Curry-Howard for incomplete first-order logic derivations using one-and-a-half level terms

Murdoch J. Gabbay[1], Dominic P. Mulligan[2]

Heriot-Watt University, Scotland, UK

Abstract

The Curry-Howard correspondence connects natural deduction derivation with the lambda-calculus. Predicates are types, derivations are terms. This supports reasoning from assumptions to conclusions, but we may want to reason backwards; from the desired conclusion towards the assumptions. At intermediate stages we may have a partial derivation, with holes.

This is natural in informal practice but it can be difficult to formalise. The informal act of filling holes in a partial derivation suggests a capturing substitution, since holes may occur in the scope of quantifier introduction rules. As other authors have observed, this is not immediately supported by the lambda-calculus. Also, universal quantification requires a 'fresh name' and it is not immediately obvious what formal meaning to assign to this notion if derivations are incomplete. Further issues arise with proof-normalisation; this corresponds with lambda-calculus reduction, which can require alpha-conversion to avoid capture when beta-reducing, and it is not immediately clear how to alpha-convert a name in an incomplete derivation.

We apply a one-and-a-half level technique based on nominal terms to construct a Curry-Howard correspondence for first-order logic. This features *two* levels of variable, but with no lambda-abstraction at the second level. Predicates are types, derivations are terms, proof-normalisation is reduction — and the two levels of variable are respectively the assumptions and the holes of an incomplete derivation.

We give notions of proof-term, typing, alpha-conversion and beta-reduction for our syntax. We prove confluence, we exhibit several admissible rules including a proof that instantiation of level two variables is type-safe — this corresponds with the act of filling holes in an incomplete derivation, and can be viewed as a form of Cut-rule — and we explore the connection with traditional Curry-Howard in the case that the derivation is in fact complete.

Our techniques are not specifically tailored to first-order logic and the same ideas should be applicable without any essential new difficulties to similar logical systems.

[1] http://www.gabbay.org.uk

[2] http://www.macs.hw.ac.uk/~dpm8

Contents

1 Introduction **3**
1.1 Overview of the paper 5

2 Syntax of types (predicates) and terms (derivations), with examples **6**
2.1 Terms, types and typable terms 6
2.2 Properties of the permutation action 10
2.3 Level two substitution 13
2.4 Examples 14

3 Admissible rules **15**
3.1 Weaken with a variable of level two (**WeakX**) 15
3.2 Weaken with a variable of level one (**Weaka**) 17
3.3 Cut 18

4 Natural deduction **21**

5 Reductions/proof-normalisation **24**
5.1 Commuting properties of (Tfr) 25
5.2 Subject reduction 28
5.3 Confluence 32

6 Conclusions **39**
6.1 Related work 40
6.2 Future work 43

A ZFA equivariance **48**

1. Introduction

The Curry-Howard correspondence [US06, PCW05] corresponds logic with typed λ-calculus as follows:

predicates		types
derivations		terms
discharge	correspond(s) with	λ-abstraction
modus-ponens		application
β-reduction		proof-normalisation

For example,[3]

$$\dfrac{\dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}}{\mathsf{B}} \qquad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\dfrac{\mathsf{C}}{\mathsf{A}\Rightarrow\mathsf{C}}\,{}^a} \quad \text{corresponds with} \quad \lambda a.((pa)qa) \tag{1}$$

where a has type A, p has type $\mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}$, and q has type $\mathsf{A}\Rightarrow\mathsf{B}$.

The typed λ-calculus yields a model of the 'forwards' construction of derivations; we plug together derivations to form larger ones, and we plug together λ-terms to form larger ones. However, we may wish to reason starting from the desired conclusion and working backwards, filling in the derivation as we go. At intermediate stages the derivation will be incomplete, for example as below with a 'hole' called X:

$$\dfrac{\dfrac{\vdots\, X}{\mathsf{B}} \qquad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\dfrac{\mathsf{C}}{\mathsf{A}\Rightarrow\mathsf{C}}\,{}^a} \qquad \textit{An incomplete derivation.}$$

Here the λ-calculus is less helpful; X corresponds with qa in the complete derivation, so (being straighforward about it) the incomplete derivation above corresponds with '$\lambda a.((pa)X)$'. But X is under a λ-binder and should be *instantiated*; substituted for without avoiding capture, and the λ-calculus does not contain a single operation to represent capturing substitution.

The reader can find extensive discussion of motivations for considering capturing substitution — which include but are not limited to representation of incomplete derivations — in [Bog02, Subsection 2.1 onwards], [Muñ97, Subsection 1.4.1], and also in [GL08, GL09, Introduction].

Most interesting logics are undecidable, so theorem-proving is often interactive (AUTOMATH [dB80] and descendents like Isabelle [Pau89] and HOL [GM93]). This motivates us to study intermediate proof-states. In addition, a well-known field of proof-theory is devoted to reasoning from conclusions to assumptions; goal-directed proof theory [GO00]. To our knowledge no Curry-Howard correspondence has been developed for goal-directed proof-theory.

This leads us to study what formal syntaxes, logics, and calculi underlie incomplete derivations and their refinement to complete ones.

[3] We are grateful to Jojgov for the examples in his paper [Joj02].

We propose an approach based on *nominal terms* [UPG04]. Nominal terms have two levels of variable: *variables of level one* $a, b, c, \ldots$ and *variables of level two* $X, Y, Z, \ldots$. Substitution of variables of level two does not avoid capture by variables of level one, so capturing substitution is directly represented by substitution of variables of level two. The incomplete derivation above is represented by the nominal term $\lambda a.((pa)X)$, and substitution of X for qa returns $\lambda a.((pa)(qa))$. Full definitions will follow.

Nominal terms can be the basis of expressive, mathematically well-behaved, and implementable systems. They were introduced in a unification algorithm (freshness conditions are constraints) [UPG04] which was implemented as the basis of a logic programming language [CU04]. The first author in collaboration has applied them to rewriting [FG07b], universal algebra [Mat07], schematic reasoning in first-order logic [GM08a], and most recently a two-level λ-calculus [GM09a][4].

In brief, nominal terms feature a two-level hierarchy of variables: **variables of level one** $a, b, c, d, \ldots$ and **variables of level two** $X, Y, Z, \ldots$. Here is an example of our system in action:

$$(\mathbf{1})\ \begin{array}{c}\vdots X\\ \bot\Rightarrow \mathsf{A}\end{array} \qquad (\mathbf{2})\ \begin{array}{c}[\bot]\\ \vdots X\\ \bot\Rightarrow \mathsf{A}\end{array} \qquad (\mathbf{3})\ \dfrac{\begin{array}{c}[\bot]\\ \vdots X'\\ \mathsf{A}\end{array}}{\bot\Rightarrow \mathsf{A}}\,(\Rightarrow\mathbf{I}) \qquad (\mathbf{4})\ \dfrac{\dfrac{[\bot]}{\mathsf{A}}\,(\bot\mathbf{E})}{\bot\Rightarrow \mathsf{A}}\,(\Rightarrow\mathbf{I})$$

$$\begin{array}{l}(\mathbf{1})\ X{:}\bot\Rightarrow A \vdash X{:}\bot\Rightarrow A\\ (\mathbf{2})\ X{:}\bot\Rightarrow A,\ a{:}\bot;\ a\#X \vdash X{:}\bot\Rightarrow A\\ (\mathbf{3})\ a{:}\bot,\ X'{:}A \vdash \lambda a.X'{:}\bot\Rightarrow A\\ (\mathbf{4})\ a{:}\bot \vdash \lambda a.\mathsf{xf}(a){:}\bot\Rightarrow A\end{array}$$

On the left is a refinement of an incomplete derivation of $\bot \Rightarrow \mathsf{A}$ to a complete derivation, represented by $\lambda a.\mathsf{xf}(a)$. Here xf (for *ex-falsum*) is a constant representing $\bot$-elimination. On the right is their representation as terms-in-context in one-and-a-half level Curry-Howard using nominal terms. Note that:

- *Assumptions* are represented by variables of level one. Types are predicates assumed.
- *Incomplete* parts of the derivation, or (terminology from theorem-proving) *subgoals*, are represented by variables of level two, and, informally, are instantiated to any term sharing their type, to obtain a well-typed term.[5] Types are predicates to be proved.
- *Freshness conditions* $a\#X$, read in the literature as 'a is fresh for X' [UPG04] mean here that 'a must be discharged in whatever X is instantiated to'.

The refinement of the lambda-term, above, follows closely the development of a proof in natural deduction; only stage $(\mathbf{2})$ is a 'technical' stage, representing the injection of a level one variable of type $\bot$ into a typing and freshness context.

We are reasonably ambitious in our choice of the types for our nominal terms. Our types correspond with first-order predicate logic without equality (see Definition 2). Note that this logic is strictly less expressive than first-order logic with function-symbols

[4]Within the context of nominal terms, we use the following naming convention: 'one-level' refers to calculi without nominal unknowns; 'one-and-a-halfth level' refers to calculi possessing unknowns but with no quantification or abstraction over them, and 'two-level' refers to calculi with nominal unknowns and the ability to quantify or abstract over them. See [GM09a] for a detailed discussion.

[5]See Figure 3 for more details.

or with equality, but because it includes a binder $\forall$ it suffices to raise the issues of interest to us in this paper. The extension to full first-order logic is future work.

This paper treats incomplete first-order logic derivations, but there seems no obvious obstacle to applying the same ideas to logics other that first-order logic. In particular, we believe that the techniques represented in this paper would extend smoothly to the application considered by Muñoz of a theory of incomplete terms-and-types in dependent type theory [Muñ97].

1.1. Overview of the paper

In Section 2 we introduce syntax in Definition 5 and typing rules for the syntax in Definition 14, with some discussion of the rules in Remark 16 and with example derivations in Subsection 2.4.

In Section 3 we describe some admissible rules, including in Subsection 3.3 a rule for instantiating variables of level two (i.e. filling in unknown parts of a derivation) which features as a form of Cut. The proof of admissibility of Cut (Lemma 35 and Theorem 36) is a proof of soundness for instantiating variables of level two, as we do when we transform a term representing an incomplete derivation to a term representing a complete one.[6]

In Section 4 we consider traditional natural deduction derivation and show that our Curry-Howard system is sound (Theorem 41) and complete (Theorem 42) for the case of terms without variables of level one (representing complete derivations). That is: once an incomplete term is instantiated to a complete term in our system, it really does represent a natural deduction derivation of the predicate represented by its type.

In Section 5 we develop a theory of β-reduction for our calculus, corresponding via the Curry-Howard paradigm to a notion of proof-normalisation. Important technical results are subject reduction (Theorem 48), the identification of canonical forms (Definition 57), and confluence (Theorem 62). β-reduction is non-trivial in our calculus, because terms containing variables of level two raise issues to do with α-equivalence and β-reduction. The reader can find discussions of this in the authors' recent publications [GL08, GL09, GM09a], as well as in other authors' work [Bog02, Subsection 2.1 onwards], [Muñ97, Subsection 1.4.1], and also in [GL08, GL09, Introduction]. The solutions used by the calculus of this paper build on ideas presented in previous work using nominal terms [GL08, GL09, GM08a, GM09a] but they are non-trivial extensions of that work, since the calculus of this paper represents derivations of first-order logic, and is therefore particularly rich.

Finally, in the Conclusions, we give a technical survey of the literature on incomplete derivations, and propose future work.

[6] This paper does not internalise proof-search. We prove that instantiation of level two variables is admissible, but we have no explicit instantiation rule, no backtracking rule, and so on. How to get from incomplete derivations to complete derivations is a separate issue, for us, from how to represent the incomplete derivations — but an interesting one which could be the topic of future work. See the discussion of McBride's system in the Conclusions.

2. Syntax of types (predicates) and terms (derivations), with examples

2.1. Terms, types and typable terms

Types (Definition 2) are the syntax of predicates of first-order logic. *Terms* (Definition 5) are the syntax of a λ-calculus. As we shall see, it will be possible to assign types to terms such that terms represent (incomplete) first-order logic derivations; this is explored with some detailed examples in Subsection 2.4.

Fix disjoint countably infinite sets of **level one** and **level two** variables. We let $a, b, c, d, \ldots$ range over level one variables, and $X, Y, Z, \ldots$ range permutatively over level two variables. We use a **permutative convention** that $a, b, c, \ldots$ and $X, Y, Z, \ldots$ range over *distinct* elements. For example 'a and b' means 'two *distinct* level one variables', and . 'X, Y, and Z' means 'three *distinct* level two variables'.

Remark 1. Level one variables are derived from nominal term *atoms*, and level two variables are derived from nominal term *unknowns* [UPG04]. In keeping with more recent work [GM09a, GL08, GL09] we prefer the 'levels' terminology. Note that the variables in this paper (also in keeping with [GM09a, GL08, GL09]) display λ-abstraction and β-reduction behaviour, which nominal terms atoms and unknowns do not display. The variables here are also typed, with types representing predicates of first-order logic as will be described below. Thus, the syntax here is not the same thing as the nominal terms of [UPG04], but if we remove β-conversion, typing, and the theorems we prove about them in this paper — then we are left with α-equivalence, and the α-equivalence used in this paper is indeed that introduced in [UPG04].

Definition 2. Fix **type-formers** $\mathsf{P}, \mathsf{Q}, \mathsf{R}$ and **type term-formers** $\mathsf{f}, \mathsf{g}, \mathsf{h}$, to each of which is associated an arity $ar(\text{-})$ which is a nonnegative integer $(0, 1, 2, \ldots)$.

Define **type terms** and **types** by:

$$g ::= a \mid \mathsf{f}(g_1, \ldots, g_{ar(\mathsf{f})})$$
$$\phi, \psi, \xi ::= \bot \mid \phi \Rightarrow \phi \mid \mathsf{P}(g_1, \cdots, g_{ar(\mathsf{P})}) \mid \forall a.\phi$$

g will range over type terms. ϕ, ψ, and ξ will range over types.

For example $\forall a.(\mathsf{P}(a, a) \Rightarrow \mathsf{P}(a, b))$ is a type if $ar(\mathsf{P}) = 2$.

We write $\equiv$ for syntactic identity and we equate types up to $\forall$-bound variables of level one (so for example $\forall a.\forall b.\mathsf{P}(a, b) \equiv \forall b.\forall a.\mathsf{P}(b, a)$).[7] Implication associates to the right; for example $\phi \Rightarrow \psi \Rightarrow \xi \equiv \phi \Rightarrow (\psi \Rightarrow \xi)$.

Intuitively, types are first-order logic predicates.

Definition 3. Define the **free level one variables** as standard by:

$$\begin{aligned} fa(a) &= \{a\} \\ fa(\mathsf{f}(g_1, \ldots, g_n)) &= fa(g_1) \cup \cdots \cup fa(g_n) \\ fa(\mathsf{P}(g_1, \ldots, g_n)) &= fa(g_1) \cup \cdots \cup fa(g_n) \\ fa(\phi \Rightarrow \psi) &= fa(\phi) \cup fa(\psi) \\ fa(\bot) &= \varnothing \\ fa(\forall a.\phi) &= fa(\phi) \backslash \{a\} \end{aligned}$$

[7] The first author is known for studying the problem of reasoning on datatypes up to α-equivalence. We do not study that problem in this paper.

Definition 4. Call a bijection π on level one variables a **permutation** when $\{a \mid \pi(a) \neq a\}$ is finite. $\pi, \pi', \pi'', \ldots$ will range over permutations.

Write id for the **identity permutation**, so $id(a) = a$ for all variables of level one, a. Call $\pi \circ \pi'$ the **composititon** of permutations π and π', so $(\pi \circ \pi')(a) = \pi(\pi'(a))$. Write π^{-1} for the inverse of π, whereby $\pi \circ \pi^{-1} = id = \pi^{-1} \circ \pi$.

Write $(a\ b)$ for the **swapping permutation**, which sends a to b, b to a and all other c to themselves.

Definition 5. Let **terms** be:

$$r, s, t, \ldots ::= g \mid \pi \cdot X \mid \lambda a.r \mid r'r \mid \mathsf{xf}(r)$$

We write $\equiv$ for syntactic equivalence.

(We do not quotient terms by α-equivalence; thus for example $\lambda a.a \not\equiv \lambda b.b$.)

We may write $(\lambda a.r)t$ as $r[a \mapsto t]$, for example $(\lambda a.b)a \equiv b[a \mapsto a]$. We may write $r'r$ as $r'(r)$. Application associates to the left, so $r''r'r \equiv (r''r')r$; sometimes we will bracket anyway.

In Definition 5, we say π in $\pi \cdot X$ is **suspended** on X, with intuition 'π acts on whatever X is instantiated to'. Consistent with previous work [UPG04] we may write 'X' for '$id \cdot X$'. Note that later, we define $\pi \cdot r$ (Definition 11); this is meta-level notation for an action of π on r, whereas $\pi \cdot X$ is directly syntax.

Definition 6. A **type assignment** is a pair of the form $a : \phi$ where $a \notin fa(\phi)$, or $X : \phi$, or $a : *$. A **typing context** Γ is a finite set of type assignments, which is functional in the sense that:

- If $a : \phi \in \Gamma$ then $a : * \notin \Gamma$. If $a : * \in \Gamma$ then $a : \phi \notin \Gamma$.
- If $a : \phi \in \Gamma$ and $a : \phi' \in \Gamma$ then $\phi = \phi'$. Similarly for X.

As is standard we may drop set brackets, writing for example $\Gamma, a : \phi$ for $\Gamma \cup \{a : \phi\}$. Intuitively, $a : \phi$ means 'a has type ϕ'; $a : *$ means 'a is a type variable'; $X : \phi$ means 'X has type ϕ'.

Remark 7. We use the same syntactic class (variables of level one) to represent type variables and term variables. The typing context differentiates them; $a : \phi \in \Gamma$ means a behaves like a term variable; $a : * \in \Gamma$ means a behaves like a type variable.

We could make a syntactic separation between variables of level one that can have types ($a : \phi \in \Gamma$), and variables of level one that can appear in types ($a : * \in \Gamma$). However, we would duplicate the treatments of λ-abstraction, application, and freshness. Our approach keeps the machinery shorter, at the expense of raw syntax being less inherently 'sorted'.

Definition 8. Call a pair $a\#r$ of a variable of level one and a term a **freshness**. Call a freshness of the form $a\#X$ **primitive**. Call a finite set of primitive freshnesses a **freshness context**. Δ will range over freshness contexts.

