, \pi(a_{Xk_X})).$$

By the syntactic criteria of Corollary 2.5.4 if $\vdash_{\mathsf{CORE}} t\varsigma = u\varsigma$ it *must* be that

$$u\varsigma \equiv [b]\mathsf{d}_X(\pi(a_{X1}), \ldots, \pi(a_{Xk_X})).$$

Here b is not equal to $\pi(a_{Xi})$ for $1 \leq i \leq k_X$. By the construction of $u\varsigma$ and the way we chose $a_{X1}, \ldots, a_{Xk_X}$ to be the atoms mentioned in Δ, t, or u which are *not* provably fresh for X in Δ, it follows that u *must* have been equal to $\pi' \cdot X$ for some π' such that $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$. It follows that $\Delta \vdash_{\mathsf{CORE}} t = u$ as required. $\square$

4.6.3 ω-Completeness

We consider completeness with respect to the ground term model (see Definition 4.6.22 below for a formal definition). This is also called *ω-completeness*. Before we go into the proof of this notion of completeness, it is useful to mention why this is an appropriate notion to consider.

In Chapter 3 we have shown that nominal algebra enjoys a general completeness result with respect to a standard class of semantics in *nominal sets* [GP02]. SUB is a nominal algebra theory, so it is automatically sound and complete with respect to the standard class of nominal sets semantics.

A completeness result is *weaker*, the *larger* the class of semantics that it uses. Can we strengthen this general result to some more specific class than the nominal sets models?

Theorem 4.6.19 can be read as a completeness result with respect to a class of models built out of syntax enriched with finitely but unboundedly many extra term-formers d_X. We could stop there, *however*, we would have an even more powerful completeness result if we could strengthen this to completeness with respect to terms of signature Σ itself.

In fact, so long as Σ contains a term-former which takes more than one argument (like $\mathsf{pair} : (\mathbb{T}, \mathbb{T})\mathbb{T})$), then we can nail SUB down to *the* theory of capture-avoiding substitution on ground terms in Σ. We now have all the machinery to do this quite easily.

Definition 4.6.22. Call σ a **ground substitution** for Δ, t and u when for every unknown X in Δ, t and u, $\sigma(X)$ is a ground term.

Call SUB **ω-complete** when for all Δ, t and u, if $t\sigma^{\downarrow} =_\alpha u\sigma^{\downarrow}$ for all Δ-consistent ground substitutions σ (for Δ, t and u), then $\Delta \vdash_{\mathsf{SUB}} t = u$.

We shall prove the contrapositive. Supposing $\Delta \nvdash_{\mathsf{SUB}} t = u$, we will exhibit some Δ-consistent ground substitution σ such that $t\sigma^{\downarrow} \neq_\alpha u\sigma^{\downarrow}$.

We cannot use ς from Subsection 4.6.2 because ς maps to an extended signature Σ' with extra term-formers d_X, but we can use ς to construct another substitution with the right properties, as we now see.

Recall from the previous subsection the chain of rewrites

$$t\varsigma \equiv t'_1 \to t'_2 \to \ldots \to t'_m \equiv t\varsigma^{\downarrow} \quad \text{and} \quad u\varsigma \equiv u'_1 \to u'_2 \to \ldots \to u'_n \equiv u\varsigma^{\downarrow}.$$

Note that $t\varsigma^{\downarrow}$ and $u\varsigma^{\downarrow}$ are ground.

Definition 4.6.23. Let $\mathcal{A}'$ be the set of all atoms mentioned in the above chains, let t' and u' range over closed terms in Σ' mentioning only atoms from $\mathcal{A}'$, and choose atoms $\{a_X \mid X \in \mathcal{X}\}$ completely fresh from $\mathcal{A}'$.

Assuming that Σ contains a binary term-former, say $\mathsf{pair} : (\mathbb{T}, \mathbb{T})\mathbb{T})$, define a translation -* from closed terms in Σ' to closed terms in Σ by:

$$a'^* \equiv a' \qquad ([a']t')^* \equiv [a']t'^* \qquad \mathsf{f}(t'_1, \ldots, t'_n)^* \equiv \mathsf{f}(t'^*_1, \ldots, t'^*_n) \quad (\mathsf{f} \notin \mathcal{D})$$
$$\mathsf{d}_X()^* \equiv \mathsf{pair}(a_X, a_X) \qquad \mathsf{d}_X(t'_1)^* \equiv \mathsf{pair}(a_X, t'^*_1)$$
$$\mathsf{d}_X(t'_1, \ldots, t'_{k_x})^* \equiv \mathsf{pair}(a_X, \mathsf{pair}(t'^*_1, \mathsf{pair}(t'^*_2, \ldots, \mathsf{pair}(t'^*_{k_x-1}, t'^*_{k_x})))) \quad (k_X > 1)$$

It is not hard to verify the following properties:

Lemma 4.6.24.

1. $\vdash (t'^*)^{\downarrow} \equiv (t'^{\downarrow})^*$.
2. $\vdash_{\mathsf{CORE}} t' = u'$ *if and only if* $\vdash_{\mathsf{CORE}} t'^* = u'^*$.
3. $\vdash_{\mathsf{SIMP}} t' = u'$ *if and only if* $\vdash_{\mathsf{SIMP}} t'^* = u'^*$.

Proof. Part 1 is by induction on the syntax of t'. For the case of $t' \equiv \mathsf{sub}(u', v')$, we use the property that $g'[h'/a']^* \equiv g'^*[h'^*/a']$ (where g', h', a' and $g'[h'/a']$ mention only atoms from $\mathcal{A}'$), which is easy to verify.

Part 2 is by induction on the syntax of t', using the syntactic criteria from Corollary 2.5.4.

For part 3, by Corollary 4.5.21 it suffices to prove the equivalent

$$\vdash_{\mathsf{CORE}} t'^{\downarrow} = u'^{\downarrow} \quad \text{if and only if} \quad \vdash_{\mathsf{CORE}} (t'^*)^{\downarrow} = (u'^*)^{\downarrow},$$

which follows by parts 1 and 2. □

Definition 4.6.25. Define the substitution ς^* by:

$$\begin{array}{rcll} \varsigma^*(X) & = & \varsigma(X)^* & (X \in \mathcal{X}) \\ \varsigma^*(X) & = & X & (X \notin \mathcal{X}) \end{array}$$

It is a fact of the construction that ς^* maps every X appearing in Δ, t, or u, to a *ground term in* Σ. This is morally 'the same' as $\varsigma(X)$, but we map each extra term-former d_X to a collection of instances of pair.

Lemma 4.6.26. *For any subterm v of t or u,* $v(\varsigma^*) \equiv (v\varsigma)^*$.

Proof. By straightforward induction on the structure of v. □

Corollary 4.6.27. $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ *if and only if* $\vdash_{\mathsf{SIMP}} t(\varsigma^*) = u(\varsigma^*)$.

Proof. By Lemma 4.6.26 and part 3 of Lemma 4.6.24. □

Theorem 4.6.28 (ω-completeness)**.** SUB *is ω-complete.*

Proof. We show that if $\Delta \nvdash_{\mathsf{SUB}} t = u$ then there exists a Δ-consistent ground substitution σ (for Δ, t and u) such that $t\sigma^{\mathtt{I}} \neq_\alpha u\sigma^{\mathtt{I}}$.

Suppose $\Delta \nvdash_{\mathsf{SUB}} t = u$. Then also $\nvdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ by Theorem 4.6.19. Now by Corollary 4.6.27 we obtain $\nvdash_{\mathsf{SIMP}} t(\varsigma^*) = u(\varsigma^*)$. Then $(t\varsigma^*)^{\mathtt{I}} \neq_\alpha (u\varsigma^*)^{\mathtt{I}}$ by Theorem 4.3.13 and Corollary 4.5.21.

Now take $\sigma = \varsigma^*$. Since ς^* maps to ground terms in Σ, the result follows. □

4.6.4 Restricting the Sort System

We now turn to our discussion in Section 4.2 on unknowns of abstraction sort $[\mathbb{A}]\mathbb{T}$ and term-formers $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$ (Remark 4.2.5). We mentioned that they are convenient, but *not* strictly necessary. One way of seeing this is by observing that we could redo all the mathematics in this chapter in the restricted setting with minor modifications. Another way is to make the argument formal by leveraging the proof of Theorem 4.6.19. We will pursue this more formal approach here.

Since the restriction of unknowns implies that we have to instantiate ($\mathsf{f}\!\mapsto$) for each term-former f, we will consider a concrete signature. We consider a typical signature for languages with binding, namely that of theory LAM from Example 4.2.4.

Definition 4.6.29. Consider the following signature:

$$\mathsf{lam} : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \mathsf{app} : (\mathbb{T}, \mathbb{T})\mathbb{T} \qquad \mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}.$$

As usual, we write $\mathsf{lam}([a]t)$ as $\lambda[a]t$, $\mathsf{app}(t, u)$ as tu and $\mathsf{sub}([a]t, u)$ as $t[a \mapsto u]$. Now let theory SUB′ over this signature have the following axioms:

$$\begin{array}{rrrrcl}
(\mathbf{var}{\mapsto}') & & \vdash & a[a \mapsto T] & = & T \\
(\#{\mapsto}') & a\#U & \vdash & U[a \mapsto T] & = & U \\
(\mathbf{lam}{\mapsto}') & b\#T & \vdash & (\lambda[b]U)[a \mapsto T] & = & \lambda[b](U[a \mapsto T]) \\
(\mathbf{app}{\mapsto}') & & \vdash & (UV)[a \mapsto T] & = & (U[a \mapsto T])(V[a \mapsto T]) \\
(\mathbf{sub}{\mapsto}') & b\#T & \vdash & V[b \mapsto U][a \mapsto T] & = & V[a \mapsto T][b \mapsto U[a \mapsto T]] \\
(\mathbf{ren}{\mapsto}') & b\#T & \vdash & T[a \mapsto b] & = & (b\ a) \cdot T
\end{array}$$

Theorem 4.6.30. *Suppose* SUB *is a theory of substitution over the signature of* LAM. *Then for any Δ, t and u (in this signature) not mentioning unknowns of sort $[\mathbb{A}]\mathbb{T}$ or terms of sort $[\mathbb{A}][\mathbb{A}]\mathbb{T}$:*

$$\Delta \vdash_{\mathsf{SUB}} t = u \quad \textit{if and only if} \quad \Delta \vdash_{\mathsf{SUB}'} t = u$$

Proof. For the right-to-left part it suffices to show that each axiom of SUB′ can be derived in SUB. Both axiom ($\mathbf{lam}{\mapsto}'$) and ($\mathbf{sub}{\mapsto}'$) follow by an instance of ($\mathsf{f}\!\mapsto$) and ($\#\!\mapsto$), the other axioms of SUB′ follow directly from their corresponding axioms in SUB.

For the left-to-right part, suppose $\Delta \vdash_{\mathsf{SUB}} t = u$. Then also $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$ by Lemma 4.6.10. We observe that there are SIMPr rewrites

$$t\varsigma \equiv t_1 \to t_2 \to \ldots \to t_m \equiv t\varsigma^{\mathtt{I}} \quad \text{and} \quad u\varsigma \equiv u_1 \to u_2 \to \ldots \to u_n \equiv u\varsigma^{\mathtt{I}},$$

such that whenever a term of sort $[\mathbb{A}][\mathbb{A}]\mathbb{T}$ is introduced by a rewrite, it is removed in the next step. More precisely, only an instance of rewrite rule (**Rf**), where $\mathsf{f} = \mathsf{lam}$, can introduce such a term, but we can always apply the (**Rabs**) rule to get rid of it.

We can use properties similar to Lemma 4.6.18 on these chains of rewrites to obtain derivations of

$$\Delta^+ \vdash_{\mathsf{SUB}'} (t\varsigma)^{-1} = (t\varsigma^{\mathbb{I}})^{-1} \quad \text{and} \quad \Delta^+ \vdash_{\mathsf{SUB}'} (u\varsigma)^{-1} = (u\varsigma^{\mathbb{I}})^{-1}.$$

That is, we need one such property for direct rewrites and another one for two-step rewrites. Note that in the current setting the inverse translation never introduces unknowns of sort $[\mathbb{A}]\mathbb{T}$ or terms of sort $[\mathbb{A}][\mathbb{A}]\mathbb{T}$.

We also have $\Delta^+ \vdash_{\mathsf{CORE}} (t\varsigma^{\mathbb{I}})^{-1} = (u\varsigma^{\mathbb{I}})^{-1}$ by Corollary 4.5.21, Lemma 4.6.17 and our assumption that $\vdash_{\mathsf{SIMP}} t\varsigma = u\varsigma$. Then $\Delta^+ \vdash_{\mathsf{SUB}} (t\varsigma)^{-1} = (u\varsigma)^{-1}$ using (**symm**) and (**tran**). Using a property similar to Lemma 4.6.13 we obtain $\Delta^+ \vdash_{\mathsf{SUB}'} t = u$. We conclude $\Delta \vdash_{\mathsf{SUB}'} t = u$ by strengthening (Theorem 2.4.14). □

4.7 Conclusions

We have provided a general axiomatisation of capture-avoiding substitution in nominal algebra. This axiomatisation can be considered *the right* one, because it is sound and complete with respect to a canonical term model (Theorem 4.6.28). It also provides a precise sense in which that axiomatisation be considered *tractable*; decidability of equality up to the axioms for substitution (Corollary 4.6.20).

4.7.1 Related Work

At first sight, Crabbé's axiomatisation of substitution [Cra04] looks much like our axiomatisation and shares (in our terminology) atoms and freshness conditions. However, his axiomatisation is not capture-avoiding from the simple fact that he does not treat binding: '...we are not concerned with the notion of bound variable' [Cra04, page 2].

We now discuss related works that do treat binding.

Nominal approaches

Most existing publications on nominal techniques have treated axiomatisations of capture-avoiding substitution. In [GP02, Pit03, CP07, FG07], such an axiomatisation serves as a motivating example of the theory presented in the paper at hand. Ohter publication put the axiomatisation to use. In [Pit06], Pitts uses simultaneous substitutions in the context of α-structural recursion and induction to give a clean but mathematically rigorous treatment of normalisation by evualation in the simply typed λ-calculus. In [FS06], Fiore and Staton use atoms-for-atoms substitution to define a congruence rule format for name-passing process calculi.

Finally, in Chapter 5 capture-avoiding substitution is used in the definition of one-and-a-halfth-order logic.

These papers show that capture-avoiding substitution can be axiomatised in a way that is close to informal practice. However, none of the papers show that the axiomatisations they provide are *complete* with respect to the ground term model. This is the major contribution of this chapter. This was far from trivial, since we needed to augment the standard nominal axiomatisation with the axioms ($\mathbf{ren}{\mapsto}$) and ($\eta{\mapsto}$).

Higher-order approaches

Higher-order approaches map substitution to the meta-level, which is capture-avoiding by default. For example, when writing down axioms for the λ-calculus, we make use of meta-level abstraction and application (example taken from [MN98]):

$$\begin{array}{lrcl} (\beta) & \mathsf{app}(\mathsf{lam}(\lambda x.F(x)), S) & = & F(S) \\ (\eta) & \mathsf{lam}(\lambda x.\mathsf{app}(S, x)) & = & S \end{array}$$

Here lam and app are constants representing object-level abstraction and application, and $\lambda x.t$ and $t(u)$ represent meta-level abstraction and application. In the (β) axiom the function variable F is parameterised by the bound variable x to make substitution capturing for x. In the (η) axiom x can never become bound in any term we substitute for S, since the meta-level substitution mechanism takes care of this.

Binding algebras

Fiore, Plotkin and Turi define substitution in their binding framework [FPT99]. It is not clear how to relate their definition of substitution to our axiomatisation, because it is not clear how to relate the frameworks altogether (see the Conclusions of Chapter 3 for details).

Cylindric algebras

Feldman [Fel82] gives an algebraic axiomatisation inspired by a concrete model of functions/evaluations. His axioms are closer in spirit to cylindric algebras [BS81] and lambda-abstraction algebras [LS04, Sal00]. The three approaches share an infinity of term-formers representing $\lambda[a]$, -$[a \mapsto$ -$]$, and $\exists[a]$.

We see the advantage of our treatment as systematising and formalising precisely what rôle the atoms really have. In any case the approaches above *cannot directly express* ($\mathbf{ren}{\mapsto}$), ($\#{\mapsto}$), and ($\mathbf{abs}{\mapsto}$), even though instantiations are derivable for closed terms by calculations parametric over their specific structure.

Combinators

Combinatory logic [CF58, Bar84] and related systems implement substitution by 'pipes' (e.g. the translation of λ-terms into combinatory logic). There is no native notion of binder, nor of capture-avoidance. General truths such as ($\#{\mapsto}$) are not provable as equalities between combinators, though they remain true and can be proved informally by calculations parametric over specific structure.

Calculi of explicit substitutions

Lescanne's classic survey [Les94] and the thesis of Bloo [Blo97] chart a vast literature on λ-calculi with explicit substitutions. These decompose β-reduction as a rule to introduce explicit substitution ($(\lambda a.u)t \to u[a \mapsto t]$), and explicit rules for that substitution's subsequent behaviour (which is to substitute, of course).

These calculi are designed to measure the cost of a β-reduction (in an implementation, which may be based on de Bruijn indexes [dB72] or on named variable symbols). They do not *axiomatise* substitution, they *implement* it. For that reason, they are more close in spirit to rewrite system SIMPr than to the equational theories SIMP and SUB. For example, 'confluence' is a typical correctness criterion for a calculus, and 'ω-completeness' often is not.

4.7.2 Future Work

Equality and *unification* up to CORE is decidable [UPG04]. Equality in SUB is decidable (Corollary 4.6.20), but *unification* up to SUB remains an open problem.

In the Conclusions of Chapter 2 we mentioned that we are interested in developing logics with hierarchies of 'increasingly meta'-variables. We could use these techniques [Gab05, Gab07b, GL07] to try to take SUB *over itself* — that is, to taking what we write in this chapter as, say, $(X[a \mapsto Y])[t/X]$ and expressing it in a stronger axiom system as $(X[a \mapsto Y])[X \mapsto \mathcal{T}]$ where $\mathcal{T}$ is a 'stronger' meta-variable.

Chapter 5

One-and-a-halfth-Order Logic

5.1 Introduction

Informal use of first-order predicate logic [Gen35, Pra65] has all the features which the tools we have developed so far are designed to handle; logical and computational content, binding — and a habit of making heavy use of meta-variables. Consider for example the following 'valid sequents':

$$\begin{array}{rl}
\vdash \phi \supset (\psi \supset \phi) & \\
\phi \vdash \phi[x \mapsto t] & \text{if } x \notin \mathit{fv}(\phi) \\
\forall x.\phi \vdash \forall y.(\phi[x \mapsto y]) & \text{if } y \notin \mathit{fv}(\phi) \\
\phi \vdash \forall x.\phi & \text{if } x \notin \mathit{fv}(\phi) \\
\phi, \psi \vdash \forall x.\phi & \text{if } x \notin \mathit{fv}(\phi) \\
\phi, \forall x.(\phi \supset \psi) \vdash \forall x.\psi & \text{if } x \notin \mathit{fv}(\phi) \\
\forall y.\forall x.\phi \vdash \forall x.(\phi[y \mapsto x]) &
\end{array}$$

These sequents cannot be derived in the syntax of first-order logic itself, since they contain meta-variables ϕ, ψ and t. These meta-variables are *not* part of the syntax, they *range over* it. Furthermore we refer to properties of syntax when we write '$x \notin \mathit{fv}(\phi)$' and '$\phi[x \mapsto t]$', but the syntax of first-order logic cannot represent these explicitly.