Definition 9. Call $\Gamma; \Delta \vdash r$ a **term-in-context**. Call $\Gamma; \Delta \vdash r : \phi$ a **typing sequent**. Call $\Gamma; \Delta \vdash a\#r$ a **freshness sequent**.

$$\frac{(\{a : * \mid a \in fa(g)\} \subseteq \Gamma)}{\Gamma; \Delta \vdash g : *}\,(\mathbf{Tg}*) \qquad \frac{(a : \phi \in \Gamma,\ \{b : * \mid b \in fa(\phi)\} \subseteq \Gamma)}{\Gamma; \Delta \vdash a : \phi}\,(\mathbf{Ta}\phi)$$

$$\frac{(X : \phi \in \Gamma,\ \Gamma \vdash \pi,\ \{b : * \mid b \in fa(\phi)\} \subseteq \Gamma,\ \{b\#X \mid b \in fa(\phi)\} \cap \Delta = \varnothing)}{\Gamma; \Delta \vdash \pi \cdot X : \pi \cdot \phi}\,(\mathbf{TX})$$

$$\frac{\Gamma; \Delta \vdash r : \bot}{\Gamma; \Delta \vdash \mathsf{xf}(r) : \phi}\,(\mathbf{T}\bot\mathbf{E}) \qquad \frac{\Gamma;\ \Delta \vdash r' : \phi \Rightarrow \psi \quad \Gamma;\ \Delta \vdash r : \phi}{\Gamma;\ \Delta \vdash r'r : \psi}\,(\mathbf{T}\Rightarrow\mathbf{E})$$

$$\frac{\Gamma;\ \Delta \vdash r : \forall a.\phi \quad \Gamma;\ \Delta \vdash g : *}{\Gamma;\ \Delta \vdash rg : \phi[a := g]}\,(\mathbf{T}\forall\mathbf{E}) \qquad \frac{\Gamma,\ a : \phi;\ \Delta \vdash r : \psi}{\Gamma,\ a : \phi;\ \Delta \vdash \lambda a.r : \phi \Rightarrow \psi}\,(\mathbf{T}\Rightarrow\mathbf{I})$$

$$\frac{\Gamma,\ a : *;\ \Delta \vdash r : \phi \quad a \notin fa(important(\Gamma,\ a : *; \Delta \vdash r))}{\Gamma,\ a : *;\ \Delta \vdash \lambda a.r : \forall a.\phi}\,(\mathbf{T}\forall\mathbf{I})$$

$$\frac{\Gamma,\ A;\ \Delta, b\#\mathcal{X} \vdash r : \phi\ \ \Gamma,\ A;\ \Delta, b\#\mathcal{X} \vdash b\#r\ \ (A \in \{b : \psi,\ b : *\},\ b \notin \Delta)}{\Gamma; \Delta \vdash r : \phi}\,(\mathbf{Tfr})$$

$$\frac{}{\Gamma,\ a : \phi,\ b : \phi; \Delta \vdash a\#b}\,(\mathbf{a}\#\mathbf{b}) \quad \frac{(a \notin fa(g))}{\Gamma; \Delta \vdash a\#g}\,(\mathbf{a}\#\mathbf{g})$$

$$\frac{}{\Gamma; \Delta \vdash a\#\lambda a.r}\,(\mathbf{a}\#\lambda\mathbf{a}) \qquad \frac{\Gamma; \Delta \vdash a\#r \quad \Gamma; \Delta \vdash a\#b}{\Gamma; \Delta \vdash a\#\lambda b.r}\,(\mathbf{a}\#\lambda\mathbf{b})$$

$$\frac{\pi^{-1}(a)\#X \in \Delta}{\Gamma; \Delta \vdash a\#\pi \cdot X}\,(\mathbf{a}\#\mathbf{X}) \qquad \frac{(a \notin fa(\phi))}{\Gamma,\ a : *,\ b : \phi; \Delta \vdash a\#b}\,(\mathbf{a}\#\mathbf{b}')$$

$$\frac{\Gamma; \Delta \vdash a\#r' \quad \Gamma; \Delta \vdash a\#r}{\Gamma; \Delta \vdash a\#r'r}\,(\mathbf{a}\#\mathbf{app}) \qquad \frac{\Gamma; \Delta \vdash a\#r}{\Gamma; \Delta \vdash a\#\mathsf{xf}(r)}\,(\mathbf{a}\#\mathsf{xf})$$

$$\frac{\bot}{\phi}\,(\bot\mathbf{E}) \qquad \frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \Rightarrow \psi}\,(\Rightarrow\mathbf{I}) \qquad \frac{\phi \Rightarrow \psi \quad \phi}{\psi}\,(\Rightarrow\mathbf{E})$$

$$\frac{\begin{array}{c}\Phi\\ \vdots\\ \phi\end{array} \quad (a \notin fa(\Phi))}{\forall a.\phi}\,(\forall\mathbf{I}) \qquad \frac{\forall b.\phi}{\phi[b := g]}\,(\forall\mathbf{E})$$

Figure 2: Natural deduction style derivation rules

Definition 10. We define some useful notions for working with types and freshnesses:

- If Φ is a set of types, write $fa(\Phi)$ for $\bigcup\{fa(\phi) \mid \phi \in \Phi\}$.
- If $\mathcal{X}$ is a set of variables of level two, write $a\#\mathcal{X}$ for the freshness context $\{a\#X \mid X \in \mathcal{X}\}$.
- Write $b \notin \Delta$ when $b\#X \notin \Delta$ for all X.

Definition 11. Define **permutation actions** on type terms, types, and terms by:

$$\begin{array}{rlrl} \pi \cdot a &\equiv \pi(a) & \pi \cdot \mathsf{f}(g_1, \ldots, g_n) &\equiv \mathsf{f}(\pi \cdot g_1, \ldots, \pi \cdot g_n) \\ \pi \cdot \bot &\equiv \bot & \pi \cdot (\phi \Rightarrow \psi) &\equiv (\pi \cdot \phi) \Rightarrow (\pi \cdot \psi) \\ \pi \cdot \forall a.\phi &\equiv \forall \pi(a).\pi \cdot \phi & \pi \cdot \mathsf{P}(g_1, \ldots, g_n) &\equiv \mathsf{P}(\pi \cdot g_1, \ldots, \pi \cdot g_n) \\ \pi \cdot (\pi' \cdot X) &\equiv (\pi \circ \pi') \cdot X & \pi \cdot (r'r) &\equiv (\pi \cdot r')(\pi \cdot r) \\ \pi \cdot \lambda a.r &\equiv \lambda \pi(a).(\pi \cdot r) & \pi \cdot \mathsf{xf}(r) &\equiv \mathsf{xf}(\pi \cdot r) \end{array}$$

Definition 12. Define a **substitution action** $\phi[a := b]$ on types by the usual capture-avoiding substitution on predicates.

For example, if f is a type term former with arity 2 and $\sim$ is a type former of arity 2, then $\big(\forall b.(\mathsf{f}(a,b) \sim c)\big)[a := \mathsf{f}(b,c)] = \forall b'.(\mathsf{f}(\mathsf{f}(b,c),b') \sim c$.

Definition 13. Write $\Gamma \vdash \pi$ when for every a such that $\pi(a) \neq a$, precisely one of the following holds:

- $a : * \in \Gamma$ and $\pi(a) : * \in \Gamma$,
- $a : \phi \in \Gamma$ and $\pi(a) : \pi \cdot \phi \in \Gamma$, for some ϕ.

Definition 14. Let the **derivable** typing and freshness sequents be inductively defined by the rules in Figure 1. We use the following notation here and later:

- Side-conditions are written in brackets.
- A ranges over typings or freshnesses, so $A \in \{r : \phi,\ a : *,\ a\#r\}$.
- If a sequent $\Gamma \vdash -$ is *not* derivable we write $\Gamma \nvdash -$.
- We write $important(\Gamma; \Delta \vdash r)$ for $\{\phi \mid a : \phi \in \Gamma,\ \Gamma; \Delta \nvdash a\#r\}$.

If ϕ exists such that $\Gamma; \Delta \vdash r : \phi$ is derivable, call $\Gamma; \Delta \vdash r$ **typable**.

We may write '$\Gamma; \Delta \vdash r : \phi$' for '$\Gamma; \Delta \vdash r : \phi$ is a derivable typing sequent', and similarly for '$\Gamma; \Delta \vdash a\#r$'.

Remark 15. A few brief comments on these rules:

- The side-condition $b \notin \Delta$ in (**Tfr**) ensures that (**Tfr**) removes all mention of b from the freshness context.
- (**TX**) is syntax-directed on the term being typed ($\pi \cdot X$). Note that $\pi \cdot \phi$ has the same top-level type term-former as ϕ; we merely permute the level one variables.

- A way to read $important(\Gamma; \Delta \vdash r)$, recalling that r represents an incomplete natural deduction derivation, is: "the possible assumptions of instantiations of r to complete natural deduction derivations".
- The condition $\Gamma; \Delta \vdash a\#b$ in $(\mathbf{a}\#\lambda\mathbf{b})$ is not redundant. For example, it fails if $a : * \in \Gamma$, $b : \psi \in \Gamma$, and $a \in fa(\psi)$.

Remark 16. We compare the rules in Figure 1, for typing nominal terms, with the rules in Figure 2, which describe standard natural deduction:

- Compare $(\mathbf{T}\bot\mathbf{E})$ with $(\bot\mathbf{E})$. 'xf' stands for *ex falsum*. $(\mathbf{T}\bot\mathbf{E})$ corresponds with $(\bot\mathbf{E})$ in a standard way. No surprises here.
- Compare $(\mathbf{T}\Rightarrow\mathbf{I})$ with $(\Rightarrow\mathbf{I})$. $(\mathbf{T}\Rightarrow\mathbf{I})$ does not discharge $a : \phi$ because r may contain a variable of level two X. We intend X to be instantiated to t which (because instantiation need not avoid capture) may mention a; see Definition 26. We remember $a : \phi$ in the typing context so that we can use it to build t, if we like. This is not a problem: we can mimic $(\Rightarrow\mathbf{I})$ using $(\mathbf{T}\Rightarrow\mathbf{I})$ and $(\mathbf{Tfr})$. An effect of this design choice is that the names of 'bound' variables in terms really do matter. For example, using $(\mathbf{T}\Rightarrow\mathbf{I})$ and $(\mathbf{Tfr})$:
  - $\varnothing; \varnothing \vdash \lambda a.\lambda a.a : \phi \Rightarrow \phi \Rightarrow \phi$ is derivable.
  - ~~$\varnothing; \varnothing \vdash \lambda a.\lambda a.a : \psi \Rightarrow \phi \Rightarrow \phi$~~ is not derivable.
  - $\varnothing; \varnothing \vdash \lambda b.\lambda a.a : \psi \Rightarrow \phi \Rightarrow \phi$ is derivable.

  We could 'solve' this problem by introducing α-equivalence into the typing rules. We choose not to; it makes no difference for the theorems of interest to us.
- Compare $(\mathbf{T}\forall\mathbf{I})$ with $(\forall\mathbf{I})$. $a \notin fa(\Phi)$ is intuitively 'a is not free in any of the assumptions Φ used to prove ϕ'. $a \notin fa(important(\Gamma, a{:}*; \Delta \vdash r))$ generalises this to take account of variables of level two and freshness assumptions on them.
- Compare $(\mathbf{a}\#\mathbf{b})$ and $(\mathbf{a}\#\mathbf{b}')$. $(\mathbf{a}\#\mathbf{b})$ is as in [UPG04]; distinct variables of level one are fresh. In $(\mathbf{a}\#\mathbf{b}')$ we account for the *type* of b. For example:

$$a : \mathsf{P}(c),\ X : \mathsf{P}(c),\ c : *;\ a\#X \vdash a\#X$$
$$a{:}\mathsf{P}(c),\ X{:}\mathsf{P}(c),\ c{:}*;\ a\#X \not\vdash c\#X \qquad a{:}\mathsf{P}(c),\ X{:}\mathsf{P}(c),\ c{:}*;\ a\#X \not\vdash c\#a$$

2.2. *Properties of the permutation action*

Here, we collect properties of the permutation action which will be useful later.

Lemma 17. *If* $\Gamma \vdash \pi$ *(Definition 13) and* $\Gamma \vdash \pi'$ *then* $\Gamma \vdash \pi \circ \pi'$.

Proof. Routine from the definitions. □

Lemma 18. $\pi' \cdot (\pi \cdot r) \equiv (\pi' \circ \pi) \cdot r$

Proof. By induction on r.

- The case a. As permutations are a bijection on variables of level one.
- The case $\pi \cdot X$. From the definition of the permutation action.
- The case $r'r$. We have $(\pi \circ \pi') \cdot r'r \equiv ((\pi \circ \pi') \cdot r')((\pi \circ \pi') \cdot r)$. By hypothesis, this is equivalent to $(\pi \cdot (\pi' \cdot r'))(\pi \cdot (\pi' \cdot r))$. The result follows from the permutation action.
- The case $\lambda a.r$. We have

  $$\pi' \cdot (\pi \cdot \lambda a.r) \equiv \pi' \cdot (\lambda \pi(a).(\pi \cdot r)) \equiv \lambda((\pi' \circ \pi) \cdot a).((\pi' \circ \pi) \cdot r)$$

  and the result follows.
- The case $\mathsf{xf}(r)$. We have $(\pi \circ \pi') \cdot \mathsf{xf}(r) \equiv \mathsf{xf}((\pi \circ \pi') \cdot r)$. By hypothesis, this is equivalent to $\mathsf{xf}(\pi \cdot (\pi' \cdot r))$. The result follows from the permutation action definition.

□

Lemma 19. *If* $\Gamma; \Delta \vdash a\#r$ *and* $\Gamma \vdash \pi$ *then* $\Gamma; \Delta \vdash \pi(a)\#\pi \cdot r$.

Proof. By induction on the derivation of $\Gamma; \Delta \vdash a\#r$.

- The case $(\mathbf{a}\#\mathbf{b})$. Suppose $a : \phi \in \Gamma$ and $b : \phi \in \Gamma$. Since $\Gamma \vdash \pi$ also $\pi(a) : \phi \in \Gamma$ and $\pi(b) : \phi \in \Gamma$. By $(\mathbf{a}\#\mathbf{b})$, also $\Gamma \vdash \pi(a)\#\pi(b)$.
- The case $(\mathbf{a}\#\mathbf{g})$ is similar.
- The case $(\mathbf{a}\#\mathbf{b}')$. Suppose $a : * \in \Gamma$ and $b : \phi \in \Gamma$ and $a \notin fa(\phi)$. Since $\Gamma \vdash \pi$ also $\pi(a) : * \in \Gamma$ and $\pi(b) : \pi \cdot \phi \in \Gamma$. By $(\mathbf{a}\#\mathbf{b}')$, also $\Gamma \vdash \pi(a)\#\pi(b)$.
- The case $(\mathbf{a}\#\lambda\mathbf{a})$. $\Gamma; \Delta \vdash \pi(a)\#\lambda\pi(a).(\pi \cdot r)$ always, using $(\mathbf{a}\#\lambda\mathbf{a})$.
- The case $(\mathbf{a}\#\lambda\mathbf{b})$. By inductive hypothesis, $\Gamma; \Delta \vdash \pi(a)\#\pi \cdot r$ and $\Gamma; \Delta \vdash \pi(a)\#\pi \cdot b$. By $(\mathbf{a}\#\lambda\mathbf{b})$ also $\Gamma; \Delta \vdash \pi(a)\#\lambda\pi(b).(\pi \cdot r)$. The result follows.
- The case $(\mathbf{a}\#\mathbf{X})$. Suppose $a\#\pi' \cdot X$ is derived using $(\mathbf{a}\#\mathbf{X})$, so that ${\pi'}^{-1}(a)\#X \in \Delta$. It is a fact that ${\pi'}^{-1}(a) = \pi \circ {\pi'}^{-1}(\pi(a))$. The result follows.
- The cases of $(\mathbf{a}\#\mathbf{app})$ and $(\mathbf{a}\#\mathsf{xf})$ are no harder.

□

Lemma 20. *Suppose* $\Gamma \vdash \pi$. *Then* $\phi \in important(\Gamma; \Delta \vdash r)$ *if and only if* $\pi \cdot \phi \in important(\Gamma; \Delta \vdash \pi \cdot r)$.

Proof. Suppose $\Gamma \vdash \pi$. Suppose $\phi \in important(\Gamma; \Delta \vdash r)$. By definition $a : \phi \in \Gamma$. Since $\Gamma \vdash \pi$, also $\pi(a) : \pi \cdot \phi \in \Gamma$. Also by definition, $\Gamma; \Delta \not\vdash a\#r$. By Lemma 19 $\Gamma; \Delta \not\vdash \pi(a)\#\pi \cdot r$.

The argument for $\pi \cdot \phi \in important(\Gamma; \Delta \vdash r)$ is similar. The result follows. □

Lemma 21. $fa(\pi \cdot \phi) = \{\pi(a) \mid a \in fa(\phi)\}$.

Proof. By a routine induction on ϕ. □

Lemma 22. *If $a \notin fa(\phi)$ and $b \notin fa(\phi)$ then $(a\ b) \cdot \phi = \phi$.*

Proof. By a routine induction on ϕ. □

Lemma 23. *If $\Gamma; \Delta \vdash r : \phi$ and $\Gamma \vdash \pi$ then $\Gamma; \Delta \vdash \pi \cdot r : \pi \cdot \phi$.*

Proof. By induction on the derivation of $\Gamma; \Delta \vdash r : \phi$.

- The case ($\mathbf{Ta}*$). Suppose $a : * \in \Gamma$. Since $\Gamma \vdash \pi$ also $\pi(a) : * \in \Gamma$. The result follows.
- The case ($\mathbf{Ta}\phi$). Suppose $a : \phi \in \Gamma$. Since $\Gamma \vdash \pi$ also $\pi(a) : \pi \cdot \phi \in \Gamma$. The result follows.
- The case ($\mathbf{TX}$). By a routine calculation on permutations.
- The cases ($\mathbf{T}\bot\mathbf{E}$), ($\mathbf{T}\Rightarrow\mathbf{E}$), and ($\mathbf{T}\Rightarrow\mathbf{I}$) are routine.
- The case ($\mathbf{T}\forall\mathbf{E}$). Suppose $\Gamma; \Delta \vdash r : \forall a.\phi$ and $\Gamma; \Delta \vdash g : *$. By inductive hypothesis $\Gamma; \Delta \vdash \pi \cdot r : \forall \pi(a).\pi \cdot \phi$. Since $\Gamma; \Delta \vdash g : *$, $x : * \in \Gamma$ for every $x \in fa(g)$. Since $\Gamma \vdash \pi$ also $\pi(x) : * \in \Gamma$ for every $x \in fa(g)$. It is a fact that $\pi \cdot (\phi[a := g]) = (\pi \cdot \phi)[\pi(a) := \pi \cdot g]$. Using ($\mathbf{T}\forall\mathbf{E}$) it follows that $\Gamma; \Delta \vdash (\pi \cdot r)(\pi \cdot g) : (\pi \cdot \phi)[\pi \cdot a := \pi \cdot g]$.
- The case ($\mathbf{T}\forall\mathbf{I}$). Suppose $\Gamma, a : *; \Delta \vdash r : \phi$ and $a \notin important(\Gamma, a : *; \Delta \vdash r)$. By inductive hypothesis $\Gamma, a : *; \Delta \vdash \pi \cdot r : \pi \cdot \phi$. By Lemma 20 $\pi \cdot a \notin important(\Gamma, a : *; \Delta \vdash \pi \cdot r)$. By assumption since $\Gamma, a : * \vdash \pi$ also $\pi(a) : * \in \Gamma, a : *$. The result follows.
- The case ($\mathbf{Tfr}$). Similar to the previous case, using Lemma 19.

□

Lemma 24. *Suppose $\Gamma; \Delta \vdash r : \phi$. Then if $\Gamma; \Delta \vdash a\#r$ then $a \notin fa(\phi)$.*

Proof. By induction on the derivation of $\Gamma; \Delta \vdash r : \phi$.

- The case ($\mathbf{Ta}\phi$). There are two cases:
  - $\Gamma; \Delta \vdash a\#b$ is derived using ($\mathbf{a\#b}$). So $a : \phi, b : \phi \in \Gamma$. By assumption, $a \notin fa(\phi)$, as required.
  - $\Gamma; \Delta \vdash a\#b$ is derived using ($\mathbf{a\#b'}$). So $a : *, b : \phi \in \Gamma$ and $a \notin fa(\phi)$.
- The case ($\mathbf{TX}$). $\Gamma; \Delta \vdash a\#\pi \cdot X$ must be derived using ($\mathbf{a\#X}$). It follows that $\pi^{-1}(a)\#X \in \Delta$. By the side-condition in ($\mathbf{TX}$), $\pi^{-1}(a) \notin fa(\phi)$. Therefore, $a \notin fa(\pi \cdot X)$.
- The cases of ($\mathbf{T}\bot\mathbf{E}$) and ($\mathbf{T}\Rightarrow\mathbf{E}$) are routine.
- The case ($\mathbf{T}\forall\mathbf{E}$). $\Gamma; \Delta \vdash a\#rg : \phi[b := g]$ must be derived using ($\mathbf{a\#app}$). Then $\Gamma; \Delta \vdash a\#r$ and $\Gamma; \Delta \vdash a\#g$. By inductive hypothesis $a \notin fa(\forall b.\phi)$, and by the structure of ($\mathbf{a\#g}$) also $a \notin fa(g)$. It is then a fact that $a \notin fa(\phi[b := g])$.

- The case ($\mathbf{T}\forall\mathbf{I}$). There are two cases.
  - The case of $\Gamma, a : *; \Delta \vdash \lambda a.r : \forall a.\phi$. $a \notin fa(\forall a.\phi)$ is immediate.
  - The case of $\Gamma, b : *; \Delta \vdash \lambda b.r : \forall b.\phi$. Since $a \in fa(\phi)$ if and only if $a \in fa(\forall b.\phi)$.
- The case ($\mathbf{Tfr}$). Immediate, since ϕ is not changed above and below the line.

□

We use Lemma 25 much later, in the proof of Theorem 48. We place it here because it fits naturally into the results in this subsection.

Lemma 25. *Suppose* $\Gamma; \Delta \vdash r : \phi$, $\Gamma \vdash (a\ b)$, $\Gamma; \Delta \vdash a\#r$, *and* $\Gamma; \Delta \vdash b\#r$. *Then* $\Gamma; \Delta \vdash (a\ b) \cdot r : \phi$.

Proof. Suppose $\Gamma; \Delta \vdash r : \phi$, $\Gamma \vdash (a\ b)$, $\Gamma; \Delta \vdash a\#r$, and $\Gamma; \Delta \vdash b\#r$. By Lemma 23 $\Gamma; \Delta \vdash (a\ b) \cdot r : (a\ b) \cdot \phi$. By Lemma 24 $a \notin fa(\phi)$ and $b \notin fa(\phi)$. By Lemma 22 $\Gamma; \Delta \vdash (a\ b) \cdot r : \phi$. □

2.3. Level two substitution

The purpose of level two variables X is to be instantiated; they represent 'unknown parts of the proof'. We now formally define this instantiation action:

Definition 26. Define a **substitution** action $r[X := t]$ by:

$$\begin{aligned} a[X := t] &\equiv a \\ (\pi \cdot X)[X := t] &\equiv \pi \cdot t \\ (\pi \cdot Y)[X := t] &\equiv \pi \cdot Y \\ (\lambda a.r)[X := t] &\equiv \lambda a.(r[X := t]) \\ (r'r)[X := t] &\equiv (r'[X := t])(r[X := t]) \\ \mathsf{xf}(r)[X := t] &\equiv \mathsf{xf}(r[X := t]) \end{aligned}$$

Lemma 27. $\pi \cdot (r[X := t]) \equiv (\pi \cdot r)[X := t]$

Proof. By induction on r.

- The case $\pi \cdot (a[X := t])$. Since $a[X := t] \equiv a$ and $\pi(a)[X := t] \equiv \pi(a)$.
- The case $\pi' \cdot ((\pi' \cdot X)[X := t])$. $\pi \cdot (\pi' \cdot X)[X := t] \equiv \pi \cdot \pi' \cdot t$. The result follows from Lemma 18.
- The case $\pi' \cdot ((\pi' \cdot X)[Y := t])$. $(\pi' \cdot X)[Y := u] \equiv \pi' \cdot X$ and similarly for $(\pi \circ \pi') \cdot X$. The result follows.
- The case $r'r$. We have $\pi \cdot ((r'r)[X := t]) \equiv (\pi \cdot (r[X := t]))(\pi \cdot (r[X := t]))$. By inductive hypothesis, $(\pi \cdot (r[X := t]))(\pi \cdot (r[X := t])) \equiv ((\pi \cdot r)[X := t])((\pi \cdot r)[X := t])$. The result follows.
- The case $\lambda a.r$. We have $\pi \cdot ((\lambda a.r)[X := t]) \equiv \lambda\pi(a).(\pi \cdot (r[X := t]))$. By inductive hypothesis, $\lambda\pi(a).(\pi \cdot (r[X := t])) \equiv \lambda\pi(a).((\pi \cdot r)[X := t])$. The result follows.
- The case $\mathsf{xf}(r)$. We have $\pi \cdot (\mathsf{xf}(r)[X := t]) \equiv \mathsf{xf}(\pi \cdot (r[X := t]))$. By inductive hypothesis, $\mathsf{xf}(\pi \cdot (r[X := t])) \equiv \mathsf{xf}((\pi \cdot r)[X := t])$. The result follows.

□

2.4. Examples

The derivations below type terms representing derivations from the Introduction; one is complete, the other incomplete. At each stage the term being typed represents a (possibly incomplete) natural deduction derivation. Write '$\Gamma; \varnothing \vdash r$' as '$\Gamma \vdash r$'. Write Γ for $a : \mathsf{A},\ p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C},\ q : \mathsf{A} \Rightarrow \mathsf{B}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash a : \mathsf{A}}\,(\mathbf{Ta}\phi) \quad \dfrac{}{\Gamma \vdash q : \mathsf{A} \Rightarrow \mathsf{B}}\,(\mathbf{Ta}\phi)}{\Gamma \vdash qa : \mathsf{B}}\,(\mathbf{T}{\Rightarrow}\mathbf{E}) \quad \dfrac{\dfrac{}{\Gamma \vdash a : \mathsf{A}}\,(\mathbf{Ta}\phi) \quad \dfrac{}{\Gamma \vdash p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}}\,(\mathbf{Ta}\phi)}{\Gamma \vdash pa : \mathsf{B} \Rightarrow \mathsf{C}}\,(\mathbf{T}{\Rightarrow}\mathbf{E})}{\Gamma \vdash (pa)qa : \mathsf{C}}\,(\mathbf{Ty}{\Rightarrow}\mathbf{E})}{\Gamma \vdash \lambda a.((pa)qa) : \mathsf{A} \Rightarrow \mathsf{C}}\,(\mathbf{T}{\Rightarrow}\mathbf{I})}{p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C},\ q : \mathsf{A} \Rightarrow \mathsf{B} \vdash \lambda a.((pa)qa) : \mathsf{A} \Rightarrow \mathsf{C}}\,(\mathbf{Tfr})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,\ X : \mathsf{B} \vdash X : \mathsf{B}}\,(\mathbf{Ta}\phi) \quad \dfrac{\dfrac{}{\Gamma,\ X : \mathsf{B} \vdash a : \mathsf{A}}\,(\mathbf{Ta}\phi) \quad \dfrac{}{\Gamma,\ X : \mathsf{B} \vdash p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}}\,(\mathbf{Ta}\phi)}{\Gamma,\ X : \mathsf{B} \vdash pa : \mathsf{B} \Rightarrow \mathsf{C}}\,(\mathbf{T}{\Rightarrow}\mathbf{E})}{\Gamma,\ X : \mathsf{B} \vdash (pa)X : \mathsf{C}}\,(\mathbf{T}{\Rightarrow}\mathbf{E})}{\Gamma,\ X : \mathsf{B} \vdash \lambda a.((pa)X) : \mathsf{A} \Rightarrow \mathsf{C}}\,(\mathbf{T}{\Rightarrow}\mathbf{I})}{p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C},\ q : \mathsf{A} \Rightarrow \mathsf{B},\ X : \mathsf{B} \vdash \lambda a.((pa)X) : \mathsf{A} \Rightarrow \mathsf{C}}\,(\mathbf{Tfr})$$

Derivations of $\Gamma \vdash a\#\lambda a.((pa)qa)$ and $\Gamma,\ X : \mathsf{B} \vdash a\#\lambda a.((pa)X)$ are elided.

Another example illustrates the side-condition on $(\mathbf{T}\forall\mathbf{I})$. The two derivations

$$\dfrac{\mathsf{A} \quad \dfrac{\dfrac{\forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c))}{\mathsf{A} \Rightarrow \mathsf{P}(c)}\,(\forall\mathbf{E})}{\mathsf{P}(c)}\,(\Rightarrow\mathbf{E})}{\dfrac{\mathsf{P}(c)}{\forall c.\mathsf{P}(c)}\,(\forall\mathbf{I})} \qquad \dfrac{\begin{array}{c}\vdots X\\ \forall c.\mathsf{P}(c)\end{array} \quad (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B}}{\mathsf{B}}\,(\Rightarrow\mathbf{E}) \tag{2}$$

are represented, writing Γ for $a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B},\ c : *$, by:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash a{:}\mathsf{A}}\,(\mathbf{Ta}\phi) \quad \dfrac{\dfrac{}{\Gamma \vdash p : \forall c.(\mathsf{A}{\Rightarrow}\mathsf{P}(c))}\,(\mathbf{Ta}\phi)}{\Gamma \vdash pc : \mathsf{A}{\Rightarrow}\mathsf{P}(c)}\,(\mathbf{T}\forall\mathbf{E})}{\Gamma \vdash pca : \mathsf{P}(c)}\,(\mathbf{T}{\Rightarrow}\mathbf{E}) \quad \begin{array}{l}(c{\notin}fa(\mathsf{A}),\\ \ c{\notin}fa(\forall c.(\mathsf{A}{\Rightarrow}\mathsf{P}(c))))\end{array}}{\Gamma \vdash \lambda c.(pca) : \forall c.\mathsf{P}(c)}\,(\mathbf{T}\forall\mathbf{I}) \quad \dfrac{}{\Gamma \vdash q : (\forall c.\mathsf{P}(c)){\Rightarrow}\mathsf{B}}\,(\mathbf{Ta}\phi)}{\Gamma \vdash q(\lambda c.(pca)) : \mathsf{B}}\,(\mathbf{T}{\Rightarrow}\mathbf{E})}{a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} \vdash q(\lambda c.(pca)) : \mathsf{B}}\,(\mathbf{Tfr})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,\ X : \mathsf{P}(c) \vdash X : \mathsf{P}(c)}\,(\mathbf{TX}) \quad \begin{array}{l}(c \notin fa(\mathsf{A})\\ \ c \notin fa(\forall c.(\mathsf{A}{\Rightarrow}\mathsf{P}(c)))\\ \ c \notin fa((\forall c.\mathsf{P}(c)){\Rightarrow}\mathsf{B}))\end{array}}{\Gamma,\ X : \mathsf{P}(c) \vdash \lambda c.X : \forall c.\mathsf{P}(c)}\,(\mathbf{T}\forall\mathbf{I}) \quad \dfrac{}{\Gamma,\ X : \mathsf{P}(c) \vdash q : (\forall c.\mathsf{P}(c)){\Rightarrow}\mathsf{B}}\,(\mathbf{Ta}\phi)}{\Gamma,\ X : \mathsf{P}(c) \vdash q(\lambda c.X) : \mathsf{B}}\,(\mathbf{T}{\Rightarrow}\mathbf{E})}{a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B},\ X : \mathsf{P}(c) \vdash q(\lambda c.X) : \mathsf{B}}\,(\mathbf{Tfr})$$

Derivations of freshnesses are elided.

$$\frac{\Gamma;\Delta \vdash r:\phi \qquad (Y \notin \Gamma)}{\Gamma,\ Y:\psi;\Delta \vdash r:\phi}\ (\mathbf{WeakX})$$

$$\frac{\Gamma;\Delta \vdash r:\phi \qquad (B \in \{b:\psi,\ b:*\},\ b \notin \Gamma)}{\Gamma,\ B;\Delta, b\#unkn(\Gamma) \vdash r:\phi}\ (\mathbf{Weaka})$$

$$\frac{\begin{array}{c}\Gamma, Y{:}\psi;\Delta, a_1\#Y,\ldots,a_n\#Y \vdash r:\phi \quad \Gamma;\Delta \vdash u{:}\psi \\ \Gamma;\Delta \vdash a_i\#u\ (1{\leq}i{\leq}n) \quad (Y{\notin}\Delta)\end{array}}{\Gamma;\Delta \vdash r[Y := u]:\phi}\ (\mathbf{Cut})$$

Figure 3: Admissible rules

3. Admissible rules

Definition 28. Say that a derivation rule:

$$\frac{A_1 \quad \ldots \quad A_n}{B}$$

is **admissible** if $A_1 \ldots A_n$ is derivable implies that B is derivable.

Definition 29. Write $unkn(\Gamma)$ for the variables of level two mentioned in Γ.

Definition 30. Figure 3 presents three admissible rules: two kinds of weakening, (**WeakX**) and (**Weaka**), and a form of Cut.

We briefly outline the meaning of these rules, notation and definitions follow:

- Weakening for variables of level two (**WeakX**) is the technically easiest result (Theorem 32). This is the usual weakening result for natural deduction. Intuitively, we introduce a new *unknown* derivation, represented by a variable of level two (but never use it).
- Weakening for variables of level one (**Weaka**) is technically harder (Theorem 33); what is it to weaken a derivation with an extra assumption (represented by a variable of level one) that is never used — if we do not yet know all parts of that derivation? We use freshness conditions.
- Cut is technically the most challenging, and intuitively the most significant result (Theorem 36). Here, Cut means intuitively that instantiating unknown parts of a derivation results in another derivation; in other words, it is sound to instantiate variables of level two.

3.1. *Weaken with a variable of level two* (**WeakX**)

Lemma 31. *Suppose* Γ' *satisfies Definition 6. Suppose also that* $\Gamma \subseteq \Gamma'$ *and* $\Delta \subseteq \Delta'$. *If* $\Gamma;\Delta \vdash a\#r$ *then* $\Gamma';\Delta' \vdash a\#r$.

Proof. By induction on the derivation of $\Gamma;\Delta \vdash a\#r$.

- The case $(\mathbf{a\#b})$. Suppose $a : \phi \in \Gamma$ and $b : \phi \in \Gamma$. Then $a : \phi \in \Gamma'$ and $b : \phi \in \Gamma'$. Applying $(\mathbf{a\#b})$, we obtain $\Gamma'; \Delta' \vdash a\#b$. The result follows.
- The case $(\mathbf{a\#b'})$. Suppose $a : * \in \Gamma$ and $b : \phi \in \Gamma$, with $a \notin fa(\phi)$. Then $a : * \in \Gamma'$ and $b : \phi \in \Gamma'$. Applying $(\mathbf{a\#b'})$, we obtain $\Gamma'; \Delta' \vdash a\#b$. The result follows.
- The case $(\mathbf{a\#g})$ is immediate.
- The case $(\mathbf{a\#X})$. Suppose $\pi^{-1}(a)\#X \in \Delta$, therefore $\pi^{-1}(a)\#X \in \Delta'$. Applying $(\mathbf{a\#X})$, we obtain $\Gamma'; \Delta' \vdash a\#\pi \cdot X$. The result follows.
- The case $(\mathbf{a\#\lambda a})$. Since $\Gamma'; \Delta' \vdash a\#\lambda a.r$ always, by $(\mathbf{a\#\lambda a})$.
- The case $(\mathbf{a\#\lambda b})$. By inductive hypothesis, $\Gamma'; \Delta' \vdash a\#r$ and $\Gamma'; \Delta' \vdash a\#b$. Applying $(\mathbf{a\#\lambda b})$, we obtain $\Gamma'; \Delta' \vdash a\#\lambda b.r$. The result follows.
- The case $(\mathbf{a\#app})$. By inductive hypothesis, $\Gamma'; \Delta' \vdash a\#r'$ and $\Gamma'; \Delta' \vdash a\#r$. Applying $(\mathbf{a\#app})$, we obtain $\Gamma'; \Delta' \vdash a\#r'r$. The result follows.
- The case $(\mathbf{a\#}\mathsf{xf})$. By inductive hypothesis, $\Gamma'; \Delta' \vdash a\#r$. Applying $(\mathbf{a\#}\mathsf{xf})$, we obtain $\Gamma'; \Delta' \vdash a\#\mathsf{xf}(r)$. The result follows.

□

Theorem 32. $(\mathbf{WeakX})$ *is admissible (Figure 3).*

Proof. By induction on derivations.

- The case $(\mathbf{Ta}*)$. There is nothing to prove.
- The case $(\mathbf{Ta}\phi)$. By assumption, $\Gamma; \Delta \vdash a : \phi$ therefore $a : \phi \in \Gamma$, hence $a : \phi \in \Gamma, Y : \psi$. The result follows.
- The case $(\mathbf{TX})$. By assumption, $\Gamma; \Delta \vdash X : \phi$ therefore $X : \phi \in \Gamma$, hence $X : \phi \in \Gamma, Y : \psi$. Note that $\Gamma \vdash \pi$ by assumption. The result follows.
- The case $(\mathbf{T}\bot\mathbf{E})$. By inductive hypothesis, $\Gamma, Y : \psi; \Delta \vdash r : \bot$ where $Y \notin \Gamma$. Applying $(\mathbf{T}\bot\mathbf{E})$, we have $\Gamma, Y : \psi; \Delta \vdash \mathsf{xf}(r) : \phi$. The result follows.
- The case $(\mathbf{T}\Rightarrow\mathbf{I})$. By inductive hypothesis, $\Gamma, a : \phi, Z : \xi; \Delta \vdash r : \psi$. Applying $(\mathbf{T}\Rightarrow\mathbf{I})$, we obtain $\Gamma, Z : \xi; \Delta \vdash \lambda a.r : \phi \Rightarrow \psi$. The result follows.
- The case $(\mathbf{T}\Rightarrow\mathbf{E})$. By inductive hypothesis, $\Gamma, Z : \xi; \Delta \vdash r' : \phi \Rightarrow \psi$ and $\Gamma, Z : \xi; \Delta \vdash r : \phi$ with $Z \notin \Gamma$. Applying $(\mathbf{T}\Rightarrow\mathbf{E})$, we obtain $\Gamma, Z : \xi; \Delta \vdash r'r : \psi$. The result follows.
- The case $(\mathbf{T}\forall\mathbf{I})$. By inductive hypothesis, $\Gamma, a : *, Y : \psi; \Delta \vdash r : \phi$ with $Y \notin \Gamma$ where it is easy to see $a \notin fa(important(\Gamma, a : *, Y : \psi; \Delta \vdash r))$ when $a \notin fa(important(\Gamma, a : *; \Delta \vdash r))$ and $Y \notin \Gamma$. Applying $(\mathbf{T}\forall\mathbf{I})$, we obtain $\Gamma, a : *, Y : \psi; \Delta \vdash \lambda a.r : \forall a.\phi$. The result follows.
- The case $(\mathbf{T}\forall\mathbf{E})$. By inductive hypothesis, $\Gamma, Y : \psi; \Delta \vdash r : \forall a.\phi$ with $Y \notin \Gamma$. Applying $(\mathbf{T}\forall\mathbf{E})$, we obtain $\Gamma, Y : \psi; \Delta \vdash rg : \phi[a := g]$. The result follows.

- The case (**Tfr**). By inductive hypothesis, $\Gamma, b : \tau, Z : \xi; \Delta, b\#\mathcal{X} \vdash r : \phi$ where $Z \notin \Gamma$, $\tau \in \{\psi, *\}$ and $b \notin \Delta$. By Lemma 31, we have $\Gamma, b : \tau, Z : \xi; \Delta, b\#\mathcal{X} \vdash b\#r$. Applying (**Tfr**), we obtain $\Gamma, Z : \xi; \Delta \vdash r : \phi$. The result follows.

□

3.2. Weaken with a variable of level one (**Weaka**)

(**Weaka**) *states* that the variable of level one is fresh for the incomplete parts in the derivation:

Theorem 33. (**Weaka**) *is an admissible rule (Figure 3).*

Proof. By induction on derivations.

- The case (**Ta**$*$). There is nothing to prove.
- The case (**Ta**ϕ). Suppose $\Gamma; \Delta \vdash a : \phi$ then $a : \phi \in \Gamma$. Further, suppose $b \notin \Gamma$ and $b : \tau$ for $\tau \in \{\psi, *\}$. Then $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash a : \phi$ is derivable. The result follows.
- The case (**TX**). Suppose $\Gamma; \Delta \vdash X : \phi$ then $X : \phi \in \Gamma$. Further, suppose $b \notin \Gamma$ and $b : \tau$ for $\tau \in \{\psi, *\}$. Then $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash X : \phi$ is derivable, and by assumption $\Gamma \vdash \pi$. The result follows.
- The case (**T**$\perp$**E**). By inductive hypothesis, $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash r : \perp$ for $\tau \in \{\phi, *\}$ and $b \notin \Gamma$. Applying (**T**$\perp$**E**), we obtain $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash \mathsf{xf}(r) : \phi$. The result follows.
- The case (**T**$\Rightarrow$**I**). By inductive hypothesis, $\Gamma, a : \phi, b : \tau; \Delta, unkn(\Gamma) \vdash r : \phi$ for $\tau \in \{\psi, *\}$ and $b \notin \Gamma$. Applying (**T**$\Rightarrow$**I**), we obtain $\Gamma, a : \phi, b : \tau; \Delta, b\#unkn(\Gamma) \vdash \lambda a.r : \phi \Rightarrow \psi$. The result follows.
- The case (**T**$\Rightarrow$**E**). By inductive hypothesis, $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash r' : \phi \Rightarrow \psi$ and $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash r : \phi$ for $\tau \in \{\phi, *\}$ and $b \notin \Gamma$. Applying (**T**$\Rightarrow$**E**), we obtain $\Gamma, b : \tau; \Delta, b\#unkn(\Gamma) \vdash r'r : \psi$ and the result follows.
- The case (**T**$\forall$**I**). By inductive hypothesis $\Gamma, a : *, b : \tau; \Delta, b\#unkn(\Gamma) \vdash r : \phi$ where $b \notin \Gamma, a : *$ and $\tau \in \{\psi, *\}$. It is not hard to calculate that $\Gamma, a : *, b : \tau; \Delta, b\#unkn(\Gamma) \vdash b\#r$ and so $a \notin fa(important(\Gamma, a : *, b : \tau; \Delta, b\#unkn(\Gamma) \vdash r))$. The result follows.
- The case (**T**$\forall$**E**). By inductive hypothesis, $\Gamma, b : *; \Delta, b\#unkn(\Gamma) \vdash r : \forall a.\phi$ where $b \notin \Gamma$. Applying (**T**$\forall$**E**), we obtain $\Gamma; \Delta, b\#unkn(\Gamma) \vdash rg : \phi[a := g]$. The result follows.
- The case (**Tfr**). By inductive hypothesis, $\Gamma, a : \tau, b : \tau'; \Delta, a\#\mathcal{X}, b\#unkn(\Gamma, a : \tau) \vdash r : \phi$ where $\tau, \tau' \in \{\phi, *\}$ and $b \notin \Gamma$. By Lemmas 31, $\Gamma, a : \tau, b : \tau'; \Delta, a\#\mathcal{X}, b\#unkn(\Gamma, a : \tau) \vdash b\#r$. Applying (**Tfr**) it is easy to see that $unkn(\Gamma) = unkn(\Gamma, a : \tau)$ and we therefore obtain $\Gamma, b : \tau'; \Delta, b\#unkn(\Gamma) \vdash r : \phi$. The result follows.