Of course to us humans this is all obvious. One reason is that the derivations fall into a limited number of schema. Consider for example the following 'derivations':

$$\dfrac{\dfrac{\dfrac{}{\psi, \phi \vdash \phi}\,(\mathbf{Ax})}{\phi \vdash \psi \supset \phi}\,(\supset\!\mathbf{R})}{\vdash \phi \supset (\psi \supset \phi)}\,(\supset\!\mathbf{R}) \qquad \dfrac{\dfrac{\dfrac{}{\bot, \bot \vdash \bot}\,(\mathbf{Ax})}{\bot \vdash \bot \supset \bot}\,(\supset\!\mathbf{R})}{\vdash \bot \supset (\bot \supset \bot)}\,(\supset\!\mathbf{R})$$

The ‘derivation’ on the left is not a derivation, but it obviously represents a schema of derivations of which the (real) derivation on the right is an instance setting ϕ and ψ to $\bot$. But is there a logic in which the beast on the left is a derivation too?

This chapter presents **one-and-a-halfth-order logic**. This logic generalises first-order logic by adding explicit meta-variables P, Q and T which in the logic represent the ϕ, ψ and t of the derivations above. The classification *one-and-a-halfth* indicates that the logic is somewhere between first- and second-order logic: it is not just first-order because of the meta-variables, yet not second-order because there is no quantification over meta-variables.

One-and-a-halfth-order logic permits generalised forms of α-equivalence, substitution, capture-avoidance, and quantifier introduction rules. Even though it is a completely formal logic with clearly-specified derivation rules, we shall see that the derivations it admits are very close to the schemas of derivations which we see all the time in informal practice. For example, one-and-a-halfth-order logic allows us to write a formal counterpart of the following schema of derivations of $\phi, \forall x.(\phi \supset \psi) \vdash \forall x.\psi$ where $x \notin fv(\phi)$ (the 6th derivation of Figure 5.3):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\phi, \vdash \psi, \phi}\,(\mathbf{Ax}) \qquad \dfrac{}{\psi, \phi \vdash \psi}\,(\mathbf{Ax})}{\phi, \phi \supset \psi \vdash \psi}\,(\supset\mathbf{L})}{\phi, \forall x.(\phi \supset \psi) \vdash \psi}\,(\forall\mathbf{L}) \quad (\phi \supset \psi \; =_\alpha \; (\phi \supset \psi)[x \mapsto x])}{\phi, \forall x.(\phi \supset \psi) \vdash \forall x.\psi}\,(\forall\mathbf{R}) \quad (x \notin fv(\phi, \forall x.(\phi \supset \psi)))$$

Furthermore, we can represent the introduction of fresh atoms, as is essential in the following justification of $\phi, \psi \vdash \forall x.\phi$, where $x \notin fv(\phi)$ (the 5th derivation of Figure 5.3):

“Let y be a variable fresh for x, ϕ and ψ. Then we derive as follows:

$$\dfrac{\dfrac{}{\phi, \psi \vdash \phi}\,(\mathbf{Ax})}{\phi, \psi \vdash \forall y.\phi}\,(\forall\mathbf{R}) \quad (y \notin fv(\phi, \psi)).$$

This is $\phi, \psi \vdash \forall x.\phi$ since $\forall x.\phi =_\alpha \forall y.\phi$ when $x \notin fv(\phi)$.”

Overview Our constructions heavily rely on definitions and results from Chapters 2 and 4. In Section 5.2 we refine some of these basic definitions to better suit our needs.

We present the sequent calculus of one-and-a-halfth-order logic in Section 5.3. We also show how we can represent the informal valid sequents and derivations from this introduction in the calculus.

In Section 5.4 we establish some proof-theoretical results. We show that formal derivations correctly represent schemas of derivations in first-order logic. Furthermore, we show that the logic satisfies cut-elimination and consistency.

In Section 5.5 we show that a subset of one-and-a-halfth-order logic precisely corresponds to first-order logic. We discuss related and future work in Section 5.6.

5.2 Sorts, Terms and Substitution

We present the signature of one-and-a-halfth-order logic in the setting of nominal terms, introduced in Chapter 2. We take the signature of theory FOL from Example 2.2.17, which contains term-formers $\bot$, $\supset$, $\forall$, $\approx$ and sub, and possibly some other term-formers.

We will use a simple sort system to keep our terms well-formed. The construction is similar to that of Chapter 4, with the major differences being that we use *two* base sorts ($\mathbb{F}$ for formulae and $\mathbb{T}$ for terms) and that we allow unknowns of base sort only. We do not admit unknowns of abstraction sort like we did in Chapter 4; detailed reasons for this are given in Remark 5.3.3.

Definition 5.2.1. Fix a **sort of formulae** $\mathbb{F}$, and a **sort of terms** $\mathbb{T}$. Define **sorts** τ by the following grammar:

$$\tau \quad ::= \quad \mathbb{F} \mid \mathbb{T} \mid [\mathbb{A}]\mathbb{F} \mid [\mathbb{A}]\mathbb{T}$$

We call $[\mathbb{A}]\mathbb{F}$ and $[\mathbb{A}]\mathbb{T}$ **abstraction sorts**.

Intuitively, sorts $[\mathbb{A}]\mathbb{F}$ and $[\mathbb{A}]\mathbb{T}$ contain all formulae and terms in which an atom is abstracted.

Definition 5.2.2. We assume the infinite collection of unknowns $X, Y, Z, \ldots$ is partitioned into two infinite collections $X_\mathbb{F}, Y_\mathbb{F}, Z_\mathbb{F}, \ldots$ of **unknowns of sort** $\mathbb{F}$ and $X_\mathbb{T}, Y_\mathbb{T}, Z_\mathbb{T}, \ldots$ of **unknowns of sort** $\mathbb{T}$.

We usually leave out the subscripts of the unknowns, and we typically let P, Q, R be unknowns of sort $\mathbb{F}$, and T, U, V be unknowns of sort $\mathbb{T}$.

Definition 5.2.3. We associate a **sorting arity** $(\tau_1, \ldots, \tau_n)\tau$ with each term-former f. We write $\mathsf{f} : (\tau_1, \ldots, \tau_n)\tau$ for f **of arity** $(\tau_1, \ldots, \tau_n)\tau$. Then the sorting arities for the main term-formers are:

$$\bot : ()\mathbb{F} \qquad \supset : (\mathbb{F}, \mathbb{F})\mathbb{F} \qquad \forall : ([\mathbb{A}]\mathbb{F})\mathbb{F} \qquad \approx : (\mathbb{T}, \mathbb{T})\mathbb{F}$$
$$\mathsf{sub} : ([\mathbb{A}]\mathbb{F}, \mathbb{F})\mathbb{F} \qquad \mathsf{sub} : ([\mathbb{A}]\mathbb{T}, \mathbb{T})\mathbb{T}$$

For any other term-former f, we allow two types of sorting assertions:

- An **object-level term-former** has sorting assertion

$$(\tau_1, \ldots, \tau_n)\mathbb{T}, \quad \text{where } \tau_i \in \{\mathbb{T}, [\mathbb{A}]\mathbb{T}\} \text{ for } 1 \leq i \leq n.$$

- A **predicate term-former** has sorting assertion

$$(\tau_1, \ldots, \tau_n)\mathbb{F}, \quad \text{where } \tau_i \in \{\mathbb{T}, \mathbb{F}, [\mathbb{A}]\mathbb{T}, [\mathbb{A}]\mathbb{F}\} \text{ for } 1 \leq i \leq n.$$

So there are two term-formers sub, one for terms of sort $\mathbb{F}$ and one for terms of sort $\mathbb{T}$. This suffices for our needs; there are no sub term-formers for abstraction sorts. Note that the sorting assertions of the object-level and predicate term-formers explicitly allow the use of abstraction sorts for the arguments.

Example 5.2.4. Possible object-level term-formers include

$$0 : ()\mathbb{T} \qquad \mathsf{S} : (\mathbb{T})\mathbb{T} \qquad + : (\mathbb{T},\mathbb{T})\mathbb{T}$$

but also binders such as

$$\lambda : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \Sigma : ([\mathbb{A}]\mathbb{T})\mathbb{T} \qquad \mathsf{fix} : ([\mathbb{A}]\mathbb{T})\mathbb{T}.$$

Possible predicate term-formers include

$$\mathsf{issocrates} : (\mathbb{T})\mathbb{F} \qquad \mathsf{greek} : (\mathbb{T})\mathbb{F} \qquad \oplus : (\mathbb{F},\mathbb{F})\mathbb{F} \qquad \exists! : ([\mathbb{A}]\mathbb{F})\mathbb{F}$$

($\oplus$ is standard notation for 'exclusive or', and $\exists!$ is standard notation for the 'there exists exactly one' quantifier).

These extra term-formers would cause no difficulties for the results which follow — aside from some extra cases.

Definition 5.2.5. The **(valid) sorting assertions** $t : \tau$ are inductively defined by:

$$\frac{}{a : \mathbb{T}} \qquad \frac{}{\pi \cdot X_\tau : \tau} \qquad \frac{t : \tau}{[a]t : [\mathbb{A}]\tau} \qquad \frac{t_1 : \tau_1 \quad \cdots \quad t_n : \tau_n}{\mathsf{f}(t_1,\ldots,t_n) : \tau} \; (\mathsf{f} : (\tau_1,\ldots,\tau_n)\tau)$$

Note that in these rules, τ is restricted to $\mathbb{T}$ and $\mathbb{F}$ by definition.

From now on, we will only consider terms that adhere to the valid sorting assertions. Furthermore, we require equalities $t = u$ and meta-level substitutions σ to be *sort-respecting.*

Definition 5.2.6. We may call terms of sort $\mathbb{F}$ **formulae**; we usually let ϕ, ψ, and ρ range over them.

We write:

$$\bot() \text{ as } \bot \qquad \supset(\phi,\psi) \text{ as } \phi \supset \psi \qquad \forall([a]\phi) \text{ as } \forall[a]\phi \qquad \approx(t,u) \text{ as } t \approx u$$
$$\mathsf{sub}([a]\phi, t) \text{ as } \phi[a \mapsto t] \qquad \mathsf{sub}([a]u, t) \text{ as } u[a \mapsto t].$$

We discuss some intuitions. Let a be an atom, π be a permutation of atoms, $T : \mathbb{T}$ and $P : \mathbb{F}$ be unknowns, $t, u : \mathbb{T}$ be terms, and ϕ, ψ be formulae (terms of sort $\mathbb{F}$). Then:

- a is a term of sort $\mathbb{T}$, representing an object-level variable symbol of sort $\mathbb{T}$.
- $\pi \cdot T$ and $\pi \cdot P$ are moderated unknowns representing unknown terms (of sort $\mathbb{T}$ and $\mathbb{F}$ respectively) in which the atoms are permuted according to π when instantiated.
- $[a]t$ and $[a]\phi$ are abstractions; the sort system is such that abstractions are the only means to form well-sorted terms of abstraction sorts $[\mathbb{A}]\mathbb{T}$ and $[\mathbb{A}]\mathbb{F}$.
- $\bot$ represents false.

$$\begin{array}{rrrrcl}
(\mathbf{var}{\mapsto}) & & \vdash & a[a\mapsto T] & = & T \\
(\#{\mapsto}) & a\#X & \vdash & X[a\mapsto T] & = & X \\
(\supset{\mapsto}) & & \vdash & (P\supset Q)[a\mapsto T] & = & P[a\mapsto T]\supset Q[a\mapsto T] \\
(\approx{\mapsto}) & & \vdash & (U\approx V)[a\mapsto T] & = & U[a\mapsto T]\approx V[a\mapsto T] \\
(\forall{\mapsto}) & b\#T & \vdash & (\forall[b]P)[a\mapsto T] & = & \forall[b](P[a\mapsto T]) \\
(\mathbf{sub}{\mapsto}) & b\#T & \vdash & X[b\mapsto U][a\mapsto T] & = & X[a\mapsto T][b\mapsto U[a\mapsto T]] \\
(\mathbf{ren}{\mapsto}) & b\#X & \vdash & X[a\mapsto b] & = & (b\ a)\cdot X
\end{array}$$

Figure 5.1 Axioms of theory SUB for one-and-a-halfth-order logic

- $\phi\supset\psi$ represents logical implication.
- $\forall[a]\phi$ represents universal quantification (which takes an *abstraction* of a formula and yields a formula). A well-sorted term of the form $\forall v$ *must* be of the form $\forall[a]\phi$ (so $v\equiv[a]\phi$),
- $t\approx u$ represents equality in the object-language.
- $\phi[a\mapsto t]$ and $u[a\mapsto t]$ represent capture-avoiding substitution in the object-language. A well-sorted term of the form $\mathsf{sub}(v,t)$ *must* be of the form $\phi[a\mapsto t]$ (so $v\equiv[a]\phi$) or $u[a\mapsto t]$ (so $v\equiv[a]u$).

Definition 5.2.7. We use standard **classical logic sugar**:

$$\begin{array}{cc}
\neg\phi \text{ is } \phi\supset\bot & \top \text{ is } \neg\bot \\
\phi\land\psi \text{ is } \neg(\phi\supset\neg\psi) & \phi\lor\psi \text{ is } \neg\phi\supset\psi \\
\phi\Leftrightarrow\psi \text{ is } (\phi\supset\psi)\land(\psi\supset\phi) & \exists[a]\phi \text{ is } \neg\forall[a]\neg\phi
\end{array}$$

Note that these are *abbreviations*, not term-formers.

To save on (unnecessary) parentheses, assign precedence in descending order as follows: $[\_]\_$, $\_[\_\mapsto\_]$, $\approx$, $\{\neg,\forall,\exists\}$, $\{\land,\lor\}$, $\supset$, $\Leftrightarrow$. Also let $\land$, $\lor$, $\supset$ and $\Leftrightarrow$ associate to the right.

Because of the assigned precedence $P\land Q\supset R\lor S$ is $(P\land Q)\supset(R\lor S)$ and $\forall[a]P\land Q$ is $(\forall[a]P)\land Q$. Also $P\supset Q\supset R$ is $P\supset(Q\supset R)$.

We conclude this section with a formal definition of the theory of substitution in our sort system.

Definition 5.2.8. Write SUB for the theory with the axioms in Figure 5.1. Here a,b are distinct atoms, P,Q,R are distinct unknowns of sort $\mathbb{F}$, T,U,V are distinct unknowns of sort $\mathbb{T}$, and X is an unknown of appropriate sort.

There may also be axioms, which we shall not dwell on, to distribute substitutions through the object-level term-formers such as $+$ and λ and predicate term-formers such as greek and $\exists!$. These cause no extra issues; if the term-former takes terms of abstraction sort the equality should include a freshness side-condition in the same style as $(\forall{\mapsto})$.

Definition 5.2.8 of theory SUB looks a bit different than the definition of SUB from Definition 4.2.8 of the previous chapter (but it looks very similar to the definition in Subsection 4.6.4). Next to the axioms, the sort systems are quite different: on the one hand the sort system presented here is richer because it has two base sorts, but on the other hand it is more restricted because it lacks unknowns of abstraction sort or term-former $\mathsf{sub} : ([\mathbb{A}][\mathbb{A}]\mathbb{T}, \mathbb{T})[\mathbb{A}]\mathbb{T}$. We will show that nevertheless derivability is the same.

First of all, the notion of having two base sorts is irrelevant for the theory of substitution: we can just treat sort $\mathbb{F}$ as sort $\mathbb{T}$. Write $\overline{\mathsf{SUB}}$ for the theory of substitution from Definition 5.2.8, where sort $\mathbb{F}$ is replaced by $\mathbb{T}$. Also write $\overline{\Delta}$, $\overline{t}$ and $\overline{u}$ for the replacement of $\mathbb{F}$ by $\mathbb{T}$ in Δ, t and u.

Lemma 5.2.9. *Let* SUB *be the theory of substitution from Definition 5.2.8. Then:*

$$\Delta \vdash_{\mathsf{SUB}} t = u \quad \textit{if and only if} \quad \overline{\Delta} \vdash_{\overline{\mathsf{SUB}}} \overline{t} = \overline{u}.$$

Proof. Each derivation of $\Delta \vdash_{\mathsf{SUB}} t = u$ is also a derivation of $\overline{\Delta} \vdash_{\overline{\mathsf{SUB}}} \overline{t} = \overline{u}$, and vice versa. □

Note that theory $\overline{\mathsf{SUB}}$ equates more terms than SUB, but these terms are not part of the $\mathbb{F}$ to $\mathbb{T}$ translation. For example $(\bot \approx \bot)[a \mapsto T] = \bot \approx \bot$ is not derivable in SUB (because the terms do not satisfy the sorting assertions), but it is derivable in $\overline{\mathsf{SUB}}$. However, this equation cannot be the result of a translation from the signature of SUB.

We use theory $\overline{\mathsf{SUB}}$ to make a formal connection with theory SUB from Definition 4.2.8 of the previous chapter.

Lemma 5.2.10. *Let* SUB *be the theory of substitution from Definition 4.2.8 over the signature of* $\overline{\mathsf{SUB}}$*. Then:*

$$\Delta \vdash_{\overline{\mathsf{SUB}}} t = u \quad \textit{if and only if} \quad \Delta \vdash_{\mathsf{SUB}} t = u.$$

Proof. Completely analogous to the proof of Theorem 4.6.30. □

So Lemmas 5.2.9 and 5.2.10 tell us that we are allowed to use the results from Chapter 4, which we will do.

5.3 A Sequent Calculus

We present the sequent calculus of one-and-a-halfth-order logic. Recall that by our terminology a formula ϕ is a term of sort $\mathbb{F}$.

Definition 5.3.1. Let **(formula) contexts** Φ, Ψ be finite (possibly empty) sets of formulae. A **(one-and-a-halfth-order) sequent** is a triple $\Phi \vdash_{\Delta} \Psi$ where Δ is a freshness context and Φ and Ψ are formula contexts; when a context appears to the right of $\vdash$ we may call it a **co-context**.

We may write ϕ for $\{\phi\}$, ϕ, Φ for $\{\phi\} \cup \Phi$, and Φ, Φ' for $\Phi \cup \Phi'$, and we may omit empty formula contexts, e.g. writing $\vdash_\vartriangle$ for $\emptyset \vdash_\vartriangle \emptyset$.

Extend the notions of occurrence and closedness to formula contexts elementwise; for example $a \in \Phi$ if $a \in \phi$ for some $\phi \in \Phi$.

Definition 5.3.2. Let the **derivable sequents** of one-and-a-halfth-order logic be inductively specified by the rules in Figure 5.2.

Our rules resemble those of Gentzen's sequent calculus for classical first-order logic with equality [DV01, Gen35, Pra65], but with the following distinctive features:

- *Unknown formulae* and *unknown terms* are represented explicitly by unknowns of sort $\mathbb{F}$ and $\mathbb{T}$.
- We can make *freshness* assumptions about unknowns (using (**Fr**)), and these affect derivability, for example in ($\forall$**R**).
- A theory of equality of terms up to α-equivalence and capture-avoiding substitution is represented by derivability in theory SUB.
- Side-conditions on substitution, freshness and atoms not occurring in terms, are all *decidable.*

(**StructL**) and (**StructR**) are so-called *structural rules.* The side-conditions of these rules refer to equational derivability in SUB, and they help us to manage α-equivalence and capture-avoiding substitution in the presence of meta-variables.