□

3.3. Cut

The main result of this subsection is Theorem 36, that (**Cut**) from Figure 3 is admissible. The proof is non-trivial, and as discussed at the start of this Section, it is significant: it states that level two variables may be instantiated, in other words, that the 'holes' in our incomplete derivations really can be filled with derivations.

The proof of Theorem 36 depends on two lemmas: Lemma 34 and Lemma 35. Recall that $important(\Gamma; \Delta \vdash r : \phi)$ (Definition 14) represents "the possible assumptions of instantiations of r". Lemma 34 captures this, by proving that the instantiations of r *do* have assumptions contained in $important(\Gamma; \Delta \vdash r : \phi)$; this is important to make sure that it is sound to instantiate level two variables in (**T**$\forall$**I**). Lemma 35 is the meat of the proof of Theorem 36. It states a precise sense in which instantiating level two variables is sound.

Lemma 34. *Suppose that:*

- $\Gamma, Y : \psi; \Delta \vdash r : \phi$ *and* $\Gamma; \Delta \vdash u : \psi$.
- $\Gamma; \Delta' \vdash a\#u$ *for every* $a\#Y \in \Delta$.

  (Here, Δ' *is* any *freshness context satisfying the above condition.)*

Then

$$important(\Gamma; \Delta' \vdash r[Y := u]) \subseteq important(\Gamma, Y : \psi; \Delta \vdash r).$$

Proof. By a routine induction on r. We consider cases:

- The case a. Note $a[Y := u] \equiv a$. Then $important(\Gamma; \Delta' \vdash a) \subseteq important(\Gamma, Y : \psi; \Delta \vdash a)$. The result follows.
- The case $\pi \cdot Y$. We have $important(\Gamma; \Delta' \vdash \pi \cdot u) = \{\phi \mid a : \phi \in \Gamma, \Gamma; \Delta' \not\vdash a\#\pi \cdot u\}$. This is equivalent to $\{\phi \mid a : \phi \in \Gamma, \Gamma; \Delta' \not\vdash \pi^{-1}(a)\#u\}$.

  Further, we have $important(\Gamma, Y : \psi; \Delta \vdash \pi \cdot Y) = \{\phi \mid a : \phi \in \Gamma, \Gamma, Y : \psi; \Delta \not\vdash a\#\pi \cdot Y\}$. This, in turn, is equivalent to $\{\phi \mid a : \phi \in \Gamma, \pi^{-1}(a)\#Y \notin \Delta\}$.

  Now, we must show $\{\phi \mid a : \phi \in \Gamma, \Gamma; \Delta' \not\vdash \pi^{-1}(a)\#u\} \subseteq \{\phi \mid a : \phi \in \Gamma, \pi^{-1}(a)\#Y \notin \Delta\}$. So, suppose $\phi \in \{\phi \mid a : \phi \in \Gamma, \Gamma; \Delta' \not\vdash \pi^{-1}(a)\#u\}$, and therefore ϕ is the type of some variable of level one in Γ such that $\Gamma; \Delta' \not\vdash \pi^{-1}(a)\#u$. By assumption, $\pi^{-1}(a)\#Y \in \Delta$ implies $\Gamma; \Delta' \vdash \pi^{-1}(a)\#u$, therefore $\Gamma; \Delta' \not\vdash \pi^{-1}(a)\#u$ implies $\pi^{-1}(a)\#Y \notin \Delta$. The result follows.
- The case $\pi \cdot X$. Note that $(\pi \cdot X)[Y := s] \equiv \pi \cdot X$. Further, note $\pi^{-1}(a)\#X \in \Delta'$. It is then easy to see that $important(\Gamma; \Delta' \vdash \pi \cdot X) \subseteq important(\Gamma, Y : \psi; \Delta \vdash \pi \cdot X)$. The result follows.
- The case $\lambda b.r$. By inductive hypothesis, $important(\Gamma; \Delta' \vdash r[Y := u]) \subseteq important(\Gamma, Y : \psi; \Delta \vdash r)$. Note that $(\lambda b.r)[Y := u] \equiv \lambda b.(r[Y := u])$. The result follows.
- The case $r'r$. Note that $(r'r)[Y := u] \equiv r'[Y := u](r[Y := u])$. The result follows from the inductive hypothesis.
- The case $\mathsf{f}(r)$. Note that $\mathsf{f}(r)[Y := u] \equiv \mathsf{f}(r[Y := u])$. The result follows from the inductive hypothesis.

□

Lemma 35. *Suppose that:*

- $\Gamma, Y : \psi; \Delta \vdash r : \phi$ *and* $\Gamma; \Delta \vdash u : \psi$.
- $\Gamma; \Delta' \vdash a\#u$ *for every* $a\#Y \in \Delta$.

Then:

- $\Gamma, Y : \psi; \Delta \vdash a\#r$ *implies* $\Gamma; \Delta' \vdash a\#r[Y := u]$, *for every* a.
- $\Gamma; \Delta' \vdash r[Y := u] : \phi$.

Proof. The first claim is shown by induction on the derivation of $\Gamma; \Delta' \vdash a\#t$.

- The case $(\mathbf{a\#b})$. Suppose $\Gamma, a : \phi, b : \phi, Y : \psi; \Delta \vdash a\#b$ is derived using $(\mathbf{a\#b})$. Then $\Gamma, a : \phi, b : \phi; \Delta' \vdash a\#b$ and $b[Y := u] \equiv b$. The result follows.
- The case $(\mathbf{a\#b'})$. Suppose $\Gamma, a : *, b : \phi, Y : \psi; \Delta \vdash a\#b$ is derived by $(\mathbf{a\#b'})$. Then $a \notin fa(\phi)$ and it follows that $\Gamma, a : *, b : \phi; \Delta' \vdash a\#b$ and $b[Y := u] \equiv b$. The result follows.
- The case $(\mathbf{a\#g})$. Suppose $\Gamma, Y : \psi; \Delta \vdash a\#g$ is derived using $(\mathbf{a\#g})$. Then $\Gamma; \Delta' \vdash a\#g$ and $g[Y := u] \equiv g$. The result follows.
- The case $(\mathbf{a\#X})$. There are two cases:
  - The case $\pi \cdot Y$. Suppose $\Gamma, Y : \psi; \Delta \vdash a\#\pi \cdot Y$ is derived using $(\mathbf{a\#X})$. This implies $\pi^{-1}(a)\#Y \in \Delta$ by inversion. By assumption, $\Gamma; \Delta' \vdash \pi^{-1}(a)\#u$. From Lemma 19 and Lemma 18 we have $\Gamma; \Delta' \vdash a\#\pi \cdot u$. The result follows.
  - The case $\pi \cdot X$. Suppose $\Gamma, X : \phi; \Delta \vdash a\#\pi \cdot X$ is derived using $(\mathbf{a\#X})$. This implies $\pi^{-1}(a)\#X \in \Delta$. By assumption, $\Gamma; \Delta' \vdash \pi^{-1}(a)\#u$. By Lemma 19, Lemma 18, and the fact that $(\pi \cdot X)[Y := u] \equiv \pi \cdot X$, the result follows.
- The case $(\mathbf{a\#\lambda a})$. Note $\Gamma; \Delta' \vdash a\#\lambda a.(r[Y := u])$ always, using $(\mathbf{a\#\lambda a})$. Since $\lambda a.(r[Y := u]) \equiv (\lambda a.r)[Y := u]$, the result follows.
- The case $(\mathbf{a\#\lambda b})$. Suppose $\Gamma, Y : \psi; \Delta \vdash a\#r$ and $\Gamma, Y : \psi; \Delta \vdash a\#b$ and $\Gamma, Y : \psi; \Delta \vdash a\#\lambda b.r$ is derived by $(\mathbf{a\#\lambda b})$. By inductive hypothesis $\Gamma; \Delta' \vdash a\#r[Y := u]$ and $\Gamma; \Delta' \vdash a\#b$ and so $\Gamma; \Delta' \vdash a\#\lambda b.(r[X := t])$. Since $\lambda b.(r[Y := u]) \equiv (\lambda b.r)[Y := u]$, the result follows.
- The case $(\mathbf{a\#app})$. By inductive hypothesis, $\Gamma; \Delta' \vdash a\#r'[Y := u]$ and $\Gamma; \Delta' \vdash a\#r[Y := u]$. Applying $(\mathbf{a\#app})$, we obtain $\Gamma; \Delta' \vdash a\#r'[Y := u](r[Y := u])$. Note that $r'[Y := u](r[Y := u]) \equiv (r'r)[Y := u]$. The result follows.
- The case $(\mathbf{a\#xf})$. By inductive hypothesis, $\Gamma; \Delta' \vdash a\#r[Y := u]$. Applying $(\mathbf{a\#xf})$, we obtain $\Gamma; \Delta' \vdash a\#\mathsf{xf}(r[Y := u])$. Note that $\mathsf{xf}(r[Y := u]) \equiv \mathsf{xf}(r)[Y := u]$. The result follows.

The second claim is shown by induction on the derivation of $\Gamma, Y : \psi; \Delta \vdash r : \phi$.

- The case ($\mathbf{Ta}*$). There is nothing to prove.
- The case ($\mathbf{Ta}\phi$). Suppose $a : \phi \in \Gamma$ and $\Gamma, Y : \psi; \Delta \vdash a : \phi$ is derived by ($\mathbf{Ta}\phi$). Then $\Gamma; \Delta' \vdash a : \phi$. Since $a \equiv a[Y := u]$, the result follows..
- The case ($\mathbf{TX}$). Suppose $X : \phi \in \Gamma$ and $\Gamma, Y : \psi; \Delta \vdash X : \phi$ is derived by ($\mathbf{TX}$). Then $\Gamma; \Delta' \vdash X : \phi$. Since $X \equiv X[Y := u]$, the result follows. Further, suppose $\Gamma, Y : \psi; \Delta \vdash Y : \psi$ is derived by ($\mathbf{TX}$). By assumption $\Gamma; \Delta' \vdash u : \psi$. Since $u \equiv Y[Y := u]$, the result follows.
- The case ($\mathbf{T}\bot\mathbf{E}$). By inductive hypothesis, $\Gamma; \Delta' \vdash r[X := t] : \bot$. Applying ($\mathbf{T}\bot\mathbf{E}$), we obtain $\Gamma; \Delta' \vdash \mathsf{xf}(r[X := t]) : \phi$. The result follows.
- The case ($\mathbf{T}\Rightarrow\mathbf{I}$). By inductive hypothesis, $\Gamma, a : \phi; \Delta' \vdash r[X := t] : \psi$. Applying ($\mathbf{T}\Rightarrow\mathbf{I}$), we obtain $\Gamma, a : \phi; \Delta' \vdash \lambda a.(r[X := t]) : \phi \Rightarrow \psi$. As $\lambda a.(r[X := t]) \equiv (\lambda a.r)[X := t]$, the result follows.
- The case ($\mathbf{T}\Rightarrow\mathbf{E}$). By inductive hypothesis, $\Gamma; \Delta' \vdash r'[X := t] : \phi \Rightarrow \psi$ and $\Gamma; \Delta' \vdash r[X := t] : \phi$. Applying ($\mathbf{T}\Rightarrow\mathbf{E}$), we obtain $\Gamma; \Delta' \vdash (r'[X := t])(r[X := t]) : \psi$. As $(r'[X := t])(r[X := t]) \equiv (r'r)[X := t]$, the result follows.
- The case ($\mathbf{T}\forall\mathbf{E}$). By inductive hypothesis, $\Gamma; \Delta' \vdash r[X := t] : \forall a.\phi$. Applying ($\mathbf{T}\forall\mathbf{E}$), we obtain $\Gamma; \Delta' \vdash r[X := t]g : \phi[a := g]$. As $r[X := t]g \equiv (rg)[X := t]$, the result follows.
- The case ($\mathbf{T}\forall\mathbf{I}$). By inductive hypothesis, $\Gamma, a : *; \Delta' \vdash r[X := t] : \phi$. Using Lemma 34, $a \notin fa(\mathit{important}(\Gamma, a : *; \Delta' \vdash r[X := t]))$. Applying ($\mathbf{T}\forall\mathbf{I}$), we have $\Gamma, a : *; \Delta' \vdash \lambda a.(r[X := t]) : \forall a.\phi$. As $\lambda a.(r[X := t]) \equiv (\lambda a.r)[X := t]$, the result follows.
- The case ($\mathbf{Tfr}$). Suppose $\Gamma,\ Z : \xi; \Delta \vdash r : \phi$ because $\Gamma,\ b : \tau,\ Z : \xi;\ \Delta,\ b\#\mathcal{X} \vdash r : \phi$ and $\Gamma; \Delta \vdash v : \xi$ and suppose that $\Gamma,\ b : \tau,\ Z : \xi;\ \Delta,\ b\#\mathcal{X} \vdash b\#r$ where $b \notin \Delta$ and $\tau \in \{\xi, *\}$. By inductive hypothesis and some calculations, using the first claim, $\Gamma,\ b : \tau;\ \Delta',\ b\#\mathcal{X}' \vdash r[Z := v] : \phi$ and $\Gamma,\ b : \tau;\ \Delta',\ b\#\mathcal{X}' \vdash b\#r[Z := v]$ for a suitable $\mathcal{X}'$ and Δ' where $b \notin \Delta'$. The result follows.

□

Cut in natural deduction is the operation of 'plugging the conclusion of one derivation into the assumption(s) of another'. Yet, for us now, these derivations may be incomplete and substitution capture-avoiding. The rule ($\mathbf{Cut}$) specifies precisely what operation this now is, and we have:

Theorem 36. ($\mathbf{Cut}$) *is an admissible rule.*

Proof. Suppose, given $Y \notin \Delta$, that $\Gamma, Y : \psi; \Delta, a_1\#Y \ldots a_n\#Y \vdash r : \phi$, $\Gamma; \Delta \vdash u : \psi$ and $\Gamma; \Delta \vdash a_i\#u$ for $1 \leq i \leq n$. Then clearly $\Gamma; \Delta \vdash a_i\#u$ for every $a_i\#Y \in \Delta, a_1\#Y, \ldots a_n\#Y$. The result now follows immediately from Lemma 35. □

4. Natural deduction

The intended semantics of a term r is 'an incomplete natural deduction derivation'. In this section, we recap the standard theory of natural deduction derivations and prove forms of soundness (Theorem 41) and completeness (Theorem 42).

Definition 37. Call a finite set of types a (natural deduction) **context**. Let Φ, Φ' range over contexts.

Write $\Phi \vdash \phi$ when ϕ may be derived using the rules in Figure 2 allowing elements of Φ as assumptions.[8] In accordance with our convention, side-conditions are in brackets. As is standard, square brackets in $(\Rightarrow\mathbf{I})$ denote *discharge* of assumptions; note that we may choose to discharge ϕ zero times (*empty* discharge).

Lemma 38. *If $\Phi \vdash \psi$ and $\Phi \subseteq \Phi'$ then $\Phi' \vdash \psi$.*

Proof. By induction on derivations. The special case where $\Phi, \phi \vdash \phi$ follows immediately. See Footnote 8 for details.

- The case $(\bot\mathbf{E})$. By inductive hypothesis, $\Phi' \vdash \bot$. Applying $(\bot\mathbf{E})$, we obtain $\Phi' \vdash \phi$ and the result follows.
- The case $(\Rightarrow\mathbf{I})$. By inductive hypothesis, $\Phi', \phi \vdash \psi$. Applying $(\Rightarrow\mathbf{I})$, we obtain $\Phi' \vdash \phi \Rightarrow \psi$ when we discharge, and the result follows. Alternatively, we obtain $\Phi', \phi \vdash \phi \Rightarrow \psi$ where we discharge zero times, and the result follows.
- The case $(\Rightarrow\mathbf{E})$. By inductive hypothesis, $\Phi' \vdash \phi \Rightarrow \psi$ and $\Phi' \vdash \phi$. Applying $(\Rightarrow\mathbf{E})$, we obtain $\Phi' \vdash \psi$ and the result follows.
- The case $(\forall\mathbf{I})$. By inductive hypothesis, $\Phi' \vdash \phi$. Picking suitable a we extend with $(\forall\mathbf{I})$, to obtain $\Phi' \vdash \forall a.\phi$ and the result follows.
- The case $(\forall\mathbf{E})$. By inductive hypothesis, $\Phi' \vdash \forall b.\phi$. Applying $(\forall\mathbf{E})$, we obtain $\Phi' \vdash \phi$ and the result follows.

□

Definition 39. Call $\Gamma; \Delta \vdash r$ **closed** when $\Delta = \varnothing$ and Γ mentions no variables of level two. Recall that we write '$\Gamma; \varnothing \vdash r$' as '$\Gamma \vdash r$'.

We will use the following fact without comment:

Lemma 40. *If $\Gamma \vdash r$ is closed (so Γ mentions no variables of level two and the freshness context is empty) and $\Gamma \vdash r$ is typable, then r mentions no variables of level two.*

Proof. By induction on derivations.

- The case $(\mathbf{Ta}*)$. $r \equiv a : *$. There is nothing to prove since $*$ is not a type.
- The case $(\mathbf{Ta}\phi)$. $r \equiv a : \phi$. The result is immediate.

[8] Note that in natural deduction, $\Phi, \phi \vdash \phi$ is automatic — 'if we allow Φ, ϕ as assumptions, then we assume ϕ, and so we have ϕ'. There is no need for a derivation rule.

- The case ($\mathbf{TX}$). Clearly $r \equiv X : \phi$, hence $\Gamma \vdash r$ cannot be closed. There is nothing to prove.
- The case ($\mathbf{T}\bot\mathbf{E}$). By inductive hypothesis, $\Gamma \vdash r : \bot$ where Γ is closed and r contains no variables of level two. Applying ($\mathbf{T}\bot\mathbf{E}$), we have $\Gamma \vdash \mathsf{xf}(r) : \phi$. The result follows.
- The case ($\mathbf{T}\Rightarrow\mathbf{I}$). By inductive hypothesis, $\Gamma, a : \phi \vdash r : \psi$ with Γ closed and r containing no variables of level two. Applying ($\mathbf{T}\Rightarrow\mathbf{I}$), we obtain $\Gamma \vdash \lambda a.r : \phi \Rightarrow \psi$. The result follows.
- The case ($\mathbf{T}\Rightarrow\mathbf{E}$). By inductive hypothesis, $\Gamma \vdash r' : \phi \Rightarrow \psi$ and $\Gamma \vdash r : \phi$ with Γ closed and r' and r containing no variables of level two. Applying ($\mathbf{T}\Rightarrow\mathbf{E}$), we obtain $\Gamma \vdash r'r : \psi$. The result follows.
- The case ($\mathbf{T}\forall\mathbf{I}$). By inductive hypothesis, $\Gamma, a : * \vdash r : \phi$ with Γ closed and r containing no variables of level two, where a satisfies the side condition. Applying ($\mathbf{T}\forall\mathbf{I}$), we obtain $\Gamma \vdash \lambda a.r : \forall a.\phi$. The result follows.
- The case ($\mathbf{T}\forall\mathbf{E}$). By inductive hypothesis, we have $\Gamma \vdash r : \forall a.\phi$ with Γ closed and r containing no level two variables. Applying ($\mathbf{T}\forall\mathbf{E}$), we obtain $\Gamma \vdash rg : \phi[a := g]$ and the result follows.
- The case ($\mathbf{Tfr}$). By inductive hypothesis, $\Gamma, b : \tau \vdash r : \phi$ and $\Gamma, b : \tau \vdash b\#r$ with r containing no variables of level two and Γ closed, where $\tau \in \{\phi, *\}$. Applying ($\mathbf{Tfr}$), we obtain $\Gamma \vdash r : \phi$. The result follows.