(**Fr**) lets us introduce fresh atoms into a derivation. The difficulty which this rule addresses is the explicit meta-variables of nominal terms; we need (**Fr**) to add explicit information that the atom is fresh for them, into the freshness context.

Remark 5.3.3. In the beginning of Section 5.2 we commented that we do not allow unknowns of abstraction sort. The reader may note that we *do* allow these in Chapter 4. We imposed this restriction on expressivity because it would allow us to write terms like $\forall X$, $\mathsf{sub}(X, a)$ and $\forall(X[a \mapsto b])$.

This may well be a desirable extension of the system we have presented, and it is not incorrect — but it would have a dramatic effect on the readability of the rules in Figure 5.2. For example, the ($\forall$**L**) and the ($\forall$**R**) rules would have to be generalised to something like

$$\frac{\mathsf{sub}(t,u),\ \Phi \vdash_\vartriangle \Psi}{\forall t,\ \Phi \vdash_\vartriangle \Psi}\,(\forall\mathbf{L}) \qquad \frac{\Phi \vdash_\vartriangle \Psi,\ \mathsf{sub}(t,a)}{\Phi \vdash_\vartriangle \Psi,\ \forall t}\,(\forall\mathbf{R}) \quad (\Delta \vdash a\#\Phi, \Psi).$$

We want (the first version of) one-and-a-halfth-order logic to speak directly to the reader already familiar with first-order logic, so we preferred the less powerful, but more intuitive, version. There are many ways that one-and-a-halfth-order logic can be extended and that is future work.

Example derivations in one-and-a-halfth-order logic can be found in Figures 5.3 and 5.4.

$$\frac{}{\phi,\ \Phi \vdash_{\vartriangle} \Psi,\ \phi}\,(\mathbf{Ax}) \qquad \frac{}{\bot,\ \Phi \vdash_{\vartriangle} \Psi}\,(\bot\mathbf{L})$$

$$\frac{\Phi \vdash_{\vartriangle} \Psi,\ \phi \quad \psi,\ \Phi \vdash_{\vartriangle} \Psi}{\phi \supset \psi,\ \Phi \vdash_{\vartriangle} \Psi}\,(\supset\mathbf{L}) \qquad \frac{\phi,\ \Phi \vdash_{\vartriangle} \Psi,\ \psi}{\Phi \vdash_{\vartriangle} \Psi,\ \phi \supset \psi}\,(\supset\mathbf{R})$$

$$\frac{\phi[a \mapsto t],\ \Phi \vdash_{\vartriangle} \Psi}{\forall[a]\phi,\ \Phi \vdash_{\vartriangle} \Psi}\,(\forall\mathbf{L}) \qquad \frac{\Phi \vdash_{\vartriangle} \Psi,\ \psi}{\Phi \vdash_{\vartriangle} \Psi,\ \forall[a]\psi}\,(\forall\mathbf{R}) \quad (\Delta \vdash a\#\Phi, \Psi)$$

$$\frac{\phi[a \mapsto t'],\ \Phi \vdash_{\vartriangle} \Psi}{t' \approx t,\ \phi[a \mapsto t],\ \Phi \vdash_{\vartriangle} \Psi}\,(\approx\mathbf{L}) \qquad \frac{}{\Phi \vdash_{\vartriangle} \Psi,\ t \approx t}\,(\approx\mathbf{R})$$

$$\frac{\phi',\ \Phi \vdash_{\vartriangle} \Psi}{\phi,\ \Phi \vdash_{\vartriangle} \Psi}\,(\mathbf{StructL}) \quad (\Delta \vdash_{\mathsf{SUB}} \phi' = \phi)$$

$$\frac{\Phi \vdash_{\vartriangle} \Psi,\ \psi'}{\Phi \vdash_{\vartriangle} \Psi,\ \psi}\,(\mathbf{StructR}) \quad (\Delta \vdash_{\mathsf{SUB}} \psi' = \psi)$$

$$\frac{\Phi \vdash_{\vartriangle} \Psi,\ \phi \qquad \phi',\ \Phi \vdash_{\vartriangle} \Psi}{\Phi \vdash_{\vartriangle} \Psi}\,(\mathbf{Cut}) \quad (\Delta \vdash_{\mathsf{SUB}} \phi = \phi')$$

$$\frac{\Phi \vdash_{\vartriangle, a\#X_1, \ldots, a\#X_n} \Psi}{\Phi \vdash_{\vartriangle} \Psi}\,(\mathbf{Fr}) \quad (n \geq 1,\ a \notin \Phi, \Psi, \Delta)$$

Figure 5.2 Sequent calculus for one-and-a-halfth-order logic

$$\dfrac{\dfrac{\dfrac{}{Q, P \vdash_{\emptyset} P}\,(\mathbf{Ax})}{P \vdash_{\emptyset} Q \supset P}\,(\supset\mathbf{R})}{\vdash_{\emptyset} P \supset (Q \supset P)}\,(\supset\mathbf{R})$$

$$\dfrac{\dfrac{}{P \vdash_{a\#P} P}\,(\mathbf{Ax})}{P \vdash_{a\#P} P[a \mapsto T]}\,(\mathbf{StructR}) \quad (a\#P \vdash_{\mathsf{SUB}} P = P[a \mapsto T])$$

$$\dfrac{\dfrac{}{\forall[a]P \vdash_{b\#P} \forall[a]P}\,(\mathbf{Ax})}{\forall[a]P \vdash_{b\#P} \forall[b](P[a \mapsto b])}\,(\mathbf{StructR}) \quad (b\#P \vdash_{\mathsf{SUB}} \forall[a]P = \forall[b](P[a \mapsto b]))$$

$$\dfrac{\dfrac{}{P \vdash_{a\#P} P}\,(\mathbf{Ax})}{P \vdash_{a\#P} \forall[a]P}\,(\forall\mathbf{R}) \quad (a\#P \vdash a\#P)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{P, Q \vdash_{a\#P, b\#P, Q} P}\,(\mathbf{Ax})}{P, Q \vdash_{a\#P, b\#P, Q} \forall[b]P}\,(\forall\mathbf{R}) \quad (a\#P, b\#P, Q \vdash b\#P, Q)}{P, Q \vdash_{a\#P, b\#P, Q} \forall[a]P}\,(\mathbf{StructR}) \quad (a\#P, b\#P, Q \vdash_{\mathsf{SUB}} \forall[b]P = \forall[a]P)}{P, Q \vdash_{a\#P} \forall[a]P}\,(\mathbf{Fr}) \quad (b \notin P,\ Q,\ \forall[a]P,\ a\#P)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash_{a\#P} Q, P}\,(\mathbf{Ax}) \qquad \dfrac{}{Q, P \vdash_{a\#P} Q}\,(\mathbf{Ax})}{P, P \supset Q \vdash_{a\#P} Q}\,(\supset\mathbf{L})}{P, (P \supset Q)[a \mapsto a] \vdash_{a\#P} Q}\,(\mathbf{StructL}) \quad (a\#P \vdash_{\mathsf{SUB}} P \supset Q = (P \supset Q)[a \mapsto a])}{P, \forall[a](P \supset Q) \vdash_{a\#P} Q}\,(\forall\mathbf{L})}{P, \forall[a](P \supset Q) \vdash_{a\#P} \forall[a]Q}\,(\forall\mathbf{R}) \quad (a\#P \vdash a\#P,\ \forall[a](P \supset Q))$$

Figure 5.3 Example derivations in one-and-a-halfth-order logic

We give some intuitions to the example derivations of Figures 5.3 and 5.4:

- $\vdash P \supset (Q \supset P)$ represents a family of tautologies $\phi \supset (\psi \supset \phi)$ of propositional logic. The only difference is that here we are using the provision of explicit meta-variables to represent this family directly as a single sequent.
- $P \vdash_{\mathsf{a\#P}} P[a \mapsto T]$ expresses a property of capture-avoiding substitution... with meta-variables. The condition $a\#P$ intuitively guarantees that whatever formula P represents, it is not one that mentions a free in its syntax. It corresponds to writing $a \notin fv(\phi)$.
  The derivation exploits the power to prove equalities in SUB.
- $\forall[a]P \vdash_{\mathsf{b\#P}} \forall[b](P[a \mapsto b])$ expresses α-equivalence... with meta-variables. As in the previous derivation, this derivation exploits the power to prove equalities in SUB.
- $P \vdash_{\mathsf{a\#P}} \forall[a]P$ represents a family of tautologies of predicate logic. We use the freshness assumption $a\#P$ in the instance of the ($\forall\mathbf{R}$) rule.
- $P, Q \vdash_{\mathsf{a\#P}} \forall[a]P$ expresses the same as the previous example, except that we have an additional assumption Q. The derivation becomes significantly more complex: we cannot use ($\forall\mathbf{R}$) on $\forall[a]P$ because we do not know $a\#Q$. The solution is to use (**Fr**) to generate $b\#P, Q$, use structural rules (**StructL**) and (**StructR**) to α-rename, and *then* use ($\forall\mathbf{R}$).
  This use of (**Fr**) is essential: we need this mechanism to introduce a fresh atom into the derivation. This relates to the discussion in Subsection 2.3.2 on the extra power that the (**fr**) rule gives to nominal algebra.
- $P, \forall[a](P \supset Q) \vdash_{\mathsf{a\#P}} \forall[a]Q$ represents another family of tautologies of predicate logic [GM02, axiom (2a) on page 33]. For the instance of ($\forall\mathbf{R}$) to be valid we must show $a\#\forall[a](P \supset Q)$. We have made no assumptions about what is fresh for Q, but the abstraction by a guarantees this property anyway.
- $\forall[b]\forall[a]P \vdash \forall[a](P[b \mapsto a])$ is a relatively non-trivial tautology which might be written in semi-formal notation as '$\forall a.\forall b.\phi(a,b)$ implies $\forall a.\phi(a,a)$'.

We still need to verify the side-conditions from Figures 5.3 and 5.4. The side-conditions on freshness and non-occurrence of terms are easy, since their derivations are completely syntax-directed. In order to verify the side-conditions on equality in SUB we could provide full derivations... but why not reuse the results on decidability of SUB from Chapter 4?

Recall substitution ς from Definition 4.6.8 that maps possibly open terms to closed terms (depending on atoms and freshness information in the context). Also recall the notation $t^{\mathbb{I}}$ for the translation of a closed term t to a ground term (a term that does not mention unknowns or explicit substitutions) from Definition 4.4.4. By Corollary 4.6.20, checking derivability of $\Delta \vdash_{\mathsf{SUB}} t = u$ is equivalent to checking derivability of $\Delta \vdash_{\mathsf{CORE}} t\varsigma^{\mathbb{I}} = u\varsigma^{\mathbb{I}}$, which we can easily do using the syntactic criteria of CORE-equality (Corollary 2.5.4).

As an example we show derivability of a side-condition of Figure 5.4. All the

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P[b\mapsto c][a\mapsto c] \vdash_{c\#P} P[b\mapsto c][a\mapsto c]}\,(\mathbf{Ax})}{\forall[a](P[b\mapsto c]) \vdash_{c\#P} P[b\mapsto c][a\mapsto c]}\,(\forall\mathbf{L})}{(\forall[a]P)[b\mapsto c] \vdash_{c\#P} P[b\mapsto c][a\mapsto c]}\,(\mathbf{StructL})\quad(1.)}{\forall[b]\forall[a]P \vdash_{c\#P} P[b\mapsto c][a\mapsto c]}\,(\forall\mathbf{L})}{\forall[b]\forall[a]P \vdash_{c\#P} \forall[c](P[b\mapsto c][a\mapsto c])}\,(\forall\mathbf{R})\quad(2.)}{\forall[b]\forall[a]P \vdash_{c\#P} \forall[a](P[b\mapsto a])}\,(\mathbf{StructR})\quad(3.)}{\forall[b]\forall[a]P \vdash_{\emptyset} \forall[a](P[b\mapsto a])}\,(\mathbf{Fr})\quad(4.)$$

Side-conditions:

(1.) $(c\#P \vdash_{\mathsf{SUB}} \forall[a](P[b\mapsto c]) = (\forall[a]P)[b\mapsto c])$
(2.) $(c\#P \vdash c\#\forall[b]\forall[a]P)$
(3.) $(c\#P \vdash_{\mathsf{SUB}} \forall[c](P[b\mapsto c][a\mapsto c]) = \forall[a](P[b\mapsto a]))$
(4.) $(c \notin \forall[b]\forall[a]P,\ \forall[a](P[b\mapsto a]))$

Figure 5.4 Another example derivation in one-and-a-halfth-order logic

other cases are similar.

Example 5.3.4. Suppose we need to show

$$c\#P \vdash_{\mathsf{SUB}} \forall[c](P[b\mapsto c][a\mapsto c]) = \forall[a](P[b\mapsto a]).$$

By Corollary 4.6.20, it suffices to show

$$\vdash_{\mathsf{CORE}} (\forall[c](P[b\mapsto c][a\mapsto c]))\varsigma^{\mathrm{I}} = (\forall[a](P[b\mapsto a]))\varsigma^{\mathrm{I}}.$$

Here ς maps the unknown P to the term $\mathsf{d}_P(a,b)$, since a and b are all atoms mentioned in the proof obligation that might not be fresh for P. Then:

$$\begin{array}{rcccc} (\forall[c](P[b\mapsto c][a\mapsto c]))\varsigma^{\mathrm{I}} & \equiv & \forall[c](\mathsf{d}_P(a,b)[c/b][c/a]) & \equiv & \forall[c]\mathsf{d}_P(c,c) \\ (\forall[a](P[b\mapsto a]))\varsigma^{\mathrm{I}} & \equiv & \forall[a](\mathsf{d}_P(a,b)[a/b]) & \equiv & \forall[a]\mathsf{d}_P(a,a) \end{array}$$

So we must show

$$\vdash_{\mathsf{CORE}} \forall[c]\mathsf{d}_P(c,c) = \forall[a]\mathsf{d}_P(a,a),$$

which is easy using the syntactic criteria of CORE-equality (Corollary 2.5.4).

5.4 Proof-Theoretical Results

This section shows two important properties of the sequent calculus for one-and-a-halfth order logic:

- In derivations, atoms may be *permuted* and unknowns may be *instantiated*. We will call these properties equivariance and substitution.
- The *cut-elimination* property of first-order predicate logic is preserved by the extension to one-and-a-halfth-order logic.

5.4.1 Equivariance and Substitution

Definition 5.4.1. We extend notation for permutation and substitution actions t^π, $\pi \cdot t$ and $t\sigma$ to formula contexts Φ, writing Φ^π, $\pi \cdot \Phi$ and $\Phi\sigma$ for the result of applying the actions to the terms in the syntax of Φ.

Write Π^π for the derivation obtained from a sequent derivation Π by replacing every sequent $\Phi \vdash_\Delta \Psi$ by $\Phi^\pi \vdash_{\Delta^\pi} \Psi^\pi$.

Theorem 5.4.2 (Meta-level equivariance)**.** *Suppose that* Π *is a valid derivation of* $\Phi \vdash_\Delta \Psi$*. Then* Π^π *is a valid derivation of* $\Phi^\pi \vdash_{\Delta^\pi} \Psi^\pi$*.*

Proof. Recall that Π is a tree labelled with (nominal algebra) syntax, Δ is a set of syntax, and Φ, and Ψ are syntax. These may all mention atoms. Recall also that the notion of 'valid derivation' depends on a nominal algebra theory SUB, and that SUB is syntax too (a set of axioms) which mention two atoms a and b. The statement

'Π is a valid derivation of $\Phi \vdash_\Delta \Psi$'

can be expressed by a ZFA predicate (Appendix A) with parameters Δ, Π, Φ, Ψ, and the axioms in SUB.[1]

By ZFA equivariance (Theorem A.2.5) validity is invariant under permuting atoms in the parameters. By a remarkable coincidence we have characterised this permutation action inductively for the data we care about; this is exactly -^π. So if the statement above is true, then so is

'Π^π is a valid derivation of $\Phi^\pi \vdash_{\Delta^\pi} \Psi^\pi$',

where we use SUB^π to check side-conditions.

By Lemma 2.4.4 derivability in SUB is identical to derivability in SUB^π. The result follows. ☐

Some terminology is useful for Theorems 5.4.5 and 5.4.7.

Definition 5.4.3. Suppose Π is a derivation. It may use the rule (**Fr**), which introduces fresh atoms into the derivation (if we read the derivation bottom-up), or which garbage-collects fresh atoms in the derivation (if we read the derivation top-down). Call these the **fresh atoms** of Π.

We will use the notion of fresh atoms to avoid accidental name clashes in the next theorems.

[1] It can be expressed by other ZFA predicates; we can choose whichever suits us best.

Definition 5.4.4. Write $\pi \cdot \Pi$ for the derivation obtained from Π by replacing each sequent $\Phi \vdash_{\Delta} \Psi$ by $\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi$.

Theorem 5.4.5 (Object-level equivariance)**.** *Suppose that π is a permutation and suppose that Π is a valid derivation of $\Phi \vdash_{\Delta} \Psi$ whose fresh atoms are disjoint from the atoms in π.*

Then $\pi \cdot \Pi$ is a valid derivation of $\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi$.

Proof. By induction on Π. Base cases (**Ax**) and ($\bot$**L**) are direct. We consider the inductive cases in turn:

- The case of ($\supset$**R**): Suppose $\Phi \vdash_{\Delta} \Psi,\ \phi \supset \psi$ is derived using ($\supset$**R**). Then there exists a derivation Π' of $\phi, \Phi \vdash_{\Delta} \Psi, \psi$, and by the inductive hypothesis $\pi \cdot \Pi'$ is a derivation of $\pi \cdot \phi,\ \pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi,\ \pi \cdot \psi$. Then

$$\dfrac{\begin{array}{c}\vdots\ \pi \cdot \Pi' \\ \pi \cdot \phi,\ \pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi,\ \pi \cdot \psi\end{array}}{\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi,\ \pi \cdot \phi \supset \pi \cdot \psi}\ (\supset\mathbf{R})$$

  is the required derivation $\pi \cdot \Pi$ of $\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi,\ \pi \cdot \phi \supset \pi \cdot \psi$.
  The cases ($\supset$**L**), ($\forall$**L**), ($\approx$**L**) and ($\approx$**R**) are similar.
- The case of ($\forall$**R**): Suppose $\Phi \vdash_{\Delta} \Psi,\ \forall[a]\psi$ is derived using ($\forall$**R**). Then $\Delta \vdash a\#\Phi, \Psi$ holds and Π' is a derivation of $\Phi \vdash_{\Delta} \Psi, \psi$. By object-level equivariance (Theorem 2.4.6) $\Delta \vdash \pi(a)\#\pi \cdot \Phi,\ \pi \cdot \Psi$ holds, and by inductive hypothesis $\pi \cdot \Pi'$ is a derivation of $\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi,\ \pi \cdot \psi$. We conclude that $\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi,\ \forall[\pi(a)]\pi \cdot \psi$ is derivable by extending $\pi \cdot \Pi'$ with ($\forall$**R**), as required.
  The cases (**StructL**), (**StructR**) and (**Cut**) are similar.
- The case of (**Fr**): Suppose $\Phi \vdash_{\Delta} \Psi$ is derived using (**Fr**). Then there exists a derivation Π' of $\Phi \vdash_{\Delta, a\#X_1,\ldots,a\#X_n} \Psi$ where $a \notin \Phi, \Psi, \Delta$. By the inductive hypothesis $\pi \cdot \Pi'$ is a derivation of $\pi \cdot \Phi \vdash_{\Delta, a\#X_1,\ldots,a\#X_n} \pi \cdot \Psi$. By assumption a is disjoint from the atoms in π, so $\pi(a) = a$. Then $a \notin \pi \cdot \Phi, \pi \cdot \Psi, \Delta$ and we conclude $\pi \cdot \Phi \vdash_{\Delta} \pi \cdot \Psi$ using (**Fr**).

□

The assumption on fresh atoms in Theorem 5.4.5 to avoid accidental clashes is not a real restriction. When this assumption cannot be satisfied directly, we can apply meta-level equivariance (Theorem 5.4.2) to rename the fresh atoms to avoid the unfortunate clash, while retaining all structural properties including inductive hypotheses.

Definition 5.4.6. Write $\Pi(\sigma, \Delta')$ for the derivation obtained from Π bottom-up in the syntax as follows:

- If Π concludes with an instance of a rule (**R**) different from (**Fr**), replace its conclusion $\Phi \vdash_{\Delta} \Psi$ by $\Phi\sigma \vdash_{\Delta'} \Psi\sigma$, and replace the derivation Π' of each premise with $\Pi'(\sigma, \Delta')$.
- If Π concludes with an instance of (**Fr**), let Π' be a derivation of its premise $\Phi \vdash_{\Delta, a\#X_1,\ldots,a\#X_n} \Psi$, let $Y_1, \ldots, Y_m$ be all unknowns mentioned in $\sigma(X_i)$ for $1 \leq i \leq n$, and let $\Delta'' = \Delta', a\#Y_1, \ldots, a\#Y_m$. Then:
  - if $m \geq 1$, replace the conclusion $\Phi \vdash_{\Delta} \Psi$ of Π by $\Phi\sigma \vdash_{\Delta'} \Psi\sigma$, and replace Π' with $\Pi'(\sigma, \Delta'')$.
  - if $m = 0$, replace Π with $\Pi'(\sigma, \Delta'')$ (or $\Pi'(\sigma, \Delta')$, since $\Delta'' = \Delta'$).

So σ is consistently applied throughout the formula contexts occurring in Π, Δ' replaces Δ, and (**Fr**) may generate slightly different freshness assumptions (when the new set of freshness assumptions is empty, (**Fr**) is removed).

Theorem 5.4.7 (Meta-level substitution)**.** *Suppose that* $\Delta' \vdash \Delta\sigma$*, that* Π *is a valid derivation of* $\Phi \vdash_{\Delta} \Psi$*, and that the fresh atoms in* Π *are disjoint from the atoms in* σ *and* Δ'*.*

Then $\Pi(\sigma, \Delta')$ *is a valid derivation of* $\Phi\sigma \vdash_{\Delta'} \Psi\sigma$*.*

Proof. By induction on Π. The proof is similar to the proof of object-level equivariance (Theorem 5.4.5).

The cases ($\forall$**R**), (**StructL**), (**StructR**) and (**Cut**) use meta-level substitution on freshness and equality (Theorem 2.4.10).

We treat the case of (**Fr**) in more detail. Suppose $\Phi \vdash_{\Delta} \Psi$ is derived using (**Fr**). Then Π' is a derivation of $\Phi \vdash_{\Delta, a\#X_1,\ldots,a\#X_n} \Psi$ where $a \notin \Phi, \Psi, \Delta$.

Let $Y_1, \ldots, Y_m$ be all the unknowns mentioned in $\sigma(X_i)$, $1 \leq i \leq n$, and let $\Delta'' = \Delta', a\#Y_1, \ldots, a\#Y_m$.

It is not hard to verify that $\Delta'' \vdash (\Delta, a\#X_1, \ldots, a\#X_n)\sigma$, so by the inductive hypothesis $\Pi'(\sigma, \Delta'')$ is a valid derivation of $\Phi\sigma \vdash_{\Delta''} \Psi\sigma$.

We proceed by case distinction on m:

- Suppose $m \geq 1$. By assumption, a is disjoint from the atoms mentioned in σ, Δ'. Then $a \notin \Phi\sigma, \Psi\sigma, \Delta'$ and we may extend $\Pi'(\sigma, \Delta'')$ with (**Fr**) to obtain our required derivation $\Pi(\sigma, \Delta')$ of $\Phi\sigma \vdash_{\Delta'} \Psi\sigma$.
- Suppose $m = 0$. By definition $\Pi(\sigma, \Delta'')$ is $\Pi'(\sigma, \Delta')$. Since $\Delta'' = \Delta'$, it is a derivation of $\Phi\sigma \vdash_{\Delta'} \Psi\sigma$, as required.

□

A useful corollary of Theorem 5.4.7 is the following:

Corollary 5.4.8. *If* $\Phi \vdash_{\emptyset} \Psi$ *is derivable for closed* Φ *and* Ψ*, then there is a derivation of* $\Phi \vdash_{\emptyset} \Psi$ *that does not mention unknowns.*

Proof. Suppose Π is a derivation of $\Phi \vdash_{\emptyset} \Psi$, which possibly mentions unknowns. Let σ be the substitution that maps all unknowns in the derivation to closed terms as follows:

- each unknown P of sort $\mathbb{F}$ is mapped to $\bot$;
- each unknown T of sort $\mathbb{T}$ is mapped to c, where c is an atom that does not occur anywhere in Π.

By Theorem 5.4.7, $\Pi(\sigma, \emptyset)$ is a valid derivation of $\Phi \vdash_{\emptyset} \Psi$. This derivation does not mention unknowns, as can be verified by an easy induction on the structure of Definition 5.4.6. $\square$

5.4.2 Cut-Elimination

Definition 5.4.9. Call the **depth** of a derivation the greatest number of derivation steps not counting rules (**Fr**), (**StructL**) and (**StructR**) between its conclusion and its leaves, over all paths. We do not count nominal algebra derivations of freshnesses and equalities that occur as side-conditions.

For example, the last derivation of Figure 5.3 and the derivation of Figure 5.4 both have depth 4.

The following results are not normally problematic but we have internalised both α-equivalence *and* being fresh — so renaming and freshening must be represented in the derivation.

Lemma 5.4.10 (Freshness weakening)**.** *If $\Phi \vdash_{\Delta} \Psi$ and $\Delta \subseteq \Delta'$ then $\Phi \vdash_{\Delta'} \Psi$. The derivation has the same depth as the original one, and no more instances of cut.*

Proof. By induction on the structure of the derivation. For the cases of ($\forall$**R**), (**StructL**), (**StructR**) and (**Cut**) we use weakening (Corollary 2.4.12) on the side-conditions. For the case of (**Fr**) we use ZFA equivariance. $\square$

Lemma 5.4.11. *If $a \notin u$ and $a\#X \subseteq \Delta$ for each $X \in u$, then $\Delta \vdash a\#u$.*

Lemma 5.4.12 (Formula weakening)**.** *If $\Phi \vdash_{\Delta} \Psi$ and $\Phi \subseteq \Phi'$ and $\Psi \subseteq \Psi'$ then $\Phi' \vdash_{\Delta} \Psi'$. The new derivation has the same depth as the original one, and no more instances of cut.*

Proof. We work by strong induction on the *pair* of the depth of the derivation and its structure, lexicographically ordered. The conditions on preserving depth and number of cuts can easily be verified from the structure of the reasoning which follows, and we do not mention them further.

- The case of (**StructL**): Suppose $\phi, \Phi \vdash_{\Delta} \Psi$ is derived using (**StructL**), and assume the inductive hypothesis on all strictly lesser derivations. So $\phi', \Phi \vdash_{\Delta} \Psi$ and $\Delta \vdash_{\mathsf{SUB}} \phi' = \phi$ are derivable for some ϕ'. This derivation has the same depth as, and a lesser structure than that of $\phi, \Phi \vdash_{\Delta} \Psi$, so we may use the inductive hypothesis to derive $\phi', \Phi' \vdash_{\Delta} \Psi'$. By (**StructL**) we obtain $\phi, \Phi' \vdash_{\Delta} \Psi'$ as required.

- The case of $(\forall\mathbf{R})$: Suppose $\Phi \vdash_{\Delta} \Psi,\ \forall[a]\psi$ is derived using $(\forall\mathbf{R})$ and suppose the inductive hypothesis of all strictly lesser derivations.
  By assumption $\Phi \vdash_{\Delta} \Psi, \psi$ has a derivation Π of strictly lesser depth, and also $\Delta \vdash a\#\Phi, \Psi$ holds. Choose a' fresh (so $a' \not\in a, \Phi', \Psi', \Pi$) and $\Delta' = \Delta, a'\#\mathcal{X}$ where $\mathcal{X}$ are the unknowns mentioned in Φ',Ψ',Δ and ψ. Then $\Delta' \vdash a\#\Phi, \Psi$ by weakening (Corollary 2.4.12), and $\Phi \vdash_{\Delta'} \Psi,\ \psi$ by freshness weakening (Lemma 5.4.10).[2]
  a' is not in Π by assumption, so in particular a' is not a fresh atom of Π. We may also assume that a is not a fresh atom in Π — if a is generated by $(\mathbf{Fr})$ somewhere in Π then we easily build another derivation where the instance of $(\mathbf{Fr})$ in question generates a different (and 'fresher') atom. This affects neither depth nor structure so we retain the inductive hypothesis.
  Then by object-level equivariance (Theorem 5.4.5) also
  $$(a'\ a)\cdot\Phi \vdash_{\Delta'} (a'\ a)\cdot\Psi,\ (a'\ a)\cdot\psi.$$
  Using $(\mathbf{perm})$ in $(\mathbf{StructL})$ and $(\mathbf{StructR})$ we obtain $\Phi \vdash_{\Delta'} \Psi,\ (a'\ a)\cdot\psi$. By inductive hypothesis (the derivation still has strictly lesser depth) there exists a derivation Π' of $\Phi' \vdash_{\Delta'} \Psi',\ (a'\ a)\cdot\psi$. Furthermore, $\Delta' \vdash a'\#\Phi', \Psi'$ by Lemma 5.4.11, and we observe $\Delta' \vdash_{\mathsf{SUB}} \forall[a'](a'\ a)\cdot\psi = \forall[a]\psi$ by simple calculations (we use $(\mathbf{perm})$, and the freshness information of a').
  Now we can conclude $\Phi' \vdash_{\Delta} \Psi',\ \forall[a]\psi$ as follows:
  $$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ \Pi'}{\Phi' \vdash_{\Delta'} \Psi',\ (a'\ a)\cdot\psi}}{\Phi' \vdash_{\Delta'} \Psi',\ \forall[a'](a'\ a)\cdot\psi}\,(\forall\mathbf{R})}{\Phi' \vdash_{\Delta'} \Psi',\ \forall[a]\psi}\,(\mathbf{StructR})}{\Phi' \vdash_{\Delta} \Psi',\ \forall[a]\psi}\,(\mathbf{Fr})$$
- The case of $(\mathbf{Fr})$: Suppose $\Phi \vdash_{\Delta, a\#X_1,\ldots,a\#X_n} \Psi$ where $a \not\in \Phi, \Psi, \Delta$. We use ZFA equivariance (Theorem A.2.5) to rename a to some $a' \not\in \Phi', \Psi', \Delta$ in the whole derivation to obtain one of $\Phi \vdash_{\Delta, a'\#X_1,\ldots,a'\#X_n} \Psi$. We can now apply the inductive hypothesis (which, as discussed above, by ZFA equivariance is preserved by the permutative renaming) to weaken to Φ' and Ψ', and finish off with $(\mathbf{Fr})$.

The other cases are easy or similar. □

Recall the classical logic sugar from Definition 5.2.7.

Corollary 5.4.13 (Admissible rules)**.** *The rules of Figure 5.5 are admissible.*

[2]It appears convenient to prove freshness weakening first separately; we do not want to weaken Φ and Ψ to Φ' and Ψ' until we have renamed a to a', in a moment.

$$\frac{}{\Phi \vdash_{\vartriangle} \Psi, \top}\,(\top\mathbf{R})$$

$$\frac{\Phi \vdash_{\vartriangle} \Psi,\ \phi}{\neg\phi,\ \Phi \vdash_{\vartriangle} \Psi}\,(\neg\mathbf{L}) \qquad \frac{\psi,\ \Phi \vdash_{\vartriangle} \Psi}{\Phi \vdash_{\vartriangle} \Psi,\ \neg\psi}\,(\neg\mathbf{R})$$

$$\frac{\phi,\ \phi',\ \Phi \vdash_{\vartriangle} \Psi}{\phi \wedge \phi',\ \Phi \vdash_{\vartriangle} \Psi}\,(\wedge\mathbf{L}) \qquad \frac{\Phi \vdash_{\vartriangle} \Psi,\ \psi \qquad \Phi \vdash_{\vartriangle} \Psi,\ \psi'}{\Phi \vdash_{\vartriangle} \Psi,\ \psi \wedge \psi'}\,(\wedge\mathbf{R})$$

$$\frac{\phi,\ \Phi \vdash_{\vartriangle} \Psi \qquad \phi',\ \Phi \vdash_{\vartriangle} \Psi}{\phi \vee \phi',\ \Phi \vdash_{\vartriangle} \Psi}\,(\vee\mathbf{L}) \qquad \frac{\Phi \vdash_{\vartriangle} \Psi,\ \psi,\ \psi'}{\Phi \vdash_{\vartriangle} \Psi,\ \psi \vee \psi'}\,(\vee\mathbf{R})$$

$$\frac{\Phi \vdash \Psi,\ \phi,\ \phi' \quad \phi,\ \phi',\ \Phi \vdash_{\vartriangle} \Psi}{\phi \Leftrightarrow \phi',\ \Phi \vdash_{\vartriangle} \Psi}\,(\Leftrightarrow\mathbf{L}) \qquad \frac{\psi,\ \Phi \vdash_{\vartriangle} \Psi,\ \psi' \quad \psi',\ \Phi \vdash_{\vartriangle} \Psi,\ \psi}{\Phi \vdash \Psi,\ \psi \Leftrightarrow \psi'}\,(\Leftrightarrow\mathbf{R})$$

$$\frac{\phi,\ \Phi \vdash_{\vartriangle} \Psi}{\exists[a]\phi,\ \Phi \vdash_{\vartriangle} \Psi}\,(\exists\mathbf{L}) \quad (\Delta \vdash a\#\Phi, \Psi) \qquad \frac{\Phi \vdash_{\vartriangle} \Psi,\ \phi[a \mapsto t]}{\Phi \vdash_{\vartriangle} \Psi,\ \exists[a]\phi}\,(\exists\mathbf{R})$$

Figure 5.5 Admissible sequent rules for one-and-a-halfth-order logic

Proof. We consider just the case of $(\neg\mathbf{R})$. Suppose we have derived $\psi,\ \Phi \vdash_{\vartriangle} \Psi$. Then by formula weakening (Lemma 5.4.12) there also exists a derivation of $\psi,\ \Phi \vdash_{\vartriangle} \Psi,\ \bot$. Extending that derivation with $(\supset\mathbf{R})$ we obtain a derivation of $\Phi \vdash_{\vartriangle} \Psi,\ \neg\psi$ as required.

The cases $(\wedge\mathbf{R})$, $(\vee\mathbf{L})$, $(\Leftrightarrow\mathbf{L})$, $(\Leftrightarrow\mathbf{R})$ and $(\exists\mathbf{L})$, are similar. Remaining cases are by directly extending derivations. In the case of $(\exists\mathbf{R})$, we use (**StructL**) to replace the $(\neg\phi)[a \mapsto t]$ by $\neg(\phi[a \mapsto t])$. □

Some of the admissible rules will turn out useful in Chapter 6 (Section 6.3).

Definition 5.4.14. Write $\Phi[a \mapsto t]$ for the elementwise application of the substitution to the elements of formula context Φ.

Lemma 5.4.15 (Object-level substitution)**.** *For any a and t, if $\Phi \vdash_{\vartriangle} \Psi$ then $\Phi[a \mapsto t] \vdash_{\vartriangle} \Psi[a \mapsto t]$. The depth of the derivation does not increase, and neither does the number of cuts it contains.*

Proof. Analogous to the proof of formula weakening (Lemma 5.4.12). □

Lemma 5.4.16. (**Fr**) *may be commuted down through all other rules. This does not increase the depth of a derivation or its number of cuts.*

Proof. We consider one typical case. Suppose (**Fr**) is followed by (⊃**L**) as follows:

$$\dfrac{\dfrac{\begin{matrix}\vdots\,\Pi_1\\ \Phi \vdash_{\Delta, a\#X_1,\dots,a\#X_n} \Psi,\, \phi\end{matrix}}{\Phi \vdash_{\Delta} \Psi,\, \phi}\,(\mathbf{Fr}) \qquad \begin{matrix}\vdots\,\Pi_2\\ \psi,\, \Phi \vdash_{\Delta} \Psi\end{matrix}}{\phi \supset \psi,\, \Phi \vdash_{\Delta} \Psi}\,({\supset}\mathbf{L})$$

Here $a \notin \Phi, \Delta, \Psi, \phi$.

Suppose we are unlucky and a is mentioned in ψ. Choose a fresh atom a' (i.e. $a' \notin \Phi, \Delta, a, \Psi, \phi, \psi$). By meta-level equivariance (Theorem 5.4.2), $\Pi_1^{(a'\ a)}$ is a valid derivation of $\Phi \vdash_{\Delta, a'\#X_1,\dots,a'\#X_n} \Psi,\, \phi$. Also, by freshness weakening (Lemma 5.4.10), there is a derivation Π_2' of $\psi,\, \Phi \vdash_{\Delta, a'\#X_1,\dots,a'\#X_n} \Psi$.

We can now put our derivation together:

$$\dfrac{\dfrac{\begin{matrix}\vdots\,\Pi_1^{(a'\ a)}\\ \Phi \vdash_{\Delta, a'\#X_1,\dots,a'\#X_n} \Psi,\, \phi\end{matrix} \qquad \begin{matrix}\vdots\,\Pi_2'\\ \psi,\, \Phi \vdash_{\Delta, a'\#X_1,\dots,a'\#X_n} \Psi\end{matrix}}{\phi \supset \psi,\, \Phi \vdash_{\Delta, a'\#X_1,\dots,a'\#X_n} \Psi}\,({\supset}\mathbf{L})}{\phi \supset \psi,\, \Phi \vdash_{\Delta} \Psi}\,(\mathbf{Fr})$$

The new derivation preserves the depth and number of cuts of the original derivation, since $\Pi_1^{(a'\ a)}$ does so by definition and Π_2' does so by Lemma 5.4.10.

All other cases are similar or simpler. □

Theorem 5.4.17 (Cut-elimination). *If $\Phi \vdash_{\Delta} \Psi$ is derivable in the sequent calculus for one-and-a-halfth-order logic then there exists a derivation of $\Phi \vdash_{\Delta} \Psi$ which does not mention* (**Cut**).