□

Theorem 41. *Suppose $\Gamma \vdash r$ is closed and suppose $\Gamma \vdash r : \phi$ is derivable. Then $important(\Gamma \vdash r) \vdash \phi$ (Definition 14) is derivable in natural deduction.*

Proof. By induction on the derivation of $\Gamma \vdash r : \phi$.

- The case ($\mathbf{Ta}\phi$). Suppose $\Gamma, a : \phi \vdash a : \phi$. It is then easy to calculate that $important(\Gamma, a : \phi \vdash a) = \{\phi\}$ and $\phi \vdash \phi$ is a fact.
- The case ($\mathbf{TX}$). As $\Gamma \vdash r$ is not closed, there is nothing to prove.
- The case ($\mathbf{T}\bot\mathbf{E}$). Suppose $\Gamma; \Delta \vdash r : \bot$. By inductive hypothesis, $important(\Gamma; \Delta \vdash r) \vdash \bot$. Note $important(\Gamma; \Delta \vdash r) = important(\Gamma; \Delta \vdash \mathsf{xf}(r))$. Applying ($\bot\mathbf{E}$), we have $important(\Gamma; \Delta \vdash \mathsf{xf}(r)) \vdash \phi$. The result follows.
- The case ($\mathbf{T}\Rightarrow\mathbf{I}$). Suppose $\Gamma, a : \phi \vdash \lambda a.r : \phi \Rightarrow \psi$ and $\Gamma, a : \phi \vdash r : \psi$. By inductive hypothesis, $important(\Gamma, a : \phi \vdash r) \vdash \psi$. We now apply ($\Rightarrow\mathbf{I}$). If $important(\Gamma, a : \phi \vdash \lambda a.r) = important(\Gamma, a : \phi \vdash r) \setminus \{\phi\}$ then we discharge ϕ. Otherwise $important(\Gamma, a : \phi \vdash \lambda a.r) = important(\Gamma, a : \phi \vdash r)$ and we discharge ϕ zero times. The result follows.

- The case of ($\mathbf{T}\Rightarrow\mathbf{E}$). Suppose $\Gamma \vdash r'r : \psi$ and $\Gamma \vdash r' : \phi \Rightarrow \psi$ and $\Gamma \vdash r : \phi$. By inductive hypothesis $important(\Gamma \vdash r') \vdash \phi \Rightarrow \psi$ and $important(\Gamma \vdash r) \vdash \phi$. By Lemma 38 and ($\mathbf{T}\Rightarrow\mathbf{E}$), $important(\Gamma \vdash r') \cup important(\Gamma \vdash r) \vdash \psi$. By the syntax-directed nature of the freshness rules in Figure 1, $\Gamma \vdash a\#r'r$ if and only if both of $\Gamma \vdash a\#r'$ and $\Gamma \vdash a\#r$ hold. Therefore, $important(\Gamma \vdash r'r) = important(\Gamma \vdash r') \cup important(\Gamma \vdash r)$. The result follows.
- The case ($\mathbf{T}\forall\mathbf{I}$). Suppose $\Gamma, a : * \vdash \lambda a.r : \forall a.\phi$ where $\Gamma, a : * \vdash r : \phi$ and $a \notin fa(important(\Gamma, a : * \vdash r))$. By inductive hypothesis $important(\Gamma, a : * \vdash r) \vdash \phi$. Applying ($\forall\mathbf{I}$), $important(\Gamma, a : * \vdash r) \vdash \forall a.\phi$. The result follows.
- The case ($\mathbf{T}\forall\mathbf{E}$). Suppose $\Gamma \vdash rg : \phi[b := g]$ and $\Gamma \vdash r : \forall b.\phi$. By inductive hypothesis $important(\Gamma \vdash r) \vdash \forall b.\phi$. Applying ($\forall\mathbf{E}$), $important(\Gamma \vdash r) \vdash \phi[b := g]$. By reasoning similar to the case of ($\mathbf{T}\Rightarrow\mathbf{E}$) we can calculate that $important(\Gamma \vdash rg) = important(\Gamma \vdash r)$. The result follows.
- The case ($\mathbf{Tfr}$). Suppose $\Gamma \vdash r : \phi$ and $\Gamma, b : \tau \vdash r : \phi$ and $\Gamma, b : \tau \vdash b\#r$ where $\tau \in \{\phi, *\}$. By inductive hypothesis $important(\Gamma, b : \tau \vdash r) \vdash \phi$. If $\tau = *$ then $important(\Gamma, b : * \vdash r) = important(\Gamma \vdash r)$ and the result follows immediately. If $\tau = \phi$ then, since $\Gamma, b : \phi \vdash b\#r$ again $important(\Gamma, b : \phi \vdash r) = important(\Gamma \vdash r)$. The result follows.

□

Theorem 42. *If $\Phi \vdash \phi$ is derivable in natural deduction then there exists some closed $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$.*

Proof. We prove by induction on the derivation of $\Phi \vdash \phi$ that there exists some closed typable $\Gamma \vdash r$ such that:

- $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$.
- Γ satisfies a *uniqueness* property: if $a : \phi \in \Gamma$ and $x : \phi \in \Gamma$ then $x = a$ (so there is at most one variable of level one of each type in Γ).[9]

We consider each possible rule in turn:

- The case of no rule; $\Phi \vdash \phi$ because $\phi \in \Phi$. Suppose $fa(\phi) = \{b_1, \ldots, b_n\}$. We take $\Gamma = a : \phi,\ b_1 : *, \ldots, b_n : *$ and $r \equiv a$. The result follows.
- The case ($\bot\mathbf{E}$). Suppose $\Phi \vdash \bot$. By inductive hypothesis, there exists $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \bot$. Applying ($\mathbf{T}\bot\mathbf{E}$), we have $\Gamma \vdash \mathsf{xf}(r) : \phi$ for arbitrary ϕ. The result follows.
- The case ($\Rightarrow\mathbf{I}$). Suppose $\Phi \vdash \phi \Rightarrow \psi$ and $\Phi,\ \phi \vdash \psi$. By inductive hypothesis there exists $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi \cup \{\phi\}$ and $\Gamma \vdash r : \psi$. If $a : \phi \in \Gamma$ for some a then let $\Gamma' = \Gamma$. If no a exists such that $a : \phi \in \Gamma$ then let $\Gamma' = \Gamma, a : \phi$ for some a not appearing in Γ. By Theorem 33 $\Gamma' \vdash r : \psi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$. Applying ($\mathbf{T}\Rightarrow\mathbf{I}$) we have $\Gamma' \vdash \lambda a.r : \phi \Rightarrow \psi$. It is a fact that $\Gamma' \vdash a\#\lambda a.r$. Therefore by uniqueness, $important(\Gamma' \vdash \lambda a.r) = important(\Gamma' \vdash r) \setminus \{\phi\}$. The result follows.

[9] Here x ranges over all variables of level one, not necessarily permutatively, so perhaps $x = a$.

- The case ($\Rightarrow\mathbf{E}$). Suppose $\Phi \vdash \phi \Rightarrow \psi$, and $\Phi \vdash \phi$. By inductive hypothesis there exist $\Gamma' \vdash r'$ such that $\mathit{important}(\Gamma' \vdash r') \subseteq \Phi$ and $\Gamma' \vdash r' : \phi \Rightarrow \psi$ and $\Gamma \vdash r$ such that $\mathit{important}(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$. Without loss of generality we may assume that $\Gamma \cup \Gamma'$ satisfies our uniqueness condition; we rename variables of level one to make this true if necessary. Applying ($\mathbf{T}\Rightarrow\mathbf{E}$) and using the fact that $\mathit{important}(\Gamma \cup \Gamma' \vdash r'r) \subseteq \Phi$, the result follows.

- The case ($\forall\mathbf{I}$). Suppose $\Phi \vdash \forall a.\phi$ where $a \notin \mathit{fa}(\Phi)$ and $\Phi \vdash \phi$. By inductive hypothesis there exists $\Gamma \vdash r$ such that $\mathit{important}(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$. Since $a \notin \mathit{fa}(\Phi)$ we know that $a \notin \mathit{fa}(\mathit{important}(\Gamma \vdash r))$. If $a : * \in \Gamma$ then let $\Gamma' = \Gamma$. If $a : * \notin \Gamma$ then let $\Gamma' = \Gamma, a : *$. If $a : \xi \in \Gamma$ for some type ξ then we are in the pathological situation that $a \notin \mathit{fa}(\phi)$ and $a : \xi \in \Gamma$ 'by mistake'; we rename a. By Theorem 33 $\Gamma' \vdash r : \phi$. It is also a fact that $\mathit{important}(\Gamma \vdash r) = \mathit{important}(\Gamma' \vdash r)$. Applying ($\mathbf{T}\forall\mathbf{I}$) and using the fact that $\mathit{important}(\Gamma' \vdash r) = \mathit{important}(\Gamma' \vdash \lambda a.r)$, tshe result follows.

- The case ($\forall\mathbf{E}$). Suppose $\Phi \vdash \phi$ and $\Phi \vdash \forall b.\phi$. By inductive hypothesis there are Γ and r such that $\mathit{important}(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \forall b.\phi$. If $\Gamma \vdash g : *$ then let $\Gamma' = \Gamma$. If $\Gamma \nvdash g : *$ then let $\Gamma' = \Gamma \cup \{a : * \mid a \in \mathit{fa}(g)\}$; we permute atoms in r and Γ to avoid 'accidental clash' with atoms in g; by ZFA equivariance (Appendix A) we retain the inductive hypothesis. By Theorem 33 $\Gamma' \vdash r : \forall b.\phi$. It is also a fact that $\mathit{important}(\Gamma \vdash r) = \mathit{important}(\Gamma' \vdash r)$. Applying ($\mathbf{T}\forall\mathbf{E}$) and using the fact that $\mathit{important}(\Gamma' \vdash r) = \mathit{important}(\Gamma' \vdash rg)$, the result follows.

□

5. Reductions/proof-normalisation

Part of the Curry-Howard correspondence is that reductions correspond with proof-normalisation. Accordingly, we provide a theory of reductions, hence proof-normaliation, for our syntax:

Definition 43. Let the **derivable reductions** $\Gamma; \Delta \vdash r \longrightarrow s$ be inductively defined by the rules in Figure 4.

Remark 44. Some comments on the the rules in Figure 4:

- ($\mathbf{cong}\alpha$) gives us α-equivalence. It is expressed in the style of nominal terms [UPG04] and more specifically in the style of the permutation rule of nominal algebra [GM07]. It is not syntax-directed, but this will not be a problem for our proofs.

- If $\Gamma; \Delta \vdash a\#\pi \cdot X$ holds, then $(\pi \cdot X)[a \mapsto t]$ can reduce in context $\Gamma; \Delta$ — otherwise, $(\pi \cdot X)[a \mapsto t]$ is simply stuck. In practice we will freshly extend the context (Definition 51) and use ($\mathbf{cong}\alpha$) to *guarantee* the side-condition.

- The freshness conditions in ($\rightarrow\lambda\mathbf{a}$) avoid 'accidental capture'. Because both a and b could have types, we must avoid capture not only with respect to t, but also with respect to these types. This is the reason for the $a\#b$ and $b\#a$ conditions. This retains subject reduction (Theorem 48) by excluding the possibility that for example $a : * \in \Gamma$ and $b : \psi \in \Gamma$ and $a \in \mathit{fa}(\psi)$.

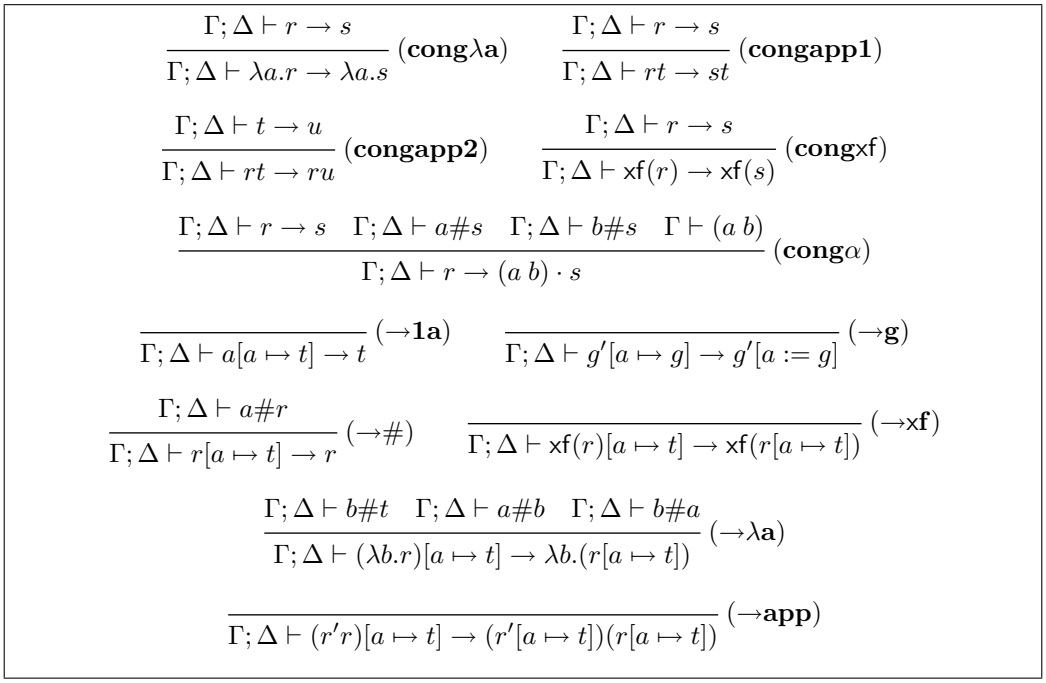


Figure 4: Congruence and reduction rules

- We prefer a small-step semantics $r \to s$ over a big-step semantics $r \Downarrow s$, because the proof-theorist can view our small step semantics as directly describing a proof-normalisation algorithm.

  Note also that terms need not have any normal form (the 'canonical form' of Definition 57 is not a normal form, but it serves a similar function), so it is not the case that there is a natural fixed normal form to 'go to' in the big step semantics.

  It is possible to identify a normal form up to rearranging substitutions in a suitable sense, but we leave this to future work, and we do not need it for our proofs.

5.1. Commuting properties of (Tfr)

Given a typing derivation, we may commute all instances of (**Tfr**) down through the tree, until they reach the root, in a sense made formal by Theorem 46. We will use this in Subsection 5.2 to work with typing derivations such that the non-syntax directed rule (**Tfr**) is convenienty isolated at the end of a derivation.

Lemma 45. (**Tfr**) *may be commuted down through all other typing rules (Figure 1). The transformations involved do not increase the depth of a derivation.*

Proof. By analysing all possibilities.

- The cases (**Ta**$*$), (**Ta**ϕ) and (**TX**). There is nothing to show.

- Suppose $(\mathbf{Tfr})$ is followed by $(\mathbf{T}\bot\mathbf{E})$. That is, we have:

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r : \bot \quad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r}{\Gamma; \Delta \vdash r : \bot}\,(\mathbf{Tfr})}{\Gamma; \Delta \vdash \mathsf{xf}(r) : \phi}\,(\mathbf{T}\bot\mathbf{E})$$

  Where $A \in \{b : \psi,\ b : *\}$ and $b \notin \Delta$. We commute $(\mathbf{Tfr})$ with $(\mathbf{T}\bot\mathbf{E})$ to obtain:

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r : \bot}{\Gamma, A; \Delta, b\#\mathcal{X} \vdash \mathsf{xf}(r) : \phi}\,(\mathbf{T}\bot\mathbf{E}) \qquad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#\mathsf{xf}(r)}{\Gamma; \Delta \vdash \mathsf{xf}(r) : \phi}\,(\mathbf{Tfr})$$

  Where $A \in \{b : \psi,\ b : *\}$ and $b \notin \Delta$. As we have $\Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r$ by assumption, we can derive $\Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#\mathsf{xf}(r)$ by $(\mathbf{a}\#\mathsf{xf})$. The result follows by Lemma 31.

- Suppose $(\mathbf{Tfr})$ is followed by $(\mathbf{T}\Rightarrow\mathbf{E})$. There are two cases:

  - The left case. Assume $\Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r'$, which can be guaranteed by ZFA equivariance (Appendix A).

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r' : \phi \Rightarrow \psi \quad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r'}{\Gamma; \Delta \vdash r' : \phi \Rightarrow \psi}\,(\mathbf{Tfr}) \qquad \Gamma; \Delta \vdash r : \phi}{\Gamma; \Delta \vdash r'r : \psi}\,(\mathbf{T}\Rightarrow\mathbf{E})$$

    Where $A \in \{b : \psi,\ b : *\}$ and $b \notin \Delta$. We commute $(\mathbf{Tfr})$ with $(\mathbf{T}\Rightarrow\mathbf{E})$ to obtain:

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r' : \phi \Rightarrow \psi \quad \Gamma, A; \Delta, b\#X \vdash r : \phi}{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r'r : \psi}\,(\mathbf{Tfr}) \qquad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r'r}{\Gamma; \Delta \vdash r'r : \psi}\,(\mathbf{T}\Rightarrow\mathbf{E})$$

    Where $A \in \{b : \psi,\ b : *\}$ and $b \notin \Delta$. The result follows.

  - The right case. Assume $\Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r$, which can be guaranteed by ZFA equivariance (Appendix A).

$$\dfrac{\Gamma; \Delta \vdash r' : \phi \Rightarrow \psi \qquad \dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r : \phi \quad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r}{\Gamma; \Delta \vdash r : \phi}\,(\mathbf{Tfr})}{\Gamma; \Delta \vdash r'r : \psi}\,(\mathbf{T}\Rightarrow\mathbf{E})$$

    Where $A \in \{b : \psi,\ b : *\}$ and $b \notin \Delta$. We commute $(\mathbf{Tfr})$ with $(\mathbf{T}\Rightarrow\mathbf{E})$ to obtain:

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r' : \phi \Rightarrow \psi \quad \Gamma, A; \Delta, b\#\mathcal{X} \vdash r : \phi}{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r'r : \psi}\,(\mathbf{T}\Rightarrow\mathbf{E}) \qquad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r'r}{\Gamma; \Delta \vdash r'r : \psi}\,(\mathbf{Tfr})$$

    Where $A \in \{b : \psi,\ b : *\}$ and $b \notin \Delta$. The result follows.

- Suppose (**Tfr**) is followed by (**T⇒I**). That is we have:

$$\dfrac{\dfrac{\Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash r : \psi \quad \Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash b\#r}{\Gamma, a : \phi; \Delta \vdash r : \psi}\,(\mathbf{Tfr})}{\Gamma, a : \phi; \Delta \vdash \lambda a.r : \phi \Rightarrow \psi}\,(\mathbf{T}\Rightarrow\mathbf{I})$$

Where $A \in \{b : \psi, b : *\}$ and $b \notin \Delta$. Renaming using ZFA equivariance (Appendix A) if necessary, we assume that $b \notin fa(\phi)$. We may commute (**Tfr**) with (**T⇒I**) to obtain:

$$\dfrac{\dfrac{\Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash r : \psi}{\Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash \lambda a.r : \phi \Rightarrow \psi}\,(\mathbf{T}\Rightarrow\mathbf{I}) \quad \Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash b\#\lambda a.r}{\Gamma, a : \phi; \Delta \vdash \lambda a.r : \phi \Rightarrow \psi}\,(\mathbf{Tfr})$$

Where $A \in \{b : \psi,\ b : *\}$ and for some $b \notin \Delta$. Then, by assumption, $\Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash b\#r$ therefore $\Gamma, a : \phi, A; \Delta, b\#\mathcal{X} \vdash b\#\lambda a.r$ by (**a**#λ**b**). The result now follows from Lemma 31. The case for

$$\dfrac{\dfrac{\Gamma, a : \phi, A; \Delta, a\#\mathcal{X} \vdash r : \psi \quad \Gamma, a : \phi, A; \Delta, a\#\mathcal{X} \vdash a\#r}{\Gamma, a : \phi; \Delta \vdash r : \psi}\,(\mathbf{Tfr})}{\Gamma, a : \phi; \Delta \vdash \lambda a.r : \phi \Rightarrow \psi}\,(\mathbf{T}\Rightarrow\mathbf{I})$$

where $A \in \{a : \phi,\ a : *\}$ and for some $a \notin \Delta$, follows immediately, using Lemma 31, as $a\#\lambda a.r$ always, by (**a**#λ**a**).

- Suppose (**Tfr**) is followed by (**T∀I**). That is we have:

$$\dfrac{\dfrac{\Gamma, a : *, A; \Delta, b\#\mathcal{X} \vdash r : \phi \quad \Gamma, a : *, A; \Delta, b\#\mathcal{X} \vdash b\#r}{\Gamma, a : *; \Delta \vdash r : \phi}\,(\mathbf{Tfr})}{\Gamma, a : *; \Delta \vdash \lambda a.r : \forall a.\phi}\,(\mathbf{T}\forall\mathbf{I})$$

Where $A \in \{b : \psi,\ b : *\}$, $b \notin \Delta$ and $a \notin fa(important(\Gamma, a : *; \Delta \vdash r))$. We may commute (**Tfr**) with (**T∀I**) to obtain:

$$\dfrac{\dfrac{\Gamma, a : *, A; \Delta, b\#\mathcal{X} \vdash r : \phi}{\Gamma, a : *, A; \Delta, b\#\mathcal{X} \vdash \lambda a.r : \forall a.\phi}\,(\mathbf{T}\forall\mathbf{I}) \quad \Gamma, a : *, A; \Delta, b\#\mathcal{X} \vdash b\#\lambda a.r}{\Gamma, a : *; \Delta \vdash \lambda a.r : \forall a.\phi}\,(\mathbf{Tfr})$$

Where $A \in \{b : \psi,\ b : *\}$, $b \notin \Delta$ and by assumption, $a \notin fa(important(\Gamma, a : *; \Delta \vdash r))$. The result follows.

- Suppose (**Tfr**) is followed by (**T∀E**). That is, we have:

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r : \forall a.\phi \quad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r}{\Gamma; \Delta \vdash r : \forall a.\phi}\,(\mathbf{Tfr})}{\Gamma; \Delta \vdash rg : \phi[a := g]}\,(\mathbf{T}\forall\mathbf{E})$$

Where $A \in \{b : \xi,\ b : *\}$ and $b \notin \Delta$. Renaming if necessary, we suppose that a and b are distinct. Also permuting if necessary, we suppose that $b \notin fa(g)$; we use ZFA equivariance (Appendix A) to retain the inductive hypothesis. We commute $(\mathbf{Tfr})$ with $(\mathbf{T}\forall\mathbf{E})$ to obtain:

$$\dfrac{\dfrac{\Gamma, A; \Delta, b\#\mathcal{X} \vdash r : \forall a.\phi}{\Gamma, A; \Delta, b\#\mathcal{X} \vdash rg : \phi[a := g]}\,(\mathbf{T}\forall\mathbf{E}) \qquad \Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#rg}{\Gamma; \Delta \vdash rg : r[a := g]}\,(\mathbf{Tfr})$$

Where $A \in \{b : \xi, b : *\}$ and $b \notin \Delta$. As $\Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#r$, we can derive $\Gamma, A; \Delta, b\#\mathcal{X} \vdash b\#rg$ using $(\mathbf{a\#app})$. The result follows by Lemma 31.

□

Theorem 46. *Suppose* Π *is a derivation of* $\Gamma; \Delta \vdash r : \phi$*. Then there exists a derivation* Π'*, also of* $\Gamma; \Delta \vdash r : \phi$*, such that* Π' *ends in some (possibly zero many) instances of* $(\mathbf{Tfr})$*, and contains no other instances of* $(\mathbf{Tfr})$*.*

In words: "all instances of $(\mathbf{Tfr})$ *may be pushed to the conclusion of the typing derivation".*

Proof. By Lemma 45. □

5.2. Subject reduction

In this subsection we investigate three forms of consistency under reduction. Lemma 47 states that 'no variables are created' by reductions; in the terminology of [FG07b, Section 6] we can say that reduction is *uniform*. Theorem 48 is a *subject-reduction* property; if we reduce a derivation of a proposition, then it remains a derivation of that same proposition. In the proof of Theorem 48, we induct on the derivation of a reduction $r \to s$ and perform case analysis on the way in which $r : \phi$ can have been derived; the results of Subsection 5.1 are needed here, to control the non-syntax-directed rule $(\mathbf{Tfr})$.

Lemma 49 is an *object-level equivariance* property; reduction is preserved under permuting variables of level one (compare this with [FG07b, Theorem 50, part 3]). Lemma 49 is technically important in Subsection 5.3 for Lemma 61, which is the key result for confluence (Theorem 62).

Lemma 47. *If* $\Gamma; \Delta \vdash a\#r$ *and* $\Gamma; \Delta \vdash r \to s$ *then* $\Gamma; \Delta \vdash a\#s$*.*

Proof. By induction on the derivation of $\Gamma; \Delta \vdash r \to s$.

- The case $(\to\mathbf{1a})$. There are two cases:
  - The case $a[a \mapsto t]$. We have $\Gamma; \Delta \vdash a[a \mapsto t] \to t$ and $\Gamma; \Delta \vdash a\#(\lambda a.a)t$. This implies that $\Gamma; \Delta \vdash a\#\lambda a.a$ and $\Gamma; \Delta \vdash a\#t$. The result follows.
  - The case $b[b \mapsto u]$. We have $\Gamma; \Delta \vdash b[b \mapsto u] \to u$ and $\Gamma; \Delta \vdash a\#(\lambda b.b)u$. This implies that $\Gamma; \Delta \vdash a\#\lambda b.b$ and $\Gamma; \Delta \vdash a\#u$. The result follows.
- The case $(\to\mathbf{g})$. We have $\Gamma; \Delta \vdash g'[a \mapsto g] \to g'[a := g]$. It is a fact that $fa(g'[a \mapsto g]) \subseteq fa(g'[a := g])$. The result follows.
- The case $(\to\#)$ where $\Gamma; \Delta \vdash a\#r$. There are two cases:

- The case $a\#r[a \mapsto t]$. The result follows by assumption.
- The case $b\#r[a \mapsto t]$. We have $\Gamma; \Delta \vdash b\#(\lambda a.r)t$ hence $\Gamma; \Delta \vdash b\#r$ and $\Gamma; \Delta \vdash b\#t$. The result follows from the fact that $\Gamma; \Delta \vdash r[a \mapsto t] \to r$.

- The case ($\to$**app**). There are two cases:
  - The case $(r'r)[a \mapsto t]$. We have $\Gamma; \Delta \vdash (r'r)[a \mapsto t] \to r'[a \mapsto t](r[a \mapsto t])$ and by assumption, $\Gamma; \Delta \vdash a\#t$. Note that $\Gamma; \Delta \vdash a\#\lambda a.r'$ $\Gamma; \Delta \vdash a\#\lambda a.r$ and $\Gamma; \Delta \vdash a\#t$. The result follows.
  - The case $\Gamma; \Delta \vdash (r'r)[b \mapsto u]$. We have $\Gamma; \Delta \vdash (r'r)[b \mapsto u] \to r'[b \mapsto u](r[b \mapsto u])$ and $\Gamma; \Delta \vdash a\#(r'r)[b \mapsto u]$. This implies that $\Gamma; \Delta \vdash a\#\lambda b.(r'r)$ therefore $\Gamma; \Delta \vdash a\#r'$ and $\Gamma; \Delta \vdash a\#r$ and also that $\Gamma; \Delta \vdash a\#u$. With these facts, we may derive $\Gamma; \Delta \vdash a\#((\lambda b.r')u)((\lambda b.r)u)$. The result follows.

- The case ($\to\lambda$**a**). There are several cases:
  - The case $(\lambda b.r)[a \mapsto t]$ where $\Gamma; \Delta \vdash b\#t$, $\Gamma; \Delta \vdash b\#a$, and $\Gamma; \Delta \vdash a\#b$. We have $\Gamma; \Delta \vdash (\lambda b.r)[a \mapsto t] \to \lambda b.(r[a \mapsto t])$ and $\Gamma; \Delta \vdash a\#(\lambda a.(\lambda b.r))t$. This implies that $\Gamma; \Delta \vdash a\#\lambda a.(\lambda b.r)$ and $\Gamma; \Delta \vdash a\#t$. With this, $\Gamma; \Delta \vdash a\#\lambda b.((\lambda a.r)t)$. The result follows.
  - The case $(\lambda b.r)[c \mapsto v]$ where $\Gamma; \Delta \vdash b\#v$, $\Gamma; \Delta \vdash b\#c$, and $\Gamma; \Delta \vdash c\#b$. We have $\Gamma; \Delta \vdash (\lambda b.r)[c \mapsto v] \to \lambda b.(r[c \mapsto v])$ and $\Gamma; \Delta \vdash a\#(\lambda c.(\lambda b.r))v$. This implies that $\Gamma; \Delta \vdash a\#\lambda c.(\lambda b.r)$ therefore $\Gamma; \Delta \vdash a\#r$ and also $\Gamma; \Delta \vdash a\#v$. With these facts, we may derive $\Gamma; \Delta \vdash a\#\lambda b.((\lambda c.r)v)$. The result follows.
  - The case $(\lambda a.r)[b \mapsto u]$ where $\Gamma; \Delta \vdash a\#u$, $\Gamma; \Delta \vdash b\#a$, and $\Gamma; \Delta \vdash a\#b$. We have $\Gamma; \Delta \vdash (\lambda a.r)[b \mapsto u] \to \lambda a.(r[b \mapsto u])$. It is a fact that $\Gamma; \Delta \vdash a\#\lambda a.(r[b \mapsto u])$ always. The result follows.

- The case ($\to$**xf**). There are two cases:
  - The case $\mathsf{xf}(r)[a \mapsto t]$. We have $\Gamma; \Delta \vdash \mathsf{xf}(r)[a \mapsto t] \to \mathsf{xf}(r[a \mapsto t])$ and $\Gamma; \Delta \vdash a\#\lambda a.(\mathsf{xf}(r))t$. This implies that $\Gamma; \Delta \vdash a\#\lambda a.(\mathsf{xf}(r))$ and $\Gamma; \Delta \vdash a\#t$. With this, we may derive $\Gamma; \Delta \vdash a\#\mathsf{xf}((\lambda a.r)t)$. The result follows.
  - The case $\mathsf{xf}(r)[b \mapsto u]$. We have $\Gamma; \Delta \vdash \mathsf{xf}(r)[b \mapsto u] \to \mathsf{xf}(r[b \mapsto u])$ and $\Gamma; \Delta \vdash a\#\lambda b.(\mathsf{xf}(r))u$. This implies that $\Gamma; \Delta \vdash a\#\lambda b.(\mathsf{xf}(r))$ therefore $\Gamma; \Delta \vdash a\#r$ and $\Gamma; \Delta \vdash a\#u$. With these facts, we may derive $\Gamma; \Delta \vdash a\#\mathsf{xf}((\lambda b.r)u)$. The result follows.

- The case (**cong**λ**a**). There is only a one case, as $\Gamma; \Delta \vdash a\#\lambda a.s$ always. Therefore, consider $\Gamma; \Delta \vdash a\#\lambda b.r$. This implies $\Gamma; \Delta \vdash a\#r$. By inductive hypothesis $\Gamma; \Delta \vdash a\#r$ implies $\Gamma; \Delta \vdash a\#s$. Therefore $\Gamma; \Delta \vdash a\#\lambda b.s$. The result follows.

- The case (**congapp1**). By assumption, $\Gamma; \Delta \vdash a\#rt$. This implies that $\Gamma; \Delta \vdash a\#r$ and $\Gamma; \Delta \vdash a\#t$. By inductive hypothesis, $\Gamma; \Delta \vdash a\#r$ implies $\Gamma; \Delta \vdash a\#s$. Therefore, $\Gamma; \Delta \vdash a\#st$. The result follows.

- The case (**congapp2**) is similar to the previous case.

- The case (**cong**xf). By assumption, $\Gamma;\Delta \vdash a\#\mathsf{xf}(r)$ therefore $\Gamma;\Delta \vdash a\#r$. By inductive hypothesis, $\Gamma;\Delta \vdash a\#r$ implies $\Gamma;\Delta \vdash a\#s$. Therefore, $\Gamma;\Delta \vdash a\#\mathsf{xf}(s)$. The result follows.
- The case (**cong**α). By assumption, $\Gamma;\Delta \vdash b\#s$, $\Gamma;\Delta \vdash c\#s$ and $\Gamma;\Delta \vdash a\#r$. By inductive hypothesis, $\Gamma;\Delta \vdash a\#s$. By Lemma 19, we have $\Gamma;\Delta \vdash a\#(b\ c)\cdot s$. The result follows.

□

Theorem 48 (Subject Reduction)**.** *If* $\Gamma;\Delta \vdash r : \phi$ *and* $\Gamma;\Delta \vdash r \to s$ *then* $\Gamma;\Delta \vdash s : \phi$.

Proof. By induction on the derivation of $\Gamma;\Delta \vdash r \to s$. In a sense illustrated in the case of (→**1a**), we may conveniently ignore (**Tfr**).

- The case (→**1a**). Suppose $\Gamma;\Delta \vdash a[a \mapsto t] \to t$. By definition $a[a \mapsto t] \equiv (\lambda a.a)t$. Suppose $\Gamma;\Delta \vdash (\lambda a.a)t : \phi$. By Theorem 46 there is a derivation of $\Gamma;\Delta \vdash (\lambda a.a)t : \phi$ such that all instances of (**Tfr**) occur at the end of the derivation.

  Therefore, for some Γ' and Δ' extending Γ and Δ, $\Gamma';\Delta' \vdash \lambda a.a : \phi \Rightarrow \phi$ and $\Gamma';\Delta' \vdash t : \phi$. Using Lemma 47 (or by concrete calculations) we can now extend the derivation of $\Gamma';\Delta' \vdash t : \phi$ to a derivation of $\Gamma;\Delta \vdash t : \phi$, as required.

  From now on, we will elide the treatment of (**Tfr**), which is identical in all the following cases.
- The case (→**g**). Suppose $\Gamma;\Delta \vdash g'[a \mapsto g] \to g[a := g]$. It is a fact that $fa(g'[a := g]) \subseteq fa(g'[a \mapsto g])$. It follows that if $\Gamma;\Delta \vdash g'[a \mapsto g] : *$ then $\Gamma;\Delta \vdash g'[a := g] : *$.
- The case (→#) where $\Gamma;\Delta \vdash a\#r$. Suppose $\Gamma;\Delta \vdash r[a \mapsto t] \to r$. By definition $r[a \mapsto t] \equiv (\lambda a.r)t$. Suppose $\Gamma;\Delta \vdash (\lambda a.r)t : \phi$. There are now two cases:
  - $a : \phi' \in \Gamma$ and $\Gamma;\Delta \vdash r : \phi$ and $\Gamma;\Delta \vdash t : \phi'$.
  - $a : * \in \Gamma$ and $\Gamma;\Delta \vdash r : \phi$ and $\Gamma;\Delta \vdash t : *$.

  In either case, $\Gamma;\Delta \vdash r : \phi$.
- The case (→**app**). Suppose $\Gamma;\Delta \vdash (r'r)[a \mapsto t] \to (r'[a \mapsto t])(r[a \mapsto t])$. By definition $(r'r)[a \mapsto t] \equiv (\lambda a.(r'r))t$ and $(r'[a \mapsto t])(r[a \mapsto t]) \equiv ((\lambda a.r')t)((\lambda a.r)t)$. There are now four cases:
  - $\Gamma;\Delta \vdash (\lambda a.(r'r))t : \phi$ because $a : \psi \in \Gamma$, $\Gamma;\Delta \vdash t : \psi$, $\Gamma;\Delta \vdash r' : \phi' \Rightarrow \phi$, and $\Gamma;\Delta \vdash r : \phi' \in \Gamma$.
  - $\Gamma;\Delta \vdash (\lambda a.(r'r))t : \phi[a := t]$ because $a : * \in \Gamma$, $\Gamma;\Delta \vdash t : *$, $\Gamma;\Delta \vdash r' : \phi' \Rightarrow \phi$, and $\Gamma;\Delta \vdash r : \phi' \in \Gamma$.
  - $(\lambda a.(r'r))t : \phi[b := r]$ because $a : \psi$, $\Gamma;\Delta \vdash t : \psi$, $\Gamma;\Delta \vdash r' : \forall b.\phi$, and $\Gamma;\Delta \vdash r : *$.
  - $(\lambda a.(r'r))t : \phi[b := r][a := t]$ because $a : * \in \Gamma$, $\Gamma;\Delta \vdash t : *$, $\Gamma;\Delta \vdash r' : \forall b.\phi$, and $\Gamma;\Delta \vdash r : *$.

  In each case, it is routine to verify the result. For example in the final case, it is routine to verify that $\Gamma;\Delta \vdash ((\lambda a.r')t)((\lambda a.r)t) : \phi[b := r][a := t]$ as required.

- The case ($\rightarrow\lambda\mathbf{a}$). Suppose $\Gamma;\Delta \vdash (\lambda b.r)[a \mapsto t] \rightarrow \lambda b.(r[a \mapsto t])$ where $\Gamma;\Delta \vdash b\#t$, $\Gamma;\Delta \vdash b\#a$, and $\Gamma;\Delta \vdash a\#b$. By definitinion $(\lambda b.r)[a \mapsto t] \equiv (\lambda a.(\lambda b.r))t$ and $\lambda b.(r[a \mapsto t]) \equiv \lambda b.((\lambda a.r)t)$. There are now two cases:
  - $\Gamma;\Delta \vdash (\lambda a.(\lambda b.r))t : \phi' \Rightarrow \phi$ because $a : \psi,\ b : \phi' \in \Gamma$, $\Gamma;\Delta \vdash t : \psi$, and $\Gamma;\Delta \vdash r : \phi$.
  - $\Gamma;\Delta \vdash (\lambda a.(\lambda b.r))t : (\phi' \Rightarrow \phi)[a := t]$ because $a : *,\ b : \phi' \in \Gamma$, $\Gamma;\Delta \vdash t : *$, and $\Gamma;\Delta \vdash r : \phi$.
  - $\Gamma;\Delta \vdash (\lambda a.(\lambda b.r))t : \forall b.\phi$ because $a : \psi,\ b : * \in \Gamma$, $\Gamma;\Delta \vdash t : \psi$, and $\Gamma;\Delta \vdash r : \phi$.
  - $\Gamma;\Delta \vdash (\lambda a.(\lambda b.r))t : (\forall b.\phi)[a := t]$ because $a : *,\ b : * \in \Gamma$, $\Gamma;\Delta \vdash t : *$, and $\Gamma;\Delta \vdash r : \phi$.

  In each case, it is routine to verify the result. For example in the final case, that $\Gamma;\Delta \vdash \lambda b.((\lambda a.r)t) : \forall b.(\phi[a := t])$.
- The case ($\rightarrow\mathsf{xf}\mathbf{f}$). Suppose $\Gamma;\Delta \vdash \mathsf{xf}(r)[a \mapsto t] : \phi$. By definition $\mathsf{xf}(r)[a \mapsto t] \equiv (\lambda a.(\mathsf{xf}(r)))t$. There are now two cases:
  - $\Gamma;\Delta \vdash (\lambda a.(\mathsf{xf}(r)))t : \bot$ because $a : \psi \in \Gamma$ and $\Gamma;\Delta \vdash r : \phi$.
  - $\Gamma;\Delta \vdash (\lambda a.(\mathsf{xf}(r)))t : \bot$ because $a : * \in \Gamma$ and $\Gamma;\Delta \vdash r : \phi$.