Proof. The commutation cases and essential cases are standard [Gen35, Pra65]; we use formula weakening (Lemma 5.4.12) for the essential case for ⊃; the non-standard case of (**Fr**) is handled by Lemma 5.4.16. The essential case for ∀ is handled by object-level substitution (Lemma 5.4.15). □

Corollary 5.4.18 (Consistency). *The sequent calculus of one-and-a-halfth-order logic is consistent, i.e. $\vdash_{\Delta}$ can never be derived.*

Proof. By contradiction. Suppose $\vdash_{\Delta}$ is derivable, then by Theorem 5.4.17 a cut-free derivation exists. Let Π be the shortest derivation of $\vdash_{\Delta}$ for all possible Δ. We check through all possible derivation rules and see by their syntax-directed nature that the derivation must conclude in (**Fr**). But then we have a shorter derivation of some $\vdash_{\Delta'}$, which is a contradiction. □

$$\frac{}{\phi,\ \Phi \vdash \Psi,\ \phi}\,(\mathbf{Ax}) \qquad \frac{}{\bot,\ \Phi \vdash \Psi}\,(\bot\mathbf{L})$$

$$\frac{\Phi \vdash \Psi,\ \phi \quad \psi,\ \Phi \vdash \Psi}{\phi \supset \psi,\ \Phi \vdash \Psi}\,(\supset\mathbf{L}) \qquad \frac{\phi,\ \Phi \vdash \Psi,\ \psi}{\Phi \vdash \Psi,\ \phi \supset \psi}\,(\supset\mathbf{R})$$

$$\frac{\phi[t/a],\ \Phi \vdash \Psi}{\forall a.\phi,\ \Phi \vdash \Psi}\,(\forall\mathbf{L}) \qquad \frac{\Phi \vdash \Psi,\ \phi}{\Phi \vdash \Psi,\ \forall a.\phi}\,(\forall\mathbf{R}) \quad (a \notin \mathit{fa}(\Phi, \Psi))$$

$$\frac{\phi[t'/a],\ \Phi \vdash \Psi}{t' \approx t,\ \phi[t/a],\ \Phi \vdash \Psi}\,(\approx\mathbf{L}) \qquad \frac{}{\Phi \vdash \Psi,\ t \approx t}\,(\approx\mathbf{R})$$

Figure 5.6 Gentzen's sequent calculus for first-order logic

5.5 Relation to First-Order Logic with Equality

Recall from Definition 4.3.1 that we call a term ground when it does not mention unknowns or explicit substitutions. In this section we show how first-order logic can be considered as the fragment of one-and-a-halfth-order logic on ground terms.

Definition 5.5.1. A **Gentzen sequent** is a pair $\Phi \vdash \Psi$ of finite sets of *ground formulae* Φ and Ψ. The **derivable sequents** of Gentzen's sequent calculus for first-order logic are the Gentzen sequents inductively specified by the rules in Figure 5.6.

Here $\phi[t/a]$ is the ground substitution action (see Definition 4.3.5), $\mathit{fa}(\Phi, \Psi)$ stands for the union of all $\mathit{fa}(\phi)$, $\phi \in \Phi, \Psi$, and we write $\forall[a]\phi$ as $\forall a.\phi$. Furthermore, we take formulae up to α-equivalence relation $=_\alpha$ (see Definition 4.3.10).

Taking formulae up to α-equivalence means for example that if $\mathsf{p} : (\mathbb{T})\mathbb{F}$ is a predicate term-former (such as $\mathsf{issocrates}$) then $\forall a.\mathsf{p}(a) \vdash \forall b.\mathsf{p}(b)$ follows directly by (**Ax**) since $\forall a.\mathsf{p}(a) =_\alpha \forall b.\mathsf{p}(b)$.

Note that typically Gentzen's sequent calculus is taken to have a *first-order* term-language, so it doesn't contain binders. We do not (need to) make this choice here; see Subsection 5.6.1 for a more detailed exposition.

Lemma 5.5.2. *For ground Φ, Ψ, if $\Phi \vdash \Psi$ is derivable in Gentzen's sequent calculus then $\Phi \vdash_\emptyset \Psi$ is derivable in the sequent calculus for one-and-a-halfth-order logic.*

Proof. The statement of this lemma is a little bit vague, since we take formulae up to α-equivalence when we define Gentzen style derivability, but we do not take formulae up to α-equivalence in one-and-a-halfth-order logic (we have structural rules (**StructL**) and (**StructR**) instead). We ignore this issue, and suppose that some arbitrary choice of representative closed nominal terms is made for us.

Suppose $\Phi \vdash \Psi$ is derivable in Genzten's sequent calculus. By induction on derivations of $\Phi \vdash \Psi$ we construct a derivation of $\Phi \vdash_{\emptyset} \Psi$. Each rule translates to its one-and-a-halfth-order counterpart, where we note the following:

- If at the meta-level in the Gentzen system we used α-conversion, or just if we wish to change representatives, then we can 'patch' the derivation in one-and-a-halfth-order logic with structural rules (**StructL**) and (**StructR**). The side-conditions follow by Theorem 4.3.13 (on ground terms $t =_\alpha u$ if and only if $\vdash_{\mathsf{CORE}} t = u$), and the fact that derivability in CORE implies derivability in SUB.
- For the case of $(\forall\mathbf{L})$ we need an extra use of (**StructL**) to manage the substitution. The side-condition $\vdash_{\mathsf{SUB}} \phi[a \mapsto t] = \phi[t/a]$ follows directly from Corollary 4.4.3.
  The case of $(\approx\mathbf{L})$ is similar.
- For the case of $(\forall\mathbf{R})$ the side-conditions $\vdash a\#\Phi, \Psi$ follow from the assumption $a \notin fa(\Phi, \Psi)$ by Lemma 4.3.3.

$\square$

Recall the notation $t^{\mathbb{I}}$ from Definition 4.4.4.

Definition 5.5.3. If Φ is a closed formula context, write $\Phi^{\mathbb{I}}$ for the ground formula context $\{\phi^{\mathbb{I}} \mid \phi \in \Phi\}$.

Theorem 5.5.4. *For* closed Φ, Ψ, $\Phi \vdash_{\emptyset} \Psi$ *is derivable in one-and-a-halfth-order logic if and only if* $\Phi^{\mathbb{I}} \vdash \Psi^{\mathbb{I}}$ *is derivable in Gentzen's sequent calculus.*

Proof. For the right-to-left direction, suppose $\Phi^{\mathbb{I}} \vdash \Psi^{\mathbb{I}}$ is derivable in Gentzen's sequent calculus. By Lemma 5.5.2, $\Phi^{\mathbb{I}} \vdash_{\emptyset} \Psi^{\mathbb{I}}$ is derivable in one-and-a-halfth-order logic. By Theorem 4.4.6 and Lemma 4.6.4 we also know that $\vdash_{\mathsf{SUB}} \phi^{\mathbb{I}} = \phi$ for each $\phi \in \Phi, \Psi$. We use this to extend the derivation of $\Phi^{\mathbb{I}} \vdash_{\emptyset} \Psi^{\mathbb{I}}$ with instances of (**StructL**) or (**StructR**) for each $\phi^{\mathbb{I}} \in \Phi^{\mathbb{I}}, \Psi^{\mathbb{I}}$ to obtain one of $\Phi \vdash_{\emptyset} \Psi$, as required.

The left-to-right direction is by induction on derivations of $\Phi \vdash_{\emptyset} \Psi$. By Corollary 5.4.8 we assume that these derivations do not mention unknowns, and by Theorem 5.4.17 we assume that they do not mention (**Cut**).

We consider the rules in turn:

- (**StructL**) and (**StructR**) are *facts*: the side conditions are of the form $\vdash_{\mathsf{SUB}} \phi = \psi$. By decidability of SUB (Corollary 4.6.20) $\vdash_{\mathsf{SUB}} \phi = \psi$ is equivalent to $\vdash_{\mathsf{CORE}} \phi^{\mathbb{I}} = \psi^{\mathbb{I}}$, which is equivalent to $\phi^{\mathbb{I}} =_\alpha \psi^{\mathbb{I}}$ by Theorem 4.3.13.
- (**Fr**) is impossible since the derivation does not mention unknowns.
- The other rules translate directly to their first-order counterparts. For the case of $(\forall\mathbf{R})$ we use the fact that $\vdash a\#\Phi, \Psi$ implies $a \notin fa(\Phi^{\mathbb{I}}, \Psi^{\mathbb{I}})$ (by Lemmas 4.4.5 and 4.3.3).

$\square$

Corollary 5.5.5 (Equivalence of $\vdash_{\emptyset}$ and $\vdash$)**.** *For ground Φ, Ψ, $\Phi \vdash_{\emptyset} \Psi$ is derivable in one-and-a-halfth-order logic, if and only if $\Phi \vdash \Psi$ is derivable in Gentzen's sequent calculus.*

Proof. This is a direct instance of Theorem 5.5.4, since $\Phi^{\mathbb{I}} \equiv \Phi$ and $\Psi^{\mathbb{I}} \equiv \Psi$ when Φ and Ψ are ground. $\square$

5.6 Conclusions

One-and-a-halfth-order logic generalises first-order logic by internalising meta-level variables that range over syntax. Thanks to the use of nominal techniques, the sequent rules from Figure 5.2 accurately reflect common practice in the handling of these meta-variables (compare for instance the examples in the Introduction to the formal derivations in Figures 5.3 and 5.4). As a result, we have been able to import first-order proof theory quite directly into our augmented setting.

5.6.1 Related Work

We discuss first-order logic with binders, second-order and higher-order logics, and deep and shallow embeddings.

First-order logic with binders

Typically Gentzen's sequent calculus is taken to have a *first-order* term-language. Logics exist in the style of first-order logic, but whose terms include binding. One of them is *binding logic* of Dowek et al. [DHK02], which was intended as a general-purpose first-order logic for studying term-languages with binding. Another one is Beeson's lambda logic [Bee04], which was intended for the specific application of manipulating λ-terms in a first-order logic.

The term-language of one-and-a-halfth-order logic natively supports binders: the sorting arities of the object-level term-formers and predicate term-formers allow for abstractions $[a]t$ and $[a]\phi$ as their arguments (see Definition 5.2.3). Therefore the results on Gentzen's sequent calculus we have stated in Section 5.5 can, we think, be extended naturally to many of the 'first-order flavoured' logics with term-formers enriched with binders; the machinery is already there.

Note that one-and-a-halfth-order logic is different from lambda logic and binding logic. For example the syntax used in this chapter includes explicit meta-variables.

Second-order and higher-order logics

Monadic second-order logic enriches first-order logic with variables ranging over sets of elements [Cou97]; these can be identified with function variables of type $i \rightarrow o$ where i is a type of individuals and o a type of truth-values. Second-order logic enriches first-order logic with variables ranging over n-ary relations; these can be identified with function variables of type $i \to \ldots \to i \to o$. Higher-order logic enriches first-order logic with variables ranging over a full type-hierarchy, inductively definable by $\tau ::= i \mid o \mid \tau \to \tau$ [vB01, Sha01, Far06]. Such function variables can be used to represent meta-variables. This representation has some distinctive features inherited from their intended functional semantics, as is explained in the Conclusions of Chapter 2.

A direct comparison between one-and-a-halfth-order logic and second-order logic is not possible. The second-order theorem $\forall P.((\forall P.P) \supset P)$ cannot be expressed in one-and-a-halfth-order logic, because one-and-a-halfth-order logic has no quantification over predicates; this gives it a first-order flavour. On the other hand the one-and-a-halfth-order logic theorem $\forall[a]P \vdash_{\mathsf{a\#P}} P$ cannot be expressed in second-order logic, because that logic cannot directly express freshness conditions. We are not aware of any truly satisfactory account of the precise relation of nominal-style unknowns and higher-order variables.

Deep and shallow embeddings

We can represent the *syntax* of first-order logic in a so-called *framework* logical system, at 'object-level', i.e. as *an inductive datatype* — the so-called *deep embedding*. Then meta-variables can be represented as meta-variables of the framework. This path is taken by higher-order abstract syntax [PE88], Fraenkel-Mostowski syntax [GP02], the theory of contexts [Mic01] and much other research. This enterprise is quite different from that undertaken in this chapter; one-and-a-halfth-order logic is about extending the syntax of the logic itself so it contains something which behaves very much like a meta-variable ranging over unknown formulae, without losing logical properties such as cut-elimination.

It is also possible to represent the *semantics* of first-order logic as a theory in a framework, for example as a pair of types i and o along with functions between them like $\supset : o \to o \to o$ or $\forall : (i \to o) \to o$. This is called a *shallow embedding*.

In [Pau90] the case is made for Isabelle and for its higher-order logic framework as an efficient basis for shallow embeddings, and for conducting mathematics in these embeddings. For example, a shallow embedding of first-order logic called Isabelle/FOL exists in Isabelle's higher-order logic framework Isabelle/Pure. In Chapter 6 we will give a shallow embedding of one-and-a-halfth-order logic in nominal algebra by means of a set of axioms.

5.6.2 Future Work

We believe that there is no technical barrier to creating a variant of one-and-a-halfth-order logic that corresponds to intuitionistic logic [vD02] with meta-variables; it can be defined in the usual way by restricting the sequents in Figure 5.2 to have a single conclusion. Also developing a formal semantics should not pose many problems: evaluate unknowns X to terms and then evaluate atoms a to elements of a set underlying domain.

The following describes some more challenging future work.

Incomplete proofs

We have internalised meta-variables that range over terms and formulae. We can also consider meta-variables ranging over *incomplete proofs*. Incomplete proofs arise naturally in proof-search in a human-assisted theorem-prover such as Isabelle [Pau89] or Coq [HKPM]; see also [Joj04] for a detailed exposition. Here the theorem-prover acts as a program to manipulate proofs which are incomplete both in the sense of having holes, and in the sense of occurring in an unknown context, and the human assistant guides the system to fill in these holes until a complete proof emerges. A quantifier introduction rule binds the quantified variable in the derivation above it (this is usually expressed by a freshness condition). In the presence of incomplete proofs it is necessary to somehow represent this binding over an *as yet unknown* derivation.

We believe that nominal terms with their unknowns and abstractions are a good match for the incomplete proofs and quantifiers binding in it. As the incomplete proof is 'filled in' by the human assistant and the system, a capturing substitution should be made for the unknown. An exciting application of our technology would be an investigation of how well (if at all) a system related to one-and-a-halfth-order logic but with proof-terms can represent this process. This would also provide us with a basis for the implementation of a theorem-prover with some of the generality of second-order logic, but with the flavour of familiar pencil-and-paper schematic derivations in first-order logic.

Hierarchies of variables

We see one-and-a-halfth-order logic as the first of a family of two-level logics as yet to be created. For example we would like to be able to quantify over unknowns anywhere in a formula, which allows us to write an expressions like

$$\forall[X]\forall[a]\big(a\#X \supset (X \Leftrightarrow \forall[a]X)\big).$$

This is current research.

In previous Conclusions (Chapters 2 and 4) we mentioned that we are also interested in logics with infinite hierarchies of meta-variables, such that at each level a meta-variable of higher level behaves to the lower level as an unknown X behaves to an atom a. This might recover some or all of the power which one-and-a-halfth-order logic lacks compared to higher-order logic, but in a different way. In short, we envisage two- three- four- and ω-and-a-halfth-order logic.

Chapter 6

An Axiomatisation of One-and-a-halfth-Order Logic

6.1 Introduction

In Chapter 2 we introduced nominal algebra and in Chapter 5 we introduced one-and-a-halfth-order logic. In this chapter we show how we can easily give one-and-a-halfth-order logic a semantics in nominal algebra.

We will show that theory FOL from Example 2.2.17 is an equational axiomatisation of one-and-a-halfth-order logic when cast in the signature and sort system as described in Section 5.2.

Definition 6.1.1. Let theory FOL have the signature from Definition 5.2.3, and the axioms of Figure 6.1 plus the axioms of theory SUB for one-and-a-halfth-order logic (Figure 5.1).

We discuss the axioms in Figure 6.1:

- We read (**MP**) as 'Modus Ponens'. Modus Ponens is the principle

  "if 'P' is true and 'P implies Q' is true, then 'Q' is true".

  In our algebraic setting the judgement 'is true' is rendered as '$= \top$'.

  Lemma 6.1.2. *If we read 'is true' as '$= \top$' then* (**MP**) *implies Modus Ponens.*

  Proof. Suppose that $P = \top$ and that $P \supset Q = \top$. Then $\top \supset Q = \top$ by the rules for equality. We conclude $Q = \top$ by (**MP**). □

(**MP**)	$\top \supset Q$	$= Q$
(**Mer**)	$((((P \supset Q) \supset (\neg R \supset \neg N)) \supset R) \supset M)$ $\supset ((M \supset P) \supset (N \supset P))$	$= \top$
(**Qinst**)	$\forall[a]P \supset P[a \mapsto T]$	$= \top$
(**Qdist**)	$\forall[a](P \wedge Q) \Leftrightarrow \forall[a]P \wedge \forall[a]Q$	$= \top$
(**Qextr**)	$a\#P \ \vdash \ \forall[a](P \supset Q) \Leftrightarrow P \supset \forall[a]Q$	$= \top$
(**Esubst**)	$U \approx T \wedge P[a \mapsto T] \supset P[a \mapsto U]$	$= \top$
(**Erefl**)	$T \approx T$	$= \top$

Figure 6.1 Axioms of theory FOL

In (**Mer**) recall that $\neg\phi$ is sugar for $\phi \supset \bot$. This axiom by Meredith [Mer53], along with Modus Ponens, is sufficient to derive all rules of classical propositional logic. Expressed as an algebra as above, the two axioms yield an implication-only version of *boolean algebra* [BS81].[1]

- Axioms (**Qinst**), (**Qdist**) and (**Qextr**) add quantifiers; (**Qextr**) exploits freshness conditions.

  These axioms appear in the literature (see e.g. [GM02, (2a) and (2b) on page 33]). What is new here is that our axioms are *not* axiom-schemes; they are *individual axioms* (three, to be precise). Note how these axioms are faithful to the usual syntactic form of the axiom-schemes found in the literature.

- Axioms (**Esubst**) and (**Erefl**) add object-level equality.

  Again, we are able to represent by two axioms what might otherwise be two infinite axiom-schemes.

Overview The rest of the chapter makes the connection between the axioms in Figure 6.1 in the context of nominal algebra, and the sequent rules in Figure 5.2. Sections 6.2 and 6.3 each establish one side of the connection. Section 6.4 glues these parts together and shows a number of useful corollaries of this connection, including consistency of FOL. We conclude in Section 6.5.

6.2 Sequent Derivability Implies FOL Derivability

Definition 6.2.1. Let **classical propositional logic** be the notion of valid sequents inductively defined by the rules (**Ax**), ($\bot$**L**), ($\supset$**L**), and ($\supset$**R**) from Figure 5.2, and removing Δ.

[1]Succinct axioms for propositional logic continue to provide fun; e.g. see [MVF$^+$02, GP06].

Theorem 6.2.2. *For any ϕ, ψ, if $\phi \Leftrightarrow \psi$ is derivable in classical propositional logic, then $\Delta \vdash_{\mathsf{FOL}} \phi = \psi$ in nominal algebra.*

Proof. By machine-checked proofs online [Met], (**MP**) and (**Mer**) suffice to derive all the logical identities of classical propositional logic. □

Corollary 6.2.3. *The following equalities are all derivable in* FOL*:*

$$\begin{array}{llll}
\Delta \vdash_{\mathsf{FOL}} & \phi \lor (\psi \lor \xi) = (\phi \lor \psi) \lor \xi & \Delta \vdash_{\mathsf{FOL}} & \phi \land (\psi \land \xi) = (\phi \land \psi) \land \xi \\
\Delta \vdash_{\mathsf{FOL}} & \phi \lor \psi = \psi \lor \phi & \Delta \vdash_{\mathsf{FOL}} & \phi \land \psi = \psi \land \phi \\
\Delta \vdash_{\mathsf{FOL}} & \phi \lor (\psi \land \phi) = \phi & \Delta \vdash_{\mathsf{FOL}} & \phi \land (\psi \lor \phi) = \phi \\
\Delta \vdash_{\mathsf{FOL}} & \phi \lor (\psi \land \xi) = (\phi \lor \psi) \land (\phi \lor \xi) & \Delta \vdash_{\mathsf{FOL}} & \phi \land (\psi \lor \xi) = (\phi \land \psi) \lor (\phi \land \xi) \\
\Delta \vdash_{\mathsf{FOL}} & \phi \lor \neg\phi = \top & \Delta \vdash_{\mathsf{FOL}} & \phi \land \neg\phi = \bot
\end{array}$$

Proof. The reader will recognise these as the equalities of boolean algebra. It is known that equality in boolean algebra characterises precisely logical equivalence in classical propositional logic [BS81]. By Theorem 6.2.2 the equality of FOL includes equalities between all formulae that are provably logically equivalent in classical propositional logic (it suffices to use (**MP**) and (**Mer**)). The result follows. □

Definition 6.2.4. We say we work by **elementary calculations (in propositional logic)** when we use Corollary 6.2.3 to transform formulae according to standard identities in classical propositional logic.

Lemma 6.2.5. $\vdash_{\mathsf{FOL}} \forall[a]\bot = \bot$ *is derivable.*

Proof. We first derive $\neg\forall[a]\bot = \top$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a\#\bot}\,(\#\mathsf{f})}{\bot[a \mapsto a] = \bot}\,(\mathbf{ax}_{\#\mapsto})}{\bot = \bot[a \mapsto a]}\,(\mathbf{symm})}{(\forall[a]\bot) \supset \bot = (\forall[a]\bot) \supset \bot[a \mapsto a]}\,(\mathbf{congf}) \qquad \dfrac{}{(\forall[a]\bot) \supset \bot[a \mapsto a] = \top}\,(\mathbf{ax_{Qinst}})}{(\forall[a]\bot) \supset \bot = \top}\,(\mathbf{tran})$$

It follows that $\vdash_{\mathsf{FOL}} \neg\neg\forall[a]\bot = \neg\top$ by (**congf**) and so by elementary calculations in propositional logic we obtain $\vdash_{\mathsf{FOL}} \forall[a]\bot = \bot$. □

An informal reading of Lemma 6.2.5 is that any semantics for $\mathbb{T}$ in FOL should be *non-empty*, for if $\mathbb{T}$ were empty then (intuitively) $\forall[a]\bot = \top$, so $\forall[a]\bot = \bot$ should not be derivable.

$\mathbb{T}$ is non-empty because it is populated by atoms (in the derivation above, we use the fact that it is populated by a). Thanks to the substitution action, atoms behave like 'object-level variable symbols'. Normally sorts of terms are populated by variable symbols, but this feature of the syntax does not show in the semantics, and that affects the notion of derivability: $(\forall x.\bot) \Leftrightarrow \top$ may be derivable. We see that in one-and-a-halfth-order logic terms are populated by unknowns *and* atoms. Although atoms represent variable symbols, the derivation above suggests that any semantics for one-and-a-halfth-order logic must differ from a 'standard' semantics, and give atoms denotational reality. One such semantics is given by a standard semantics for nominal algebra in nominal sets (see Chapter 3); developing other denotations-containing-variables is very much of current research interest.

We need some meta-level properties.

Lemma 6.2.6. *For any formulae ϕ, ψ:*

1. $\Delta \vdash_{\mathsf{FOL}} \phi \land \psi = \top$ *if and only if* $\Delta \vdash_{\mathsf{FOL}} \phi = \top$ *and* $\Delta \vdash_{\mathsf{FOL}} \psi = \top$.
2. $\Delta \vdash_{\mathsf{FOL}} \phi \Leftrightarrow \psi = \top$ *if and only if* $\Delta \vdash_{\mathsf{FOL}} \phi = \psi$.

Proof. By elementary calculations in propositional logic. □

We also need some *scope extrusion* properties.

Lemma 6.2.7. *The following are derivable:*

1. $a\#P \vdash_{\mathsf{FOL}} \forall[a](P \supset Q) = P \supset \forall[a]Q$.
2. $a\#P \vdash_{\mathsf{FOL}} \forall[a](\neg P) = \neg P$.
3. $a\#P \vdash_{\mathsf{FOL}} \forall[a]P = P$.
4. $a\#P \vdash_{\mathsf{FOL}} \forall[a](P \lor Q) = P \lor \forall[a]Q$.
5. $a\#P \vdash_{\mathsf{FOL}} \forall[a](P \land Q) = P \land \forall[a]Q$.

Proof. The first part is an an instance or axiom (**Qextr**), by part 2 of Lemma 6.2.6. The second part follows by the first part and Lemma 6.2.5, since $\neg P \equiv P \supset \bot$. The third and fourth part are corollaries of the first two parts, since $\vdash_{\mathsf{FOL}} P = \neg\neg P$ and $P \lor Q \equiv \neg P \supset Q$. The last part is a corollary of axiom (**Qdist**) and part 3 of this lemma. □

We are now in a position to derive the following 'sequent-like' properties of FOL:

Lemma 6.2.8. *For all formulae $\phi, \phi', \psi, \psi', \theta, \varepsilon$, atoms a, terms $t, t' : \mathbb{T}$, and unknowns $X_1, \ldots, X_n$:*

1. $\Delta \vdash_{\mathsf{FOL}} \phi \land \theta \supset \varepsilon \lor \phi \ = \top$
2. $\Delta \vdash_{\mathsf{FOL}} \bot \land \theta \supset \varepsilon \ = \top$
3. *if* $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \lor \phi \ = \top$ *and* $\Delta \vdash_{\mathsf{FOL}} \psi \land \theta \supset \varepsilon \ = \top$
   then $\Delta \vdash_{\mathsf{FOL}} (\phi \supset \psi) \land \theta \supset \varepsilon \ = \top$
4. *if* $\Delta \vdash_{\mathsf{FOL}} \phi \land \theta \supset \varepsilon \lor \psi \ = \top$
   then $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \lor (\phi \supset \psi) \ = \top$
5. *if* $\Delta \vdash_{\mathsf{FOL}} \phi[a \mapsto t] \land \theta \supset \varepsilon \ = \top$
   then $\Delta \vdash_{\mathsf{FOL}} \forall[a]\phi \land \theta \supset \varepsilon \ = \top$

6. *if* $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \psi \ = \top$ *and* $\Delta \vdash a\#\theta, \varepsilon$
then $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \forall[a]\psi \ = \top$
7. *if* $\Delta \vdash_{\mathsf{FOL}} \phi[a \mapsto t'] \wedge \theta \supset \varepsilon \ = \top$
then $\Delta \vdash_{\mathsf{FOL}} (t' \approx t) \wedge \phi[a \mapsto t] \wedge \theta \supset \varepsilon \ = \top$
8. $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee (t \approx t) \ = \top$
9. *if* $\Delta \vdash_{\mathsf{FOL}} \phi' \wedge \theta \supset \varepsilon \ = \top$ *and* $\Delta \vdash_{\mathsf{SUB}} \phi' = \phi$
then $\Delta \vdash_{\mathsf{FOL}} \phi \wedge \theta \supset \varepsilon \ = \top$
10. *if* $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \psi' \ = \top$ *and* $\Delta \vdash_{\mathsf{SUB}} \psi' = \psi$
then $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \psi \ = \top$
11. *if* $\Delta, a\#X_1, \dots, a\#X_n \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \ = \top$ *and* $a \notin \theta, \varepsilon, \Delta$
then $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \ = \top$
12. *if* $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \phi \ = \top,\ \Delta \vdash_{\mathsf{FOL}} \phi' \wedge \theta \supset \varepsilon \ = \top$
and $\Delta \vdash_{\mathsf{SUB}} \phi = \phi'$ *then* $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \ = \top$

Proof. The first four parts follow by elementary calculations in propositional logic.

For part 5, suppose that $\Delta \vdash_{\mathsf{FOL}} \phi[a \mapsto t] \wedge \theta \supset \varepsilon \ = \top$. By axiom (**Qinst**) we know $\Delta \vdash_{\mathsf{FOL}} \forall[a]\phi \supset \phi[a \mapsto t] = \top$. By Lemma 6.2.6 we obtain

$$\Delta \vdash_{\mathsf{FOL}} (\forall[a]\phi \supset \phi[a \mapsto t]) \wedge (\phi[a \mapsto t] \wedge \theta \supset \varepsilon) \ = \top.$$

Using further elementary calculations we conclude $\Delta \vdash_{\mathsf{FOL}} \forall[a]\phi \wedge \theta \supset \varepsilon \ = \top$ as required.

For part 6, suppose that $\Delta \vdash a\#\theta, \varepsilon$ and $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \psi \ = \top$. Using (**cong**[]) and (**congf**) we obtain $\Delta \vdash_{\mathsf{FOL}} \forall[a](\theta \supset \varepsilon \vee \psi) \ = \forall[a]\top$. We use Lemma 6.2.7 and (**tran**) to conclude $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \forall[a]\psi \ = \top$.

Parts 7 and 8 use axioms (**Esubst**) and (**Erefl**), respectively. Parts 9 and 10 follow by (**tran**) and (**congf**), since $\Delta \vdash_{\mathsf{SUB}} \phi' = \phi$ implies $\Delta \vdash_{\mathsf{FOL}} \phi' = \phi$. Part 11 is immediate using (**fr**).

Part 12: Since $\Delta \vdash_{\mathsf{SUB}} \phi = \phi'$ implies $\Delta \vdash_{\mathsf{FOL}} \phi = \phi'$, we may suppose

$$\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon \vee \phi = \top \qquad \text{and} \qquad \Delta \vdash_{\mathsf{FOL}} \phi \wedge \theta \supset \varepsilon = \top.$$

By Lemma 6.2.6 we obtain $\Delta \vdash_{\mathsf{FOL}} (\theta \supset \varepsilon \vee \phi) \wedge (\phi \wedge \theta \supset \varepsilon) = \top$. By elementary calculations in propositional logic $\Delta \vdash_{\mathsf{FOL}} \phi \wedge \theta \supset \varepsilon = \theta \supset \varepsilon \vee \neg\phi$, so we conclude $\Delta \vdash_{\mathsf{FOL}} (\theta \supset \varepsilon \vee \phi) \wedge (\theta \supset \varepsilon \vee \neg\phi) = \top$. By further calculations we reduce this to $\Delta \vdash_{\mathsf{FOL}} \theta \supset \varepsilon = \top$. □

Definition 6.2.9. For any one-and-a-halfth-order logic context $\Phi = \{\phi_1, \dots, \phi_n\}$, define its **conjunctive form** $\Phi^\wedge$ and **disjunctive form** $\Phi^\vee$ as follows:

- $\Phi^\wedge \equiv \top$ when $n = 0$, and $\Phi^\wedge \equiv \phi_1 \wedge \cdots \wedge \phi_n$ when $n > 0$.
- $\Phi^\vee \equiv \bot$ when $n = 0$, and $\Phi^\vee \equiv \phi_1 \vee \cdots \vee \phi_n$ when $n > 0$.

The order of the ϕ_i is irrelevant; we promise never to do anything such that it matters.

Theorem 6.2.10. *If* $\Phi \vdash_{\Delta} \Psi$ *is derivable in one-and-a-halfth-order logic then* $\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} = \top$.

Proof. By induction on the structure of the derivation of $\Phi \vdash_{\Delta} \Psi$. For every rule (**R**), the derivation has the following format:

$$\frac{\Pi_1 \quad \cdots \quad \Pi_k}{\Phi \vdash_{\Delta} \Psi}\,(\mathbf{R}) \quad (cond)$$

Here $k \in \{0,1,2\}$, Π_i are derivations of $\Phi_i \vdash_{\Delta_i} \Psi_i$, $1 \leq i \leq k$, and *cond* is a possibly trivial side-condition (the non-trivial cases are (**Fr**), (**Cut**), (**StructL**), (**StructR**), and ($\forall$**R**)).

So $\Phi_i \vdash_{\Delta_i} \Psi_i$ are derivable. Then $\Delta_i \vdash_{\mathsf{FOL}} {\Phi_i}^{\wedge} \supset {\Psi_i}^{\vee} = \top$ holds by the inductive hypothesis. We use this together with *cond* to prove $\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} = \top$. For each inference rule (**R**), this is an instance of a part of Lemma 6.2.8.

For example, if (**R**) is (**Cut**) then $\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} = \top$ should follow from

$$\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} \vee \phi = \top, \;\; \Delta \vdash_{\mathsf{FOL}} \phi' \wedge \Phi^{\wedge} \supset \Psi^{\vee} = \top, \;\text{ and }\; \Delta \vdash_{\mathsf{SUB}} \phi = \phi'.$$

This is an instance of part 12 of Lemma 6.2.8, using $\theta \equiv \Phi^{\wedge}$ and $\varepsilon \equiv \Psi^{\vee}$.

And if (**R**) is ($\forall$**R**) then $\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} \vee \forall[a]\psi = \top$ should follow from

$$\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} \vee \psi = \top \;\text{ and }\; \Delta \vdash a\#\Phi^{\wedge}, \Psi^{\vee}.$$

This is an instance of part 6, again using $\theta \equiv \Phi^{\wedge}$ and $\varepsilon \equiv \Psi^{\vee}$. □

6.3 FOL Derivability Implies Sequent Derivability

We now show that the sequent calculus of one-and-a-halfth-order logic (Figure 5.2) can mimic the axioms of nominal algebra theory FOL (Figures 5.1 and 6.1). In some proofs in this section, we will use the admissible sequent rules from Corollary 5.4.13 (Figure 5.5).

Lemma 6.3.1. *For all formulae* $\phi, \psi, \rho, \theta, \varepsilon$, *terms* $t, u : \mathbb{T}$, *atoms* a *and freshness contexts* Δ, *the following are derivable in one-and-a-halfth-order logic:*

1. $\vdash_{\Delta} (\top \supset \phi) \Leftrightarrow \phi$
2. $\vdash_{\Delta} ((((\phi \supset \psi) \supset (\neg\rho \supset \neg\theta)) \supset \rho) \supset \varepsilon) \supset ((\varepsilon \supset \phi) \supset (\theta \supset \phi))$
3. $\vdash_{\Delta} \forall[a]\phi \supset \phi[a \mapsto t]$
4. $\vdash_{\Delta} \forall[a](\phi \wedge \psi) \Leftrightarrow \forall[a]\phi \wedge \forall[a]\psi$
5. *if* $\Delta \vdash a\#\phi$ *then* $\vdash_{\Delta} \forall[a](\phi \supset \psi) \Leftrightarrow \phi \supset \forall[a]\psi$
6. $\vdash_{\Delta} u \approx t \wedge \phi[a \mapsto t] \supset \phi[a \mapsto u]$
7. $\vdash_{\Delta} t \approx t$

Proof. We give details of parts 5 and 6. The derivation of part 6 is completely syntax-directed:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\phi[a\mapsto u]\vdash_\triangle \phi[a\mapsto u]}\,(\mathbf{Ax})}{u\approx t, \phi[a\mapsto t]\vdash_\triangle \phi[a\mapsto u]}\,(\approx\mathbf{L})}{u\approx t\wedge \phi[a\mapsto t]\vdash_\triangle \phi[a\mapsto u]}\,(\wedge\mathbf{L})}{\vdash_\triangle u\approx t\wedge\phi[a\mapsto t]\supset\phi[a\mapsto u]}\,(\supset\mathbf{R})$$

For part 5, we assume $\Delta\vdash a\#\phi$. By ($\Leftrightarrow$**R**), it suffices to derive

(a) $\phi, \forall[a](\phi\supset\psi)\vdash_\triangle \forall[a]\psi$, and
(b) $\phi\supset\forall[a]\psi\vdash_\triangle\forall[a](\phi\supset\psi)$.

We conclude (a) by Theorem 5.4.7 and the last derivation of Figure 5.3. We derive (b) as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\phi\vdash_\triangle \psi,\phi}\,(\mathbf{Ax}) \qquad \dfrac{\dfrac{\dfrac{}{\phi,\psi\vdash_\triangle\psi}\,(\mathbf{Ax})}{\phi,\psi[a\mapsto a]\vdash_\triangle\psi}\,(\mathbf{StructL})\quad(\Delta\vdash_{\mathsf{SUB}}\psi=\psi[a\mapsto a])}{\phi,\forall[a]\psi\vdash_\triangle\psi}\,(\forall\mathbf{L})}{\phi,\phi\supset\forall[a]\psi\vdash_\triangle\psi}\,(\supset\mathbf{L})}{\phi\supset\forall[a]\psi\vdash_\triangle\phi\supset\psi}\,(\supset\mathbf{R})}{\phi\supset\forall[a]\psi\vdash_\triangle\forall[a](\phi\supset\psi)}\,(\forall\mathbf{R})\quad(\Delta\vdash a\#\phi\supset\forall[a]\psi)$$

□

Lemma 6.3.2. *In the sequent calculus of one-and-a-halfth-order logic:*

- *bi-implication* $\Leftrightarrow$ *is an equivalence relation (a reflexive symmetric transitive relation), i.e. the following rules are admissible:*

$$\dfrac{}{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\phi}\qquad\dfrac{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\psi}{\Phi\vdash_\triangle\Psi,\psi\Leftrightarrow\phi}\qquad\dfrac{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\psi\quad\Phi\vdash_\triangle\Psi,\psi\Leftrightarrow\xi}{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\xi}$$

- *bi-implication is a congruence:*

$$\dfrac{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\psi}{\Phi\vdash_\triangle\Psi,\xi[\phi/P]\Leftrightarrow\xi[\psi/P]}$$

- $\top$ *is the left and right identity of bi-implication:*

$$\dfrac{\Phi\vdash_\triangle\Psi,\phi}{\Phi\vdash_\triangle\Psi,\top\Leftrightarrow\phi}\qquad\dfrac{\Phi\vdash_\triangle\Psi,\top\Leftrightarrow\phi}{\Phi\vdash_\triangle\Psi,\phi}\qquad\dfrac{\Phi\vdash_\triangle\Psi,\phi}{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\top}\qquad\dfrac{\Phi\vdash_\triangle\Psi,\phi\Leftrightarrow\top}{\Phi\vdash_\triangle\Psi,\phi}$$

Proof. By straightforward calculations using the derivation rules in Figure 5.2 (and the admissible rules in Figure 5.5). In the congruence case we use induction on the structure of ξ. $\square$

We use Lemmas 6.3.1 and 6.3.2 to show that if the equality $t = u$ is derivable in FOL then bi-implications $\phi[t/X] \Leftrightarrow \phi[u/X]$ are derivable in the sequent calculus for any ϕ and X.