  In each case, it is routine to verify the result. For example in the first case, that $\Gamma;\Delta \vdash \mathsf{xf}(r) : \bot$.
- The case ($\mathbf{cong}\alpha$). Suppose $\Gamma;\Delta \vdash r : \phi$, Suppose $\Gamma;\Delta \vdash (a\ b)$ and $\Gamma;\Delta \vdash a\#r$ and $\Gamma;\Delta \vdash b\#r$. Since $\Gamma;\Delta \vdash r \rightarrow s$, by inductive hypothesis $\Gamma;\Delta \vdash s : \phi$. By Lemma 47, $\Gamma;\Delta \vdash a\#s$ and $\Gamma;\Delta \vdash b\#s$. It follows by Lemma 25 that $\Gamma;\Delta \vdash (a\ b) \cdot s : \phi$, as required.
- The case ($\mathbf{cong}\lambda\mathbf{a}$). Suppose $\Gamma;\Delta \vdash r \rightarrow s$ and $\Gamma;\Delta \vdash \lambda a.r \rightarrow \lambda a.s$. There are now two cases:
  - $\Gamma;\Delta \vdash \lambda a.r : \phi' \Rightarrow \phi$ and $a : \phi' \in \Gamma$. By inductive hypothesis $\Gamma;\Delta \vdash s : \phi$. By ($\mathbf{T}\Rightarrow\mathbf{I}$) $\Gamma;\Delta \vdash s : \phi' \Rightarrow \phi$.
  - $\Gamma;\Delta \vdash \lambda a.r : \forall a.\phi$ and $a : * \in \Gamma$. By inductive hypothesis $\Gamma;\Delta \vdash s : \phi$. Using Lemma 47 we see that $important(\Gamma \vdash s) \subseteq important(\Gamma \vdash r)$. It follows using ($\mathbf{T}\forall\mathbf{I}$) that $\Gamma;\Delta \vdash \lambda a.s : \forall a.\phi$.
- The cases of ($\mathbf{congapp1}$), ($\mathbf{congapp2}$), and ($\mathbf{cong}\mathsf{xf}$) are no harder.

□

Lemma 49. *If* $\Gamma;\Delta \vdash r \rightarrow s$ *then* $\Gamma;\Delta \vdash \pi \cdot r \rightarrow \pi \cdot s$.

Proof. By induction on derivations.

- The case ($\rightarrow\mathbf{1a}$). We have $\pi \cdot (a[a \mapsto t]) \equiv \pi(a)[\pi(a) \mapsto \pi \cdot t]$. The result follows from ($\rightarrow\mathbf{1a}$).

- The case $(\rightarrow\mathbf{g})$ is similar.
- The case $(\rightarrow\#)$ where $\Gamma;\Delta \vdash a\#r$. We have $\pi \cdot (r[a \mapsto t]) \equiv (\pi \cdot r)[\pi(a) \mapsto \pi \cdot t]$. By Lemma 19, we have $\Gamma;\Delta \vdash \pi(a)\#\pi \cdot r$. Applying $(\rightarrow\#)$, we derive $\Gamma;\Delta \vdash (\pi \cdot r)[\pi(a) \mapsto \pi \cdot t] \rightarrow \pi \cdot r$. The result follows.
- The case $(\rightarrow\mathbf{app})$. We have $\pi \cdot ((r'r)[a \mapsto t]) \equiv ((\pi \cdot r')(\pi \cdot r))[\pi(a) \mapsto \pi \cdot t]$. Applying $(\rightarrow\mathbf{app})$, we obtain $((\pi \cdot r')[\pi(a) \mapsto \pi \cdot t])((\pi \cdot r)[\pi(a) \mapsto \pi \cdot t])$. The result follows from the definition of the permutation action.
- The case $(\rightarrow\lambda\mathbf{a})$ where $\Gamma;\Delta \vdash b\#t$, $\Gamma;\Delta \vdash b\#a$, and $\Gamma;\Delta \vdash a\#b$. We have $\pi \cdot ((\lambda b.r)[a \mapsto t]) \equiv (\lambda\pi(b).(\pi \cdot r))[\pi(a) \mapsto \pi \cdot t]$. By Lemma 19, we have $\Gamma;\Delta \vdash \pi(b)\#\pi \cdot t$. Applying $(\rightarrow\lambda\mathbf{a})$, we have $\Gamma;\Delta \vdash (\lambda\pi(b).(\pi \cdot r))[\pi(a) \mapsto \pi \cdot t] \rightarrow \lambda\pi(b).((\pi \cdot r)[\pi(a) \mapsto \pi \cdot t])$. The result follows by the definition of the permutation action.
- The case $(\rightarrow\mathsf{x}\mathbf{f})$. We have $\pi \cdot (\mathsf{x}\mathsf{f}(r)[a \mapsto t]) \equiv \mathsf{x}\mathsf{f}(\pi \cdot r)[\pi(a) \mapsto \pi \cdot t]$. Applying $(\rightarrow\mathsf{x}\mathbf{f})$, we have $\Gamma;\Delta \vdash \mathsf{x}\mathsf{f}(\pi \cdot r)[\pi(a) \mapsto \pi \cdot t] \rightarrow \mathsf{x}\mathsf{f}((\pi \cdot r)[\pi(a) \mapsto \pi \cdot r])$. The result follows.
- The case $(\mathbf{congapp1})$. By inductive hypothesis, we have $\Gamma;\Delta \vdash \pi \cdot r \rightarrow \pi \cdot s$. Applying $(\mathbf{congapp1})$, we have $\Gamma;\Delta \vdash (\pi \cdot r)(\pi \cdot t) \rightarrow (\pi \cdot s)(\pi \cdot t)$. The result follows.
- The case $(\mathbf{congapp2})$ is similar to the previous case.
- The case $(\mathbf{cong}\lambda\mathbf{a})$. By inductive hypothesis, we have $\Gamma;\Delta \vdash \pi \cdot r \rightarrow \pi \cdot s$. Applying $(\mathbf{cong}\lambda\mathbf{a})$, we have $\Gamma;\Delta \vdash \lambda\pi(a).(\pi \cdot r) \rightarrow \lambda\pi(a).(\pi \cdot s)$. The result follows.
- The case $(\mathbf{cong}\mathsf{x}\mathsf{f})$. By inductive hypothesis, we have $\Gamma;\Delta \vdash \pi \cdot r \rightarrow \pi \cdot s$. Applying $(\mathbf{cong}\mathsf{x}\mathsf{f})$, we have $\Gamma;\Delta \vdash \mathsf{x}\mathsf{f}(\pi \cdot r) \rightarrow \mathsf{x}\mathsf{f}(\pi \cdot s)$. The result follows.
- The case $(\mathbf{cong}\alpha)$. By inductive hypothesis, we have $\Gamma;\Delta \vdash \pi \cdot r \rightarrow \pi \cdot s$. Further, by Lemma 19, we have $\Gamma;\Delta \vdash \pi(a)\#\pi \cdot s$ and $\Gamma;\Delta \vdash \pi(b)\#\pi \cdot s$. Applying $(\mathbf{cong}\alpha)$, we obtain $\Gamma;\Delta \vdash \pi \cdot r \rightarrow (\pi(a)\ \pi(b)) \cdot (\pi \cdot s)$. The result follows from elementary properties of permutations.

□

5.3. Confluence

The next notion of correctness, following subject reduction discussed in Theorem 48 in Subsection 5.2, is confluence and the identification of a canonical reduced version of any derivation. This is Definition 57 and Theorem 62.

Choose some fixed but arbitrary order on variables of level one. If S is a finite set of variables of level one say 'for the first variable of level one not in S' to mean 'for the least variable of level one, in our fixed but arbitrary order, that is not an element of S'. This is convenient, though not necessary, for expressing the proofs to follow. We will never make infinitely many choices of fresh variable of level one, and nowhere will the truth of a result depend on our choice of order.

Definition 50. Define a **substitution action on terms**, with respect to a typing and freshness context, $\Gamma; \Delta$ by the rules below. Earlier rules take priority.

- $r[a := t] \equiv r$ if $\Gamma; \Delta \vdash a\#r$.
- $a[a := t] \equiv t$.
- $(\pi \cdot X)[a := t] \equiv (\pi \cdot X)[a \mapsto t]$.
- $(\lambda b.r)[a := t] \equiv \lambda b.(r[a := t])$ if $\Gamma; \Delta \vdash b\#t$, $\Gamma; \Delta \vdash a\#b$, and $\Gamma; \Delta \vdash b\#a$.
- $(\lambda b.r)[a := t] \equiv \lambda c.(((b\ c) \cdot r)[a := t])$ if $\Gamma; \Delta \nvdash b\#t$. Here c is the first fresh variable of level one of the same type as b in Γ if such a variable exists.

  That is, c is the first variable of level one, with the same type as b, according to our order, that is not mentioned in r, t, a, or b (so $\Gamma; \Delta \vdash c\#r$ and $\Gamma; \Delta \vdash c\#t$), if such a c exists.
- $(r'r)[a := t] \equiv r'[a := t](r[a := t])$
- $\mathsf{xf}(r)[a := t] \equiv \mathsf{xf}(r[a := t])$

Note in Definition 50 that $r[a := t]$ depends on $\Gamma; \Delta$ and that $r[a := t]$ need not necessarily exist. This is a partial definition; however, we may guarantee that $r[a := t]$ exists for suitably extended versions of Γ and Δ.

Definition 51. Suppose that $\Gamma; \Delta$ and $\Gamma^+; \Delta^+$ are pairs of typing and freshness contexts. Say $\Gamma^+; \Delta^+$ **freshly extends** $\Gamma; \Delta$ when there exist Γ' and Δ' such that:

- Γ' gives types to the variables of level one appearing in Δ', and to no other variables of level one (and to no variables of level two).
- If $a\#X \in \Delta'$ then X is given a type in Γ.
- If $a\#X \in \Delta'$ then a does not appear in Γ or Δ.
- $\Delta^+ = \Delta \cup \Delta'$
- $\Gamma^+ = \Gamma \cup \Gamma'$

Intuitively, $\Gamma^+; \Delta^+$ 'extends $\Gamma; \Delta$ with some fresh variables of level one'.

Definition 52. Define $\twoheadrightarrow$ as the **reflexive, transitive closure** of $\rightarrow$.

Lemma 53. *Suppose that* $\Gamma \subseteq \Gamma'$ *and* $\Delta \subseteq \Delta'$.
If $\Gamma; \Delta \vdash r \rightarrow s$ *then* $\Gamma'; \Delta' \vdash r \rightarrow s$.

Proof. By induction on the derivation of $\Gamma; \Delta \vdash r \rightarrow s$.

- The case ($\rightarrow$**1a**). Since $\Gamma'; \Delta' \vdash a[a \mapsto t] \rightarrow t$ always, by ($\rightarrow$**1a**).
- The case ($\rightarrow$**g**) is similar.
- The case ($\rightarrow\#$). Suppose $\Gamma; \Delta \vdash a\#r$. By Lemma 31, $\Gamma'; \Delta' \vdash a\#r$. Then $\Gamma'; \Delta' \vdash r[a \mapsto t] \rightarrow r$. The result follows.

- The case $(\rightarrow \mathsf{xf})$. Since $\Gamma';\Delta' \vdash \mathsf{xf}(r)[a \mapsto t] \rightarrow \mathsf{xf}(r[a \mapsto t])$ always, by $(\rightarrow \mathbf{xf})$.
- The case $(\rightarrow \lambda \mathbf{a})$. Suppose $\Gamma;\Delta \vdash b\#t$, $\Gamma;\Delta \vdash b\#a$, and $\Gamma;\Delta \vdash a\#b$. By Lemma 31, $\Gamma';\Delta' \vdash b\#t$, and similarly for $b\#a$ and $a\#b$. Then $\Gamma';\Delta' \vdash (\lambda b.r)[a \mapsto t] \rightarrow \lambda b.(r[a \mapsto t])$ always, by $(\rightarrow \lambda \mathbf{a})$. The result follows.
- The case $(\rightarrow \mathbf{app})$. Since $\Gamma';\Delta' \vdash (r'r)[a \mapsto t] \rightarrow r'[a \mapsto t](r[a \mapsto t])$ always, by $(\rightarrow \mathbf{app})$.
- The case $(\mathbf{cong}\lambda \mathbf{a})$. By inductive hypothesis, $\Gamma';\Delta' \vdash r \rightarrow s$. Applying $(\mathbf{cong}\lambda \mathbf{a})$, we obtain $\Gamma';\Delta' \vdash \lambda a.r \rightarrow \lambda a.s$. The result follows.
- The case $(\mathbf{congapp1})$. By inductive hypothesis, $\Gamma';\Delta' \vdash r \rightarrow s$. Applying $(\mathbf{congapp1})$, we obtain $\Gamma';\Delta' \vdash rt \rightarrow st$. The result follows.
- The case $(\mathbf{congapp2})$. By inductive hypothesis, $\Gamma';\Delta' \vdash t \rightarrow u$. Applying $(\mathbf{congapp1})$, we obtain $\Gamma';\Delta' \vdash rt \rightarrow ru$. The result follows.
- The case $(\mathbf{cong}\mathsf{xf})$. By inductive hypothesis, $\Gamma';\Delta' \vdash r \rightarrow s$. Applying $(\mathbf{cong}\mathsf{xf})$, we obtain $\Gamma';\Delta' \vdash \mathsf{xf}(r) \rightarrow \mathsf{xf}(s)$. The result follows.
- The case $(\mathbf{cong}\alpha)$. Suppose $\Gamma;\Delta \vdash r \rightarrow s$, with $\Gamma;\Delta \vdash a\#s$, $\Gamma;\Delta \vdash b\#s$ and $\Gamma \vdash (a\ b)$. By inductive hypothesis, $\Gamma';\Delta' \vdash r \rightarrow s$. By Lemma 31, $\Gamma';\Delta' \vdash a\#s$ and $\Gamma';\Delta' \vdash b\#s$. As Γ' satisfies Definition 6, we have $\Gamma' \vdash (a\ b)$. Applying $(\mathbf{cong}\alpha)$, we obtain $\Gamma';\Delta' \vdash r \rightarrow (a\ b) \cdot s$. The result follows.

□

The following definition and lemma are used in the proof of Lemma 56.

Definition 54. Define the **depth** of a term by:

$$depth(a) = 1 \quad depth(\pi \cdot X) = 1 \quad depth(\lambda a.r) = 1 + depth(r)$$
$$depth(r'r) = depth(r') + depth(r) \quad depth(\mathsf{xf}(r)) = 1 + depth(r)$$

Lemma 55. $depth(\pi \cdot r) = depth(r)$

Proof. By induction on r.

- The case a. Since π is a bijection on variables of level one.
- The case $\pi' \cdot X$. By Lemma 18, $\pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X$. Further, $depth((\pi \circ \pi') \cdot X) = depth(\pi' \cdot X)$. The result follows.
- The case $\lambda a.r$. We have $\pi \cdot \lambda a.r \equiv \lambda \pi(a).(\pi \cdot r)$. By inductive hypothesis, $depth(\pi \cdot r) = depth(r)$. The result follows.
- The case $r'r$. By inductive hypothesis, $depth(\pi \cdot r') = depth(r')$ and $depth(\pi \cdot r) = depth(r)$. The result follows.
- The case $\mathsf{xf}(r)$. By inductive hypothesis, $depth(\pi \cdot r) = depth(r)$. The result follows.

□

Lemma 56. *For every $\Gamma; \Delta$, r, a, s and t there exist Γ^+ freshly extending Γ and Δ^+ freshly extending Δ such that*

$$\Gamma^+; \Delta^+ \vdash r[a \mapsto t] \twoheadrightarrow r[a := t].$$

($r[a := t]$ calculated for $\Gamma^+; \Delta^+$.)

Proof. By induction on the depth of r.

- The case a. We have $a[a := t] \equiv t$ and also $\Gamma; \Delta \vdash a[a \mapsto t] \twoheadrightarrow t$.
- The case b. We have $b[a := t] \equiv b$ and also $\Gamma; \Delta \vdash b[a \mapsto t] \twoheadrightarrow b$.
- The case $\pi \cdot X$. If $\Gamma; \Delta \vdash a\#\pi \cdot X$ then we have $(\pi \cdot X)[a := t] \equiv \pi \cdot X$ and $\Gamma; \Delta \vdash (\pi \cdot X)[a \mapsto t] \twoheadrightarrow (\pi \cdot X)$ also. Otherwise, we have $\Gamma; \Delta \vdash (\pi \cdot X)[a \mapsto t] \twoheadrightarrow (\pi \cdot X)[a \mapsto t]$ and $(\pi \cdot X)[a := t] \equiv (\pi \cdot X)[a \mapsto t]$. The result follows.
- The case $\lambda a.r$. We have $(\lambda a.r)[a := t] \equiv \lambda a.r$ and $\Gamma; \Delta \vdash (\lambda a.r)[a \mapsto t] \twoheadrightarrow \lambda a.r$.
- The case $\lambda b.r$. If $\Gamma; \Delta \vdash a\#\lambda b.r$, then $(\lambda b.r)[a := t] \equiv \lambda b.r$ and $\Gamma; \Delta \vdash (\lambda b.r)[a \mapsto t] \to \lambda b.r$. Otherwise, there are two cases:
  - The case $\Gamma; \Delta \vdash b\#t$, $\Gamma; \Delta \vdash b\#a$, and $\Gamma; \Delta \vdash a\#b$. We have $(\lambda b.r)[a := t] \equiv \lambda b.(r[a := t])$. Applying $(\to\lambda\mathbf{a})$ $\Gamma; \Delta \vdash (\lambda b.r)[a \mapsto t] \to \lambda b.(r[a \mapsto t])$. By inductive hypothesis $\Gamma^+; \Delta^+ \vdash r[a \mapsto t] \twoheadrightarrow r[a := t]$ for some Γ^+ and Δ^+ freshly extending Γ and Δ. The result follows using Lemma 53.
  - Otherwise, we have $(\lambda b.r)[a := t] \equiv (\lambda c.((b\ c) \cdot r))[a := t]$ where c is a suitably fresh variable of level one of suitable type, obtained from the freshly extended contexts Γ^+ and Δ^+. Applying $(\mathbf{cong}\alpha)$ and $(\to\lambda\mathbf{a})$

    $$\Gamma^+; \Delta^+ \vdash (\lambda b.r)[a \mapsto t] \twoheadrightarrow (\lambda c.((b\ c) \cdot r))[a \mapsto t].$$

    By inductive hypothesis and Lemma 55, we have

    $$\Gamma^+; \Delta^+ \vdash ((b\ c) \cdot r)[a \mapsto t] \twoheadrightarrow ((b\ c) \cdot r)[a := t].$$

    The result follows.
- The case $r'r$. If $\Gamma; \Delta \vdash a\#r'r$ then the result follows immediately. Otherwise, $(r'r)[a := t] \equiv r'[a := t](r[a := t])$ and $\Gamma; \Delta \vdash (r'r)[a \mapsto t] \twoheadrightarrow r'[a \mapsto t](r[a \mapsto t])$ by $(\to\mathbf{app})$. The result follows by the inductive hypothesis for r and r'.
- The case $\mathsf{xf}(r)$. We have $\mathsf{xf}(r)[a := t] \equiv \mathsf{xf}(r[a := t])$. The result follows by the inductive hypothesis for r.

□

Definition 57. Let the **canonical form** $r^\star$ of r in the context $\Gamma;\Delta$ be inductively defined by the following rules, where earlier rules take priority:[10]

$$\begin{aligned} g^\star &\equiv g \\ (\pi \cdot X)^\star &\equiv (\pi \cdot X) \\ (\lambda a.r)^\star &\equiv \lambda a.r^\star \\ (r[a \mapsto t])^\star &\equiv r^\star[a := t^\star] \\ (r'r)^\star &\equiv (r'^\star)(r^\star) \quad \text{where r' is not an abstraction} \\ \mathsf{xf}(r)^\star &\equiv \mathsf{xf}(r^\star) \end{aligned}$$

Note that $r^\star$ depends on the context $\Gamma;\Delta$. It need not be defined, because $[a := t^\star]$ need not always be defined (see the comment following Definition 50). However, in a sense made formal by Lemma 58 this does not matter because we can freshly extend the context.

Note also that $(r^\star)^\star \not\equiv r^\star$ in general, so taking the canonical form is not idempotent. This will not be a problem. $r^\star$ is a choice of representative of the terms that r rewrites to; a choice which happens to be very useful for proving confluence (Theorem 62). Intuitively, $r^\star$ is a rewrite of r such that

> "all substitutions have been pushed down, and through each other, at least once".

Lemma 58. *For every $\Gamma;\Delta$ and r there exist Γ^+ and Δ^+ such that $\Gamma^+;\Delta^+ \vdash r \twoheadrightarrow r^\star$. ($r^\star$ calculated for $\Gamma^+;\Delta^+$.)*

Proof. By induction on r.