Theorem 6.3.3. *For all sorts τ, terms $t, u : \tau$, unknowns $X : \tau$, formulae ϕ, and freshness contexts Δ:*

$$\textit{if} \quad \Delta \vdash_{\mathsf{FOL}} t = u \quad \textit{then} \quad \vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[u/X]$$

Proof. By induction on the structure of FOL derivations of $t = u$ from Δ.
(**refl**): $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[t/X]$ follows by reflexivity of $\Leftrightarrow$.
(**symm**): $\vdash_{\Delta} \phi[u/X] \Leftrightarrow \phi[t/X]$ follows from $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[u/X]$ by symmetry of $\Leftrightarrow$. By inductive hypothesis this follows from the assumption.
(**tran**): Similarly, $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[v/X]$ follows from $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[u/X]$ and $\vdash_{\Delta} \phi[u/X] \Leftrightarrow \phi[v/X]$ by transitivity of $\Leftrightarrow$. By the inductive hypothesis these follow from the assumptions.
(**cong[]**): By the inductive hypothesis $\vdash_{\Delta} \psi[t/Y] \Leftrightarrow \psi[u/Y]$ for any Y and ψ. We must show $\vdash_{\Delta} \phi[[a]t/X] \Leftrightarrow \phi[[a]u/X]$, which is syntactically equivalent to

$$\vdash_{\Delta} \phi[[a]Z/X][t/Z] \Leftrightarrow \phi[[a]Z/X][u/Z],$$

where Z is an unknown (of appropriate sort) that does not occur in ϕ. This follows directly from the inductive hypothesis, taking $\psi \equiv \phi[[a]Z/X]$ and $Y \equiv Z$.
(**congf**): Analogous to the previous case.
(**perm**): We show $\vdash_{\Delta} \phi[(a\ b) \cdot t/X] \Leftrightarrow \phi[t/X]$ as follows:

$$\dfrac{\dfrac{}{\vdash_{\Delta} \phi[(a\ b) \cdot t/X] \Leftrightarrow \phi[(a\ b) \cdot t/X]}\,(\mathbf{Ax})}{\vdash_{\Delta} \phi[(a\ b) \cdot t/X] \Leftrightarrow \phi[t/X]}\,(\mathbf{StructR})$$

where $\Delta \vdash_{\mathsf{SUB}} \phi[(a\ b) \cdot t/X] \Leftrightarrow \phi[(a\ b) \cdot t/X] = \phi[(a\ b) \cdot t/X] \Leftrightarrow \phi[t/X]$ is the side-condition of (**StructR**). By (**congf**) and congruence Lemma 2.4.9, this follows from the assumption $\Delta \vdash_{\mathsf{SUB}} (a\ b) \cdot t = t$.
(**fr**): So $\Delta \vdash_{\mathsf{FOL}} t = u$ follows from $\Delta, a\#X_1, \ldots, a\#X_n \vdash_{\mathsf{FOL}} t = u$ where $a \notin t, u, \Delta$. We must show $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[u/X]$. We cannot apply (**Fr**) directly, since ϕ might mention a. Using ZFA equivariance (Theorem A.2.5) we rename a to a fresh a' (i.e. $a' \notin t, u, \phi, \Delta$) while preserving the inductive hypothesis, to obtain

$$\vdash_{\Delta, a'\#X_1, \ldots, a'\#X_n} \phi[t/X] \Leftrightarrow \phi[u/X].$$

We conclude $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[u/X]$ by (**Fr**), since $a' \notin \phi[t/X] \Leftrightarrow \phi[u/X], \Delta$.
(**ax$_{\mathbf{A}}$**): We work by cases:

- If A is an axiom of SUB (from Figure 5.1) then we have derived $\Delta \vdash_{\mathsf{SUB}} t = u$. By congruence Lemma 2.4.9 we know $\Delta \vdash_{\mathsf{SUB}} \phi[t/X] = \phi[u/X]$. We must show $\vdash_{\Delta} \phi[t/X] \Leftrightarrow \phi[u/X]$. By $(\Leftrightarrow\mathbf{R})$, this follows from $\phi[t/X] \vdash_{\Delta} \phi[u/X]$ and $\phi[u/X] \vdash_{\Delta} \phi[t/X]$. The former can be derived:

$$\dfrac{\dfrac{}{\phi[t/X] \vdash_{\Delta} \phi[t/X]}\,(\mathbf{Ax})}{\phi[t/X] \vdash_{\Delta} \phi[u/X]}\,(\mathbf{StructR}) \quad (\Delta \vdash_{\mathsf{SUB}} \phi[t/X] = \phi[u/X])$$

  The latter derivation is analogous, using $(\mathbf{StructL})$.
- If A is an axiom from Figure 6.1 then the derivation is of the form

$$\dfrac{\Pi}{\phi^{\pi}\sigma = \psi^{\pi}\sigma}\,(\mathbf{ax}_{\nabla\vdash\phi=\psi})$$

  where Π is a derivation of $\nabla^{\pi}\sigma$. We must show $\vdash_{\Delta} \xi[\phi^{\pi}\sigma/P] \Leftrightarrow \xi[\psi^{\pi}\sigma/P]$. By congruence of $\Leftrightarrow$ (Lemma 6.3.2) this follows from $\vdash_{\Delta} \phi^{\pi}\sigma \Leftrightarrow \psi^{\pi}\sigma$. In case $\psi \equiv \top$, this follows from $\vdash_{\Delta} \phi^{\pi}\sigma$ by right identity of $\Leftrightarrow$. For each axiom the remaining proof obligation is an instance of a part of Lemma 6.3.1, using the assumption $\Delta \vdash \nabla^{\pi}\sigma$.

□

6.4 Equivalence of the Sequent Calculus and FOL

In Sections 6.2 and 6.3 we have established the essential properties to show that derivability in the sequent calculus and derivability in theory FOL is equivalent.

Lemma 6.4.1. *If* $\vdash_{\Delta} \Phi^{\wedge} \supset \Psi^{\vee}$ *then* $\Phi \vdash_{\Delta} \Psi$.

Proof. By $(\mathbf{Cut})$ $\Phi \vdash_{\Delta} \Psi$ follows from $\Phi \vdash_{\Delta} \Psi,\ \Phi^{\wedge} \supset \Psi^{\vee}$ and $\Phi^{\wedge} \supset \Psi^{\vee},\ \Phi \vdash_{\Delta} \Psi$. Then $\Phi \vdash_{\Delta} \Psi,\ \Phi^{\wedge} \supset \Psi^{\vee}$ follows from the assumption $\vdash_{\Delta} \Phi^{\wedge} \supset \Psi^{\vee}$ using formula weakening (Lemma 5.4.12). The remaining proof obligation $\Phi^{\wedge} \supset \Psi^{\vee},\ \Phi \vdash_{\Delta} \Psi$ follows from $\Phi \vdash_{\Delta} \Psi,\ \Phi^{\wedge}$ and $\Psi^{\vee},\ \Phi \vdash_{\Delta} \Psi$ by $(\supset\mathbf{L})$. The result follows by an induction on the size of Φ and Ψ. □

Theorem 6.4.2 (Equivalence of $\vdash_{\Delta}$ and FOL)**.** *For any* Δ, Φ, Ψ*:*

$$\Phi \vdash_{\Delta} \Psi \quad \textit{if and only if} \quad \Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} = \top.$$

Proof. The left-to-right part is Theorem 6.2.10.

For the right-to-left part, assume $\Delta \vdash_{\mathsf{FOL}} \Phi^{\wedge} \supset \Psi^{\vee} = \top$. Then by Theorem 6.3.3 (taking ϕ to be X), $\vdash_{\Delta} \Phi^{\wedge} \supset \Psi^{\vee} \Leftrightarrow \top$ is derivable. By right identity of $\Leftrightarrow$, also $\vdash_{\Delta} \Phi^{\wedge} \supset \Psi^{\vee}$. By Lemma 6.4.1 we obtain $\Phi \vdash_{\Delta} \Psi$, as required. □

Since Theorem 6.4.2 is stated using formula contexts Φ and Ψ, the reader might get the impression that we have only shown that we can represent derivability of arbitrary sequents by derivable equations from theory FOL, but not the other way round. However, the following corollary shows that we can also represent derivable equations on arbitrary formulae in FOL by derivable sequents.

Corollary 6.4.3. *For any Δ, ϕ, ψ:*

$$\Delta \vdash_{\mathsf{FOL}} \phi = \psi \quad \textit{if and only if} \quad \phi \vdash_{\Delta} \psi \quad \textit{and} \quad \psi \vdash_{\Delta} \phi.$$

Proof. By Theorem 6.4.2 $\phi \vdash_{\Delta} \psi$ and $\psi \vdash_{\Delta} \phi$ are equivalent to $\Delta \vdash_{\mathsf{FOL}} \phi \supset \psi = \top$ and $\Delta \vdash_{\mathsf{FOL}} \psi \supset \phi = \top$. This is equivalent to $\Delta \vdash_{\mathsf{FOL}} \phi \Leftrightarrow \psi = \top$, by part 1 of Lemma 6.2.6. Finally, by part 2 of that lemma, this is equivalent to $\Delta \vdash_{\mathsf{FOL}} \phi = \psi$. □

Now that we have established a formal connection between the sequent calculus and FOL in Theorem 6.4.2, a number of properties on the sequent calculus easily carry over to FOL.

Corollary 6.4.4 (Consistency of FOL)**.** FOL *is consistent. That is:*

$$\Delta \nvdash_{\mathsf{FOL}} \top = \bot \quad \textit{for any} \quad \Delta.$$

Proof. By contradiction. Suppose $\Delta \vdash_{\mathsf{FOL}} \top = \bot$. Using elementary calculations in propositional logic, also $\Delta \vdash_{\mathsf{FOL}} \top \supset \bot = \top$. Note that $\top \equiv \emptyset^{\wedge}$ and $\bot \equiv \emptyset^{\vee}$, so by Theorem 6.4.2 $\vdash_{\Delta}$ is derivable, which contradicts consistency of one-and-a-halfth-order logic (Corollary 5.4.18). □

Corollary 6.4.5 (Equivalence of FOL and $\vdash$)**.** *For ground ϕ, ψ, $\vdash_{\mathsf{FOL}} \phi = \psi$ in nominal algebra if and only if $\phi \vdash \psi$ and $\psi \vdash \phi$ are derivable in Gentzen's sequent calculus.*

Proof. By Corollaries 6.4.3 and 5.5.5. □

6.5 Conclusions

One-and-a-halfth-order logic can be axiomatised in nominal algebra and the treatment of quantification is smooth. Since the sequent calculus of one-and-a-halfth-order logic (Figure 5.2) and the axioms of theory FOL (Figures 5.1 and 6.1) use the same notions of terms and freshness, we obtain a proof theory that is both syntax-directed *and* algebraic.

The axiomatisation presented in this chapter is one (more) element in a long line of investigations into algebraic logic [ANS01]; for example cylindric [HMT85], polyadic [Hal56], and quantifier [Pin73] algebra. There too, meta-variables are made explicit. However, as mentioned before (the Conclusions of Chapter 2), we claim that these approaches do not provide a *natural* representation of object-variables and binding.

We see no difficulties in principle with axiomatising substructural logics such as linear logic [Gir87], bunched implications [OP99], relevance logics [DR02], and so on; if the logic is susceptible to a (nominal) algebraic treatment then it can be axiomatised in nominal algebra. This is not necessarily so in a higher-order setting because structural properties of the framework's connectives may 'infect' those of the logic being axiomatised.

Chapter 7

Conclusions

In the previous chapters we have presented two logics with explicit meta-variables: one for equality (Chapter 2) and another one for first-order logic with equality (Chapter 5). We have argued that these logics accurately and effectively represent the meta-level of equality and first-order logic, using unknowns and freshness conditions. Next to studying the proof theory of these logics, we have given a connection between the two (Chapter 6), and we investigated the logic of equality in the areas of denotational semantics (Chapter 3) and capture-avoiding substitution (Chapter 4).

For detailed concluding remarks, including related and future work, we refer to reader to the respective chapters mentioned above. Here we give a some of the concrete technical challenges we are facing:

- *Semantics.* We have paid relatively little attention to models. Although we have specified a sound and complete semantics for nominal algebra theories in nominal sets, we have not yet proved Birkhoff's theorem (also known as the HSP theorem). Since this is one of the basic results of universal algebra, which gives a valuable insight into the structure of models of algebraic theories, this is important future work.

- *Capture-avoiding substitution.* We have proved that nominal algebra theory SUB from Chapter 4 axiomatises capture-avoiding substitution on syntax. However the λ-calculus provides an alternative model of substitution, via β-conversion. It remains to relate these two models, e.g. by providing a formal translation between terms in SUB and a suitable λ-calculus.

- *One-and-a-halfth-order logic.* We have set up one-and-a-halfth-order logic such that freshness conditions and (implicit) quantification over unknowns both occur at top-level. We would like to allow freshness conditions and quantification anywhere in a formula. The logic we obtain could be called two-and-a-halfth-order logic: it is related to second-order logic because of the quantification over unknowns but also more general since formulas may be enriched with freshness conditions.

- *Implementation.* How suited are nominal algebra and one-and-a-halfth-order logic to complement existing frameworks such as Isabelle [Pau89], Coq [HKPM] or mCRL2 [GMR$^+$07]? How do they need to be extended?

- *Methodology.* Can we employ the method we have advocated in this thesis to formalise reasoning on other systems with binding such as process calculi and substructural logics?

From a wider perspective, this thesis pursues an overall vision that *names* are worth talking about as mathematical entities, both in logic and in denotation. The use of nominal techniques may be a useful complement to existing methods predicated on function-spaces.

We have seen this in the fact that we have been able to axiomatise *and then study* a representative sample of some of the most influential tools of computer science, including equational logic, first-order logic and substitution. Furthermore, and perhaps most promisingly, we have seen how the flavour of current mathematical systems is maintained in the nominal setting. We are particularly pleased how successfully nominal algebra and one-and-a-halfth-order logic capture the informal practice of mathematics, in a completely rigorous setting.

Appendix A

ZFA Equivariance

A.1 Introduction

We use atoms in this thesis — we introduce them when we write 'Fix a countably infinite collection of **atoms** $a, b, c, \ldots$' in Definition 2.2.1.

We can represent atoms as numbers $0, 1, 2, 3, \ldots$, or as sets $\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \ldots$. In principle we might 'accidentally' use some property of atoms specific to their implementation, such as $a \leq b$ or $a \in b$. However, we know we do not do this, because we consider atoms to be… atomic.

By explicitly bearing this in mind, we can rename atoms. This is *equivariance*, proved below, which we use freely in this thesis to give structural inductive proofs (renaming atoms in inductive hypotheses where convenient) while remaining fully formal.

If we wish to be fully formal but ignore equivariance we must work by induction on measures such as term length or derivation depth. These are longer, harder to read, and are rarely given in full detail outside of a theorem-prover.

To give a precise statement and formal proof of equivariance and how it can be used to rename atoms in proofs by structural induction on the syntax of terms and derivation-trees we use a foundational theory: Zermelo-Fraenkel set theory with atoms (ZFA). ZFA has *equivariance* [Bru96] and because it is known that set theory can be used to formalise mathematics, equivariance is a meta-mathematical property which we can use in the proofs in this thesis.

We do *not* mean that equivariance refers to, using terminology from Chapter 5, the terms of sort $\mathbb{F}$, nor the formulae of first-order logic, nor sequents of one-and-a-halfth-order logic. We mean that it refers to the assertions written *in English in this thesis* — *about* one-and-a-halfth-order logic, nominal algebra, and so on.

ZFA features a set $\mathbb{A}$ of *atoms* $a, b, c, \ldots$ (originally called Urelemente [Bar75]). The original motivation of atoms was to address the question 'what set is equal to Plato?'. Obviously Plato is not any of the 'normal' sets of set theory, such as the empty set $\emptyset$ or the set containing just the empty set $\{\emptyset\}$, and so on. The answer was to accommodate 'the real world' by introducing it *en masse* into the set model, as atoms.

As far as the set theory is concerned atoms are atomic objects with no internal (set-)structure, so it is quite natural to use these to model variable symbols. This idea appears already in [Bar75].

A.2 ZFA Set Theory and Equivariance

Definition A.2.1. For the language of ZFA set theory, in addition to the basic language of first-order logic with equality, we assume:

- A binary predicate symbol $\in$ called *set inclusion.*
- A constant term-former $\mathbb{A}$ called *the set of atoms.*

We use standard sugar of classical logic (similar to the sugar mentioned in Subsection 5.2).

Definition A.2.2. ZFA set theory has the axioms in Figure A.1.

In Figure A.1, ϕ ranges over all predicates, $\phi[y/x]$ denotes the predicate obtained by capture-avoiding substitution of x by y, and $F(y)$ represents a function on the sets universe (strictly speaking, this is itself sugar, which is briefly described in Corollary A.2.6). We also use the following sugar:

$x = \{z \mid z \in x\}$	is sugar for	$\forall y.(\forall z.(z \in x \Leftrightarrow z \in y) \supset x = y)$
$y = \{z \in x \mid \phi\}$	is sugar for	$\forall z.(z \in y \Leftrightarrow (z \in x \wedge \phi))$
$z = \{F(y) \mid y \in x\}$	is sugar for	$\forall u.(u \in z \Leftrightarrow \exists y.(F(y) = u \wedge y \in x))$
$z = \{x, y\}$	is sugar for	$\forall u.(u \in z \Leftrightarrow (u = x \vee u = y))$
$z = \{y \mid \exists y'.(y \in y' \wedge y' \in x)\}$	is sugar for	$\forall y.(y \in z \Leftrightarrow \exists y'.(y \in y' \wedge y' \in x))$
$z = \{y \mid y \subseteq x\}$	is sugar for	$\forall y.(y \in z \Leftrightarrow \forall y'.(y' \in y \supset y' \in x))$
$\emptyset \in x$	is sugar for	$\exists z.(z \in x \wedge \forall z'.z' \notin z)$
$y \cup \{z\} \in x$	is sugar for	$\exists u.(u \in x \wedge \forall u'.(u' \in u \Leftrightarrow u \in y \vee u = z))$

The syntactic sugar used in set theory is very rich; further details can be found elsewhere [Joh87].

Definition A.2.3. We define a **permutation action** on ZFA sets by:

$$\pi \cdot a = \pi(a) \qquad \pi \cdot X = \{\pi \cdot x \mid x \in X\} \quad (X \notin \mathbb{A})$$

This definition is by *ϵ-induction*, a standard method in set theory [Joh87]; if it is true that a property holds of $y \in \mathbb{A}$, and *if* it holds of all $x \in y$ *then* it holds of all y, then that property holds of all y. Written informally: *sets are well-founded trees with daughter-of given by set inclusion* $\in$.

Lemma A.2.4. *$\pi \cdot (\pi' \cdot z) = (\pi \circ \pi') \cdot z$.*

Proof. The proof is by ϵ-induction.

(**Sets**)	$\forall x.((\exists y.y \in x) \supset x \notin \mathbb{A})$	
(**Extensionality**)	$\forall x.(x \notin \mathbb{A} \supset x = \{z \mid z \in x\})$	
(**Comprehension**)	$\forall x.\exists y.(y \notin \mathbb{A} \wedge y = \{z \in x \mid \phi\})$	(y not free in ϕ)
(**$\in$-Induction**)	$\forall x.(\forall y.(y \in x \supset \phi[y/x]) \supset \phi) \supset \forall x.\phi$	
(**Replacement**)	$\forall x.\exists z.(z \notin \mathbb{A} \wedge z = \{F(y) \mid y \in x\})$	
(**Pairset**)	$\forall x.\forall y.\exists z.(z = \{x, y\})$	
(**Union**)	$\forall x.\exists z.(z \notin \mathbb{A} \wedge z = \{y \mid \exists y'.(y \in y' \wedge y' \in x)\})$	
(**Powerset**)	$\forall x.\exists z.(z = \{y \mid y \subseteq x\})$	
(**Infinity**)	$\exists x.(\emptyset \in x \wedge \forall y.(y \in x \supset y \cup \{y\} \in x))$	

Figure A.1 Axioms of ZFA set theory

- By definition if $a \in \mathbb{A}$ then $\pi \cdot (\pi' \cdot a) = \pi(\pi'(a)) = (\pi \circ \pi') \cdot a$.
- Suppose $Z \notin \mathbb{A}$. Then by definition of the permutation action and by the inductive hypothesis

$$\begin{aligned}\pi \cdot (\pi' \cdot Z) &= \{\pi \cdot (\pi' \cdot u) \mid u \in Z\} = \{(\pi \circ \pi') \cdot u \mid u \in Z\} \\ &= (\pi \circ \pi') \cdot \{u \mid u \in Z\} = (\pi \circ \pi') \cdot Z.\end{aligned}$$

□

Recall that ϕ ranges over predicates of ZFA. Write $\phi(x_1, \dots, x_n)$ to range over predicates which mention at most $x_1, \dots, x_n$ as free variable symbols.

Theorem A.2.5 (ZFA equivariance). *If $\phi(x_1, \dots, x_n)$ is a predicate of ZFA set theory then*

$$\phi(x_1, \dots, x_n) \Leftrightarrow \phi(\pi \cdot x_1, \dots, \pi \cdot x_n)$$

is always provable.

As a corollary, $\phi(x_1, \dots, x_n)$ and $\phi(\pi \cdot x_1, \dots, \pi \cdot x_n)$ are interchangeable in proof and in validity on models.

Proof. We work by induction on the syntax of ϕ.

- By definition, $x \in y$ implies $\pi \cdot x \in \pi \cdot y$ follows directly from the fact that $\pi \cdot y = \{\pi \cdot y' \mid y' \in y\}$. The reverse implication is easy using π^{-1}.
- Similarly, $x = y$ if and only if $\pi \cdot x = \pi \cdot y$.
- The case of $\bot$ is trivial, and the cases of $\phi_1 \supset \phi_2$ and $\forall z.\phi'$ follow using the inductive hypothesis.
- $\pi \cdot \mathbb{A} = \mathbb{A}$ is provable, so $x \in \mathbb{A}$ if and only if $\pi \cdot x \in \mathbb{A}$, and $\mathbb{A} \in y$ if and only if $\mathbb{A} \in \pi \cdot y$, and similarly $x = \mathbb{A}$ if and only if $\pi \cdot x = \mathbb{A}$ and $\mathbb{A} = y$ if and only if $\mathbb{A} = \pi \cdot y$.

The result follows. □

Corollary A.2.6. *If $F(x_1, \ldots, x_n)$ is function (not a function-set) from the set universe to itself then*

$$\pi \cdot (F(x_1, \ldots, x_n)) = F(\pi \cdot x_1, \ldots, \pi \cdot x_n)$$

is always provable.

Proof. In set theory, we specify F using a predicate $\phi(x_1, \ldots, x_n, z)$ such that

$$\forall x_1, \ldots, x_n.(\exists z.\phi(x_1, \ldots, x_n, z) \;\; \wedge \forall z, z'.(\phi(x_1, \ldots, x_n, z) \wedge \phi(x_1, \ldots, x_n, z') \supset z = z')).$$

$\square$

Bibliography

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *CCS '97: Proc. of the 4th ACM conf. on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[ANS01] H. Andréka, I. Németi, and I. Sain. Algebraic logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 2, pages 133–249. Kluwer, 2001.

[Bar75] Jon Barwise. *Admissible Sets and Structures: an approach to definability theory.* Perspectives in mathematical logic. Springer, 1975.

[Bar84] H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics (revised ed.).* North-Holland, 1984.

[Bar00] H.P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 2*, pages 117–309. OUP, 2000.

[Bee04] Michael Beeson. Lambda logic. In *IJCAR 2004: The 2nd International Joint Conference on Automated Reasoning*, volume 3097 of *LNCS*, pages 460–474. Springer, 2004.

[Blo97] Roel Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Eindhoven University of Technology, Eindhoven, 1997.

[BR95] Roel Bloo and Kristoffer Høgsbro Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*, 1995.

[Bru96] N. Brunner. 75 years of independence proofs by Fraenkel-Mostowski permutation models. *Mathematica Japonica*, 43:177–199, 1996.

[BS81] S. Burris and H. Sankappanavar. *A Course in Universal Algebra.* Springer, 1981.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[Cou97] Bruno Courcelle. The expression of graph properties in some fragments of monadic second-order logic. *DIMACS Series in Discrete Mathematics*, 31, 1997.

[CP07] Ranald A. Clouston and Andrew M. Pitts. Nominal equational logic. *ENTCS*, 172:223–257, 2007.

[Cra04] Marcel Crabbé. On the notion of substitution. *Logic Journal of the IGPL*, 12(2):111–124, 2004.

[dB72] N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 5(34):381–392, 1972.

[dB91] N.G. de Bruijn. Checking mathematics with computer assistance. *Notices of the American Mathematical Society (AMS)*, 38(1):8–15, 1991.

[DHK02] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Binding logic: Proofs and models. In *LPAR '02: 9th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, LNCS, pages 130–144, London, UK, 2002. Springer-Verlag.

[DR02] Mike Dunn and Greg Restall. Relevance logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 6, pages 1–128. Kluwer, 2002.

[DV01] Anatoli Degtyarev and Andrei Voronkov. Equality reasoning in sequent-based calculi. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 611–706. Elsevier and MIT Press, 2001.

[Far06] William M. Farmer. The seven virtues of simple type theory. Technical Report 18, McMaster University, SQRL, 2003 (revised 2006).

[Fel82] Norman Feldman. Axiomatization of polynomial substitution algebras. *Journal of Symbolic Logic*, 47(3):481–492, 1982.

[FG07] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting. *Information and Computation*, 205:917–965, 2007.

[FPT99] Marcelo Fiore, Gordon Plotkin, and Daniele Turi. Abstract syntax and variable binding. In *14th Annual Symposium on Logic in Computer Science*, pages 193–202. IEEE Computer Society Press, 1999.

[FS06] Marcelo Fiore and Sam Staton. A congruence rule format for name-passing process calculi from mathematical structural operational semantics. In *21st Annual Symposium on Logic in Computer Science (LICS'06)*, pages 49–58. IEEE, Computer Society Press, 2006.

[Gab05] Murdoch J. Gabbay. A new calculus of contexts. In *PPDP '05: Proc. of the 7th ACM SIGPLAN international conference on Principles and Practice of Declarative Programming*, pages 94–105. ACM Press, 2005.

[Gab06] Murdoch J. Gabbay. Fresh logic. *Journal of Logic and Computation*, 2006. In press.

[Gab07a] Murdoch J. Gabbay. A general mathematics of names. *Information and Computation*, 205(7):982–1011, 2007.

[Gab07b] Murdoch J. Gabbay. Hierarchical nominal terms and their theory of rewriting. *ENTCS*, 174(5):37–52, 2007.

[Gen35] G. Gentzen. Untersuchungen über das logische schließen [Investigations into logical deduction]. *Mathematische Zeitschrift 39*, pages 176–210,405–431, 1935. Translated in [Sza69], pages 68–131.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[GJ02] Herman Geuvers and Gueorgui I. Jojgov. Open proofs and open terms: A basis for interactive logic. In *Computer Science Logic: 16th International Workshop*, pages 537–552, 2002.

[GL07] Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus. In *LFMTP'07: International Workshop on Logical Frameworks and Meta-Languages*, to appear in ENTCS, 2007.

[GM02] D.M. Gabbay and G. Malod. Naming worlds in modal and temporal logic. *Journal of Logic, Language and Information*, 11(1):29–65, 2002.

[GMR$^+$07] Jan Friso Groote, Aad Mathijssen, Michel Reniers, Yaroslav Usenko, and Muck van Weerdenburg. The formal specification language mCRL2. In *Methods for Modelling Software Systems (MMOSS)*, number 06351 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[GP02] Murdoch J. Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2002.

[GP06] A. Gareau and R. Padmanabhan. Two axioms for implication algebras. *Notre Dame Journal of Formal Logic*, 2006.

[Gro97] Jan Friso Groote. The syntax and semantics of timed μCRL. Technical Report SEN-R9709, CWI, Amsterdam, 1997.

[Hal56] P. Halmos. Algebraic logic, ii. homogeneous locally finite polyadic boolean algebras of infinite degree. *Fundamenta Mathematicae*, 43:255–325, 1956.

[HKPM] Gérard Huet, Gilles Kahn, and Christine Paulin-Mohring. The Coq proof assistant, a tutorial.
`http://coq.inria.fr/V8.1/tutorial.html`.

[HMT85] L. Henkin, J.D. Monk, and A. Tarski. *Cylindric Algebras.* North Holland, 1971 and 1985. Parts I and II.

[Joh87] P.T. Johnstone. *Notes on logic and set theory.* Cambridge University Press, 1987.

[Joj04] Gueorgui I. Jojgov. *Incomplete Proofs and Terms and Their Use in Interactive Theorem Proving.* PhD thesis, Technische Universiteit Eindhoven, 2004.

[KD02] Joost-Pieter Katoen and Pedro R. D'Argenio. General distributions in process algebra. In *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*, pages 375–429. Springer, 2002.

[KKSdV97] J.R. Kennaway, J.W. Klop, M.R. Sleep, and F.J. de Vries. Infinitary lambda calculus. *Theoretical Computer Science*, 175:93–125, 1997.

[KvOvR93] Jan Willem Klop, Vincent van Oostrom, and Femke van Raamsdonk. Combinatory reduction systems, introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[Les94] Pierre Lescanne. From lambda-sigma to lambda-upsilon a journey through calculi of explicit substitutions. In *POPL '94: Proc. 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 60–69. ACM Press, 1994.

[LS04] Stefania Lusin and Antonino Salibra. The lattice of lambda theories. *Journal of Logic and Computation*, 14(3):373–394, 2004.

[Lut02] Bas Luttik. *Choice Quantification in Process Algebra.* PhD thesis, University of Amsterdam, 2002.

[Mei92] Karl Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100(2):385–417, 1992.

[Mer53] C.A. Meredith. Single axioms for the systems (C, N), (C, O) and (A, N) of the two-valued propositional calculus. *The Journal of Computing Systems*, 3:155–164, 1953.

[Met] Metamath.org. Derivation of classical propositional logic from Meredith's axiom.
`http://us.metamath.org/mpegif/meredith.html`.

[Mic01] Marino Miculan. Developing (meta)theory of lambda-calculus in the theory of contexts. *ENTCS*, 58(1), 2001.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Extensions of Logic Programming*, 475:253–281, 1991.

[MN98] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.

[MVF$^+$02] William McCune, Robert Veroff, Branden Fitelson, Kenneth Harris, Andrew Feist, and Larry Wos. Short single axioms for boolean algebra. *Journal of Automated Reasoning*, 29(1):1–16, 2002.

[OP99] Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.

[Par01] Joachim Parrow. An introduction to the pi-calculus. In Jan Bergstra, Alban Ponse, and Scott Smolka, editors, *Handbook of Process Algebra*, pages 479–543. Elsevier Science, 2001.

[Pau89] Lawrence C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[Pau90] Lawrence C. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[PE88] Frank Pfenning and Conal Elliot. Higher-order abstract syntax. In *PLDI '88: the ACM SIGPLAN 1988 conference on Programming Language design and Implementation*, pages 199–208. ACM Press, 1988.

[Pin73] Charles Pinter. A simple algebra of first-order logic. *Notre Dame Journal of Formal Logic*, 14(3):361–366, 1973.

[Pit03] Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.

[Pit06] Andrew M. Pitts. Alpha-structural recursion and induction. *Journal of the ACM*, 53(3):459–506, 2006.

[Pra65] Dag Prawitz. *Natural Deduction: A Proof Theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.

[Sal00] Antonino Salibra. On the algebraic models of lambda calculus. *Theoretical Computer Science*, 249(1):197–240, 2000.

[Sha01] Stewart Shapiro. Systems between first-order and second-order logics. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 1, pages 131–188. Kluwer, 2001.

[Sun99] Yong Sun. An algebraic generalization of Frege structures - binding algebras. *Theoretical Computer Science*, 211:189–232, 1999.

[Sza69] M.E. Szabo, editor. *Collected Papers of Gerhard Gentzen*. North Holland, 1969.

[UPG04] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.

[vB01] Johan van Benthem. Higher-order logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 1, pages 189–244. Kluwer, 2001.

[vD02] Dirk van Dalen. Intuitionistic logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 5, pages 1–114. Kluwer, 2002.

Summary

In informal mathematical usage we often reason about languages involving binding of object-variables. We find ourselves writing assertions involving *meta-variables* and *capture-avoidance constraints* on where object-variables can and cannot occur free. Formalising such assertions is problematic because the standard logical frameworks cannot express capture-avoidance constraints directly.

In this thesis we make the case for *extending* logical frameworks with meta-variables and capture-avoidance constraints. We use nominal techniques that allow for a direct formalisation of meta-level assertions, while remaining close to informal practice. Our focus is on *derivability* and we show that our derivation rules support the following key features of meta-level reasoning:

- instantiation of meta-variables, by means of *capturing* substitution of terms for meta-variables;
- *α-renaming* of object-variables and *capture-avoiding* substitution of terms for object-variables in the presence of meta-variables;
- generation of *fresh* object-variables inside a derivation.

We apply our nominal techniques to the following two logical frameworks:

- *Equational logic.* We investigate proof-theoretical properties, give a semantics in nominal sets and compare the notion of α-renaming to existing notions of α-equivalence with meta-variables. We also provide an axiomatisation of capture-avoiding substitution, and show that it is sound and complete with respect to the usual notion of capture-avoiding substitution.
- *First-order logic with equality.* We provide a sequent calculus with meta-variables and capture-avoidance constraints, and show that it represents schemas of derivations in first-order logic. We also show how we can axiomatise this notion of derivability in the calculus for equational logic.

Samenvatting

In informeel wiskundig gebruik redeneren we vaak over talen met binding van object-variabelen. Hierbij schrijven we asserties met *meta-variabelen* en zogenaamde *capture-avoidance restricties* die uitdrukken waar object-variabelen wel en niet vrij kunnen voorkomen. Het formaliseren van zulke asserties is problematisch omdat de standaard logische raamwerken capture-avoidance restricties niet rechtstreeks kunnen uitdrukken.

In dit proefschrift pleiten we ervoor om de logische raamwerken *uit te breiden* met meta-variabelen en capture-avoidance restricties. We gebruiken nominale technieken die ons toelaten om asserties op meta-niveau direct te formaliseren, terwijl we dicht bij de informele praktijk blijven. We richten ons op *afleidbaarheid* en laten zien dat onze afleidingsregels het redeneren op meta-niveau ondersteunen op de volgende belangrijke punten:

- concretisering van meta-variabelen, door middel van *capturing* substitutie van termen voor meta-variabelen;
- *α-hernoeming* van object-variabelen en *capture-avoiding* substitutie van termen voor object-variabelen in de aanwezigheid van meta-variabelen;
- het genereren van *verse* object-variabelen binnen een afleiding.

We passen onze nominale technieken toe op twee logische raamwerken:

- *Equationele logica.* We bestuderen bewijs-theoretische eigenschappen, we geven een semantiek in nominale sets en we vergelijken de notie van α-hernoeming met de bestaande noties van α-gelijkheid met meta-variabelen. We geven ook een axiomatisering van capture-avoiding substitutie, en tonen aan dat deze compleet is met betrekking tot de gebruikelijke notie van capture-avoiding substitutie.
- *Eerste-orde logica met gelijkheid.* We presenteren een sequent calculus met meta-variabelen en capture-avoiding substitutie, en tonen aan dat deze een schema van afleidingen in eerste-orde logica representeert. We laten ook zien hoe we deze notie van afleidbaarheid kunnen axiomatiseren in de calculus voor equationele logica.

Curriculum Vitae

Aad Mathijssen was born on the 5th of February 1979 in Breda. In August 1997 he completed secondary school (VWO) at the Mencia de Mendoza Lyceum in Breda. In September 1997 Aad started to study Computer Science at the Technische Universiteit Eindhoven. He obtained his Master's degree in June 2003 on his master's thesis entitled *Formal Derivations of Binary Arithmetic*. In the meantime he also worked part-time as a lead software engineer for the Dutch company Rare Records.

From September 2003 Aad worked as a Ph.D. Student at the Design and Analysis of Systems Group at the Computer Science department of the Technische Universiteit Eindhoven. The first three years he was involved in the *GenSpecT* project, and the fourth year he was involved in the NWO funded project *VeriGEM: A Verification Grid for Enhanced Model Checking*.

Index

abstractor, 7
action
  permutation
    meta-level, 12
    nominal sets, 34
    object-level, 11
    ZFA, 124
  substitution
    ground, 52
    meta-level, 11
α-equivalence, 56
arity, 6, 49, 87
atom, 6
axiom, 9

classical propositional logic, 110
closed term, 8
completeness, 40
  SUB, 77, 79
confluence, 64
  SIMPr, 66
conservativity
  SIMP over CORE, 66
  SUB over CORE, 78
consistency
  CORE, 27
  FOL, 118
  SIMP, 67
  one-and-a-halfth-order logic, 102
context
  formula, 90
  freshness, 8
cut-elimination, 102

decidability
  CORE, 27
  SIMP, 70
  SUB, 78
depth, 99
derivability
  alpha-equivalence
    syntax-directed, 25
  α-equivalence, 56
  equality, 14
  freshness, 13
  Gentzen's sequent calculus, 103
  one-and-a-halfth-order logic, 91
    admissible, 100
difference set, 22

equality, 9
  derivability, 14
  syntactic, 8
equivariance, 35
  meta-level, 20, 96
  object-level, 21, 97
  ZFA, 123

finite support
  nominal sets, 34
  permutation, 7
  substitution, 11
formula, 88
  classical logic sugar, 89
  context, 90
free atoms, 52
free term, 39
fresh atom, 96
freshly extends, 24
freshness, 8

context, 8
derivability, 13
nominal sets, 34
primitive, 8

ground substitution action, 52
ground term, 52
size, 52

interpretation, 36

judgement form
equality, 9
freshness, 9

model
free term, 39
ground term, 52
nominal sets, 37

nominal algebra, 6
nominal rewrite system, 62
SIMPr, 65
transitive reflexive closure, 63
nominal set, 34
normal form
SIMPr, 66

occurrence, 8
ω-completeness, 79
one-and-a-halfth-order logic, 86

permutation, 7
meta-level action, 12
nominal sets action, 34
object-level action, 11
ZFA action, 124

sequent
Gentzen, 103
one-and-a-halfth-order, 90
signature, 7
suitable for SUB, 50
size
ground term, 52

sort, 48, 87
abstraction, 48, 87
formulae, 87
terms, 48, 87
sorting assertion, 49, 88
soundness
SIMP, 67
SUB, 72
nominal sets, 37
strong normalisation, 64
SIMPr, 69
substitution
ground action, 52
meta-level, 11, 23, 98
action, 11
closing, 67
Δ-consistent, 67
ground, 79
term-former, 10, 49
support, 34
swapping, 7

term, 7
closed, 8
free, 38
ground, 52
term-former, 6
object-level, 87
predicate, 87
sorting arity, 49, 87
substitution, 10, 49
theory, 9
CORE, 10, 25
FOL, 9, 109
LAM, 9, 49
SIMP, 58
SUB, 10, 50

unknown, 6, 49
moderated, 7

validity, 36, 37
valuation, 36

ZFA set theory, 124

Titles in the IPA Dissertation Series since 2002