- The case a. Since $a^\star \equiv a$.
- The case $\pi \cdot X$. Since $(\pi \cdot X)^\star \equiv \pi \cdot X$.
- The case $\lambda a.r$. By inductive hypothesis, $\Gamma^+;\Delta^+ \vdash r \twoheadrightarrow r^\star$. The result follows from $\lambda a.(r^\star) \equiv (\lambda a.r)^\star$.
- The case $r'r$. The special case $r[a \mapsto t]$, by definition syntactically identical to $(\lambda a.r)t$, follows from Lemma 56. Otherwise, by inductive hypothesis, $\Gamma^+;\Delta^+ \vdash r' \twoheadrightarrow r'^\star$ and $\Gamma^+;\Delta^+ \vdash r \twoheadrightarrow r^\star$. The result follows from $r'^\star(r^\star) \equiv (r'r)^\star$.
- The case $\mathsf{xf}(r)$. By inductive hypothesis, $\Gamma^+;\Delta^+ \vdash r \twoheadrightarrow r^\star$. The result follows from $\mathsf{xf}(r^\star) \equiv \mathsf{xf}(r)^\star$.

□

For the rest of this subsection, $\Gamma^+;\Delta^+$ will be a context freshly extending $\Gamma;\Delta$ with sufficient freshness such that all the canonical forms we require exist. We will always be clear about what these canonical forms are.

[10] It has been pointed out to us that the technique we use to prove confluence resembles a proof by Makato Takahashi [Tak95]. This 'convergent evolution' is interesting and suggests to us the existence of some deeper mathematics, which we have not fully understood. At the very least, there should be a general method here which could be brought out.

Lemma 59. *Fix* $\Gamma;\Delta$. *Then* $\Gamma;\Delta \vdash a\#s$ *implies* $\Gamma;\Delta \vdash a\#s^\star$. *($s^\star$ calculated for* $\Gamma;\Delta$.*)*

Proof. By induction on derivations.

- The case $(\mathbf{a\#b})$. Since $b^\star \equiv b$.
- The case $(\mathbf{a\#g})$. Since $g^\star \equiv g$.
- The case $(\mathbf{a\#\lambda a})$. We have $(\lambda a.r)^\star \equiv \lambda a.r^\star$ and $\Gamma;\Delta \vdash a\#\lambda a.r^\star$ always.
- The case $(\mathbf{a\#\lambda b})$. By inductive hypothesis, $\Gamma;\Delta \vdash a\#r^\star$. Applying $(\mathbf{a\#\lambda b})$, we have $\Gamma;\Delta \vdash a\#\lambda b.r^\star$. The result follows.
- The case $(\mathbf{a\#X})$. As $(\pi \cdot X)^\star \equiv \pi \cdot X$.
- The case $(\mathbf{a\#b'})$. Since $b^\star \equiv b$.
- The case $(\mathbf{a\#app})$. By inductive hypothesis, $\Gamma;\Delta \vdash a\#r'^\star$ and $\Gamma;\Delta \vdash a\#r^\star$. Applying $(\mathbf{a\#app})$ and simple manipulation, we have $\Gamma;\Delta \vdash a\#(r'r)^\star$. The result follows.
- The case $(\mathbf{a\#}\mathsf{xf})$. By inductive hypothesis, $\Gamma;\Delta \vdash a\#r^\star$. Applying $(\mathbf{a\#xf})$ and simple manipulation, we obtain $\Gamma;\Delta \vdash a\#\mathsf{xf}(r)^\star$. The result follows.

□

Lemma 60. $(\pi \cdot r)^\star \equiv \pi \cdot r^\star$

Proof. By induction on r.

- The case a. Since $a^\star \equiv a$ for all variables of level one.
- The case $\pi' \cdot X$. We have $(\pi' \cdot X)^\star \equiv \pi' \cdot X$. The result follows from Lemma 18.
- The case $r'r$. We have $\pi \cdot ((r'r)^\star) \equiv (\pi \cdot (r'^\star))(\pi \cdot (r^\star))$. The result follows from the inductive hypothesis.
- The case $\lambda a.r$. Since $(\pi \cdot \lambda a.r)^\star \equiv (\lambda\pi(a).(\pi \cdot r))^\star \equiv \lambda\pi(a).((\pi \cdot r)^\star)$. Applying the inductive hypothesis, we have $\lambda\pi(a).((\pi \cdot r)^\star) \equiv \lambda\pi(a).(\pi \cdot r^\star)$. The result follows from the definition of the permutation action.
- The case $\mathsf{xf}(r)$. Since $(\pi \cdot \mathsf{xf}(r))^\star \equiv \mathsf{xf}(\pi \cdot r)^\star \equiv \mathsf{xf}((\pi \cdot r)^\star)$. By application of the inductive hypothesis, $\mathsf{xf}((\pi \cdot r)^\star) \equiv \mathsf{xf}(\pi \cdot r^\star)$. The result follows.

□

Lemma 61. $\Gamma;\Delta \vdash r \to s$ *implies* $\Gamma^+;\Delta^+ \vdash s^\star \to^* r^\star$.

Proof. By induction on derivations. Note the reversal from $r \to s$ to $s^\star \to^* r^\star$.

- The case $(\to\mathbf{1a})$. We have:

$$\Gamma;\Delta \vdash a[a \mapsto t] \to t$$
$$\Gamma^+;\Delta^+ \vdash t^\star \to^* (a[a \mapsto t])^\star$$

  as $(a[a \mapsto t])^\star \equiv t^\star$ and the result follows.

- The case ($\rightarrow$**g**). We have:

$$\Gamma; \Delta \vdash g'[a \mapsto g] \rightarrow g'[a := g]$$
$$\Gamma^+; \Delta^+ \vdash (g'[a := g])^\star \twoheadrightarrow (g'[a \mapsto g])^\star$$

as $(g'[a \mapsto g])^\star \equiv g'[a := g] \equiv (g'[a := g])^\star$ and the result follows.

- The case ($\rightarrow\#$) when $\Gamma; \Delta \vdash a\#r$. We have:

$$\Gamma; \Delta \vdash r[a \mapsto t] \rightarrow r$$
$$\Gamma^+; \Delta^+ \vdash r^\star \twoheadrightarrow (r[a \mapsto t])^\star$$

as $(r[a \mapsto t])^\star \equiv r^\star$ when $\Gamma; \Delta \vdash a\#r$.

- The case ($\rightarrow\lambda$**a**) when $\Gamma; \Delta \vdash b\#t$, $\Gamma; \Delta \vdash a\#b$, and $\Gamma; \Delta \vdash b\#a$. We have:

$$\Gamma; \Delta \vdash (\lambda b.r)[a \mapsto t] \rightarrow \lambda b.(r[a \mapsto t])$$
$$\Gamma^+; \Delta^+ \vdash (\lambda b.(r[a \mapsto t]))^\star \twoheadrightarrow ((\lambda b.r)[a \mapsto t])^\star$$

as $(\lambda b.(r[a \mapsto t]))^\star \equiv \lambda b.(r^\star[a := t^\star]) \equiv ((\lambda b.r)[a \mapsto t])^\star$.

- The case ($\rightarrow$**xf**). We have:

$$\Gamma; \Delta \vdash (\mathsf{xf}(r))[a \mapsto t] \rightarrow \mathsf{xf}(r[a \mapsto t])$$
$$\Gamma^+; \Delta^+ \vdash (\mathsf{xf}(r[a \mapsto t]))^\star \twoheadrightarrow ((\mathsf{xf}(r))[a \mapsto t])^\star$$

as $(\mathsf{xf}(r[a \mapsto t]))^\star \equiv (\mathsf{xf}(r^\star[a := t^\star])) \equiv ((\mathsf{xf}(r))[a \mapsto t])^\star$.

- The case (**cong**λ**a**). By inductive hypothesis, $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r^\star$. We now argue by induction on the path length of $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r^\star$. If the path is empty, then $s^\star \equiv r^\star$ and the result follows. Otherwise, we have the case $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r'^\star \rightarrow r^\star$. By inductive hypothesis, it follows that $\Gamma^+; \Delta^+ \vdash \lambda a.s^\star \twoheadrightarrow \lambda a.r'^\star$. Applying (**cong**$\lambda$**a**), it follows that $\Gamma^+; \Delta^+ \vdash \lambda a.r'^\star \rightarrow \lambda a.r^\star$. It now follows, by the definition of the reflexive transitive closure, that $\Gamma^+; \Delta^+ \vdash \lambda a.s^\star \twoheadrightarrow \lambda a.r^\star$. The result follows.

- The case (**congapp1**). By inductive hypothesis, $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r^\star$. Applying (**congapp1**) and an inductive argument similar to the previous case we derive $\Gamma^+; \Delta^+ \vdash s^\star(t^\star) \twoheadrightarrow r^\star(t^\star)$. The result follows.

- The case (**congapp2**) is similar to the previous case.

- The case (**cong**xf). By inductive hypothesis, $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r^\star$. By induction on the path length of $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r^\star$ and applying (**cong**xf), the result follows.

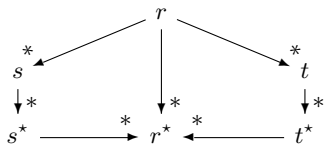
- The case (**cong**α). Suppose $s \equiv (a\ b) \cdot s'$. Suppose $\Gamma; \Delta \vdash r \rightarrow s'$ and $\Gamma; \Delta \vdash a\#s'$ and $\Gamma; \Delta \vdash b\#s'$. By inductive hypothesis, $\Gamma^+; \Delta^+ \vdash s'^\star \twoheadrightarrow r^\star$. By Lemma 59 $\Gamma^+; \Delta^+ \vdash a\#s'^\star$ and $\Gamma^+; \Delta^+ \vdash b\#s'^\star$. By Lemma 47 it follows that $\Gamma^+; \Delta^+ \vdash a\#r^\star$ and $\Gamma^+; \Delta^+ \vdash b\#r^\star$. Applying (**cong**$\alpha$) it follows that $\Gamma^+; \Delta^+ \vdash s'^\star \twoheadrightarrow (a\ b) \cdot (r^\star)$. By Lemmas 49 and 18 it follows that $\Gamma; \Delta^+ \vdash (a\ b) \cdot (s'^\star) \twoheadrightarrow r^\star$. By Lemma 60 $\Gamma^+; \Delta^+ \vdash s^\star \twoheadrightarrow r^\star$. The result follows.

□

Theorem 62. *If* $\Gamma;\Delta \vdash s \ {}^{*}\!\leftarrow\ r \twoheadrightarrow\ t$ *then* $\Gamma^{+};\Delta^{+} \vdash s \twoheadrightarrow\ u \ {}^{*}\!\leftarrow\ t$. *That is,* $\rightarrow$ *is Church–Rosser.*

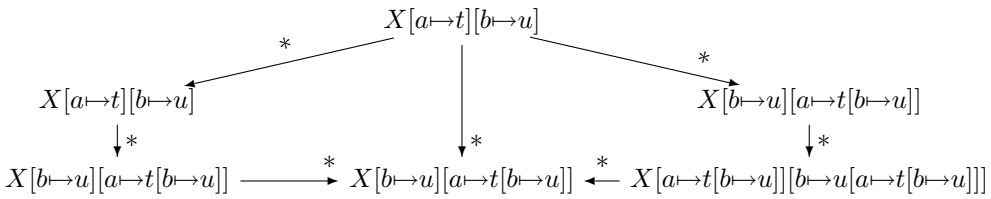
Proof. Suppose $\Gamma;\Delta \vdash r \twoheadrightarrow\ s$ and $\Gamma;\Delta \vdash r \twoheadrightarrow\ t$. By Lemma 58 fresh extensions $\Gamma_1;\Delta_1$ of $\Gamma;\Delta$ exist such that $\Gamma_1;\Delta_1 \vdash r \twoheadrightarrow\ r^{\star}$. Again by Lemma 58, fresh extensions $\Gamma_2;\Delta_2$ and $\Gamma_3;\Delta_3$ of $\Gamma;\Delta$, exist such that $\Gamma_2;\Delta_2 \vdash s \twoheadrightarrow\ s^{\star}$ and $\Gamma_3;\Delta_3 \vdash t \twoheadrightarrow\ t^{\star}$. By Lemma 61 a fresh extension $\Gamma_4;\Delta_4$ of $\Gamma_2;\Delta_2$, and a fresh extension $\Gamma_5;\Delta_5$ of $\Gamma_3;\Delta_3$, exist such that $\Gamma_4;\Delta_4 \vdash s^{\star} \twoheadrightarrow\ r^{\star}$ and $\Gamma_5;\Delta_5 \vdash t^{\star} \twoheadrightarrow\ r^{\star}$. Taking $\Gamma^{+} = \bigcup_{1\leq i\leq 5}\Gamma_i$ and $\Delta^{+} = \bigcup_{1\leq i\leq 5}\Delta_i$, the result follows.

For the reader's convenience we rewrite this argument as a diagram:



□

We illustrate this for $X[a{\mapsto}t][b{\mapsto}u]$, where $\Gamma;\Delta \vdash a\#u$:



6. Conclusions

We have shown how nominal terms, endowed with a non-trivial typing system, model incomplete derivations in first-order logic.

We use a one-and-a-half level syntax building on ideas from one-and-a-halfth order logic [GM08a]: variables of level one model variable symbols and can be quantified (we use variables of level one to model both type and term variables); variables of level two model 'holes'. This reflects informal practice in which instantiation of meta-variables (modelled here by variables of level two) captures object-level abstraction (modelled by variables of level one and their abstraction).

This paper is part of a larger project to develop nominal techniques in general, and in particular to study a multi-level syntax, in which capturing-substitution is explicitly represented, and using permutations to manage α-conversion in the presence of multiple levels of variable.

6.1. Related work

This paper builds on a conference version [GM08b]. This journal paper gives full proofs, and it also extends results as follows:

- The type language is now full first-order logic, rather than first-order logic without term-formers.[11]
- We have added to the syntax of our Curry-Howard system a theory of α-equivalence, following nominal terms and based on permutations in the style of [GP01, UPG04]. This is $(\mathbf{cong}\alpha)$, and more specifically, it is based on the permutation rule $(\mathbf{perm})$ from nominal algebra [GM07].
- We have given notions of β-reduction/proof-normalisation.

This paper 'just' studies a type system for nominal terms. Has this not been done before? Not in a way that helps us for constructing Curry-Howard for first-order logic. A sorting system for nominal terms from [UPG04] is not suitable; it is designed to construct abstract syntax and atoms (variables of level one in our terminology) have sort 'the sort of atoms', thus they do not populate other types as do the variables of level one in this paper. A typing system [FG07a] is not suitable, for two reasons:

- Types corresponded with propositional logic with quantifiers whereas here, we want first-order logic.
- Here, we want to represent $(\forall\mathbf{I})$ and $(\forall\mathbf{E})$ (Figure 2) so terms may λ-abstract over and be applied to type variables, and we require freshness for type variables. In [FG07a] type-variables do not inhabit terms in any way.

A body of research exists, aimed at providing formalisms for representing incomplete derivations:

Muñoz, Asperti. Of all the work we have seen in the literature, that by Muñoz [Muñ96] seems closest in spirit to ours. Muñoz works towards representing incomplete derivations in dependent types — a goal more ambitious than ours, of representing incomplete derivations of first-order logic, but not one which creates any essential difficulties aside from problems of scale and presentation. We believe that what Muñoz tries to do, and what we try to do, are essentially the same thing. The major difference, which we believe is an advantage of our approach, is our use of nominal terms — Muñoz uses de Bruijn indexes. The first sixty pages of [Muñ96] are devoted to discussing the problems, and solutions, of representing reduction in a λ-calculus with explicit substitutions based on de Bruijn indexes. Then 'meta-variables' (level two variables in our terminology) are introduced. These are essentially identical to our level two variables and admit a capturing substitution (see Definition 3.1.7 in [Muñ96]) — but they exist in the de Bruijn framework.

The Matita [ACTZ07] proof assistant, based on the Curry-Howard isomorphism, employs Muñoz's approach for representing unspecified proofs. It would be interesting future work to create a similar proof assistant based on our approach. We hope this

[11] Thanks to an anonymous reviewer for pointing this out.

could permit a higher-level style of programming in the same way that for example FreshOCaml or α-Prolog (also based on the nominal model of names and binding) are intended to allow a higher-level style of programming on datatypes with names [SP05, CU03]. This is future research.

Jojgov and Geuvers. Jojgov and Geuvers [GJ02] study incomplete derivations in higher-order logic (a popular logic for proof assistants). Their work may be seen as a generalisation of that of Muñoz, wherein they generalise the approach and re-establish the Curry-Howard isomorphism, but may also be viewed as a systemization of the techniques used in the Coq and Matita proof assistants.

They create o-HOL, which conservatively extends HOL with 'open' terms and 'open' (incomplete) proofs. 'Metavariables' (level two variables in our terminology) are represented in o-HOL syntax. Their treatment of α-conversion is not nominal, but neither is it quite 'traditional'. In particular it does capture a kind of capturing substitution which they, like us, call instantiation. See the definition of metavariable instantiation on page 546 (page 10 of the paper, just after Definition 11), which in simplified form reads like this:

$$\underline{n}[q]\{\underline{n}[y] := t\} = t[q/y]$$

Here $\underline{n}$ is a metavariable symbol, q and t are terms, and y is a(n ordinary, level one) variable. $\underline{n}[q]$ seems to correspond with what we might write as '$X[\text{-} \mapsto q]$' and the metavariable instantiation $\{\underline{n}[y] := t\}$ seems to correspond with what we might write as '$[y \mapsto \text{-}][X := t]$'. By this (rather ingenious) device Jojgov and Geuvers avoid many of the pitfalls described in [Bog02, Subsection 2.1 onwards], [Muñ97, Subsection 1.4.1], and [GL08, Introduction] of combining (in our terminology) variables of different levels.

The precise technical relationship between our term language and that of Jojgov and Geuvers is currently unclear and might be the topic of future work.

Miller and McBride. McBride's Oleg system [McB99] is a theorem proving environment, based on Luo's ECC, augmented with what McBride calls 'holes', which a Prolog programmer might recognise as 'logical variables' or 'existential variables'. Intuitively, an existential variable has operational or logical content 'please find a suitable value for me'.[12] McBride's syntax for a existential variable is $?x.p$ where p is a term. See Section 2.3 of [McB99] where rules are presented for searching for values for, and instantiating values for, existential variables.

If the existential variables represent a datatype of derivation-trees, then terms with existential variables represent incomplete derivations. There is no *a priori* proof-theory or reduction system for such a datatype built in to McBride's system (it is just a datatype).

McBride's system comes closer to our goals if we consider that a term containing a existential variable in McBride's system does in a suitable sense represent an 'incomplete inhabitant' of the type it populates. If that type is a type of truth-values then the term represents an 'incomplete inhabitant' of a type of truth-values, thus, an 'incomplete proof'. In fact our level two variables X are neither overtly existential nor universal. We

[12]The operational behaviour of a language with existential variables is typically to return an instantiation satisfying constraints on those variables, as in Prolog. Contrast this with functional or 'universal' variables $\lambda x.p$ with operational or logical content 'you can replace me with anything you like', as in function application in functional languages.

have given no operational semantics for instantiating them (note that we have studied instantiation of X — see Definition 50 and the subsequent results — we just have not internalised this to, for example, a derivation-instantiating strategy). However, level two variables X exist at top level and we, looking at the system from the outside, typically want to 'find some complete derivation', i.e. we want to instantiate X with *some* term of the right type. Thus for us, the X do intuitively possess an existential flavour. Then the difference between our work and McBride's is our 'nominal style' term language; McBride's treatment of existential variables and instantiation is based on 'traditional' α-equivalence and capture-avoiding substitution. Essentially, our work and McBride's are two complementary systems; we do not study existential variables for their own sake (though they do appear in our system to represent 'unknown derivations'!) — and McBride does not study instantiation. In that light, our argument is that for Curry-Howard for incomplete derivations we need logical variables *and* instantiation. Thus, it might be interesting to take the ideas about treating logical variables from McBride's research and combine them with our nominal treatment of level two variables in order to build a system in which our meta-level consideration of instantiating X can be internalised in, say, the tactics language of a theorem-prover. This is future work.

Miller's work [Mil92] also considers existential variables, by considering unification with a mixed (i.e. $\forall/\exists$) prefix. Normally, a unification problem takes the form $\exists x_1 \dots x_n.s_1 = t_1 \dots s_m = t_m$. Miller is interested in finding solutions to problems of the form $\mathsf{Q}x_1 \dots \mathsf{Q}x_n.s_1 = t_1 \dots s_m = t_m$, where Q may represent either a universal or an existential quantifier, and the terms, s_i, t_i are simply typed λ-terms. A universally quantified variable behaves like a constant within its scope, whereas an existentially quantified variable is available for substitution. Differing sequences of quantifiers modify the behaviour of substitutions applied to the equations. For example, $\forall_{\iota \Rightarrow \iota} f.\exists_\iota x.\forall_\iota w.[(fw) = x]$ does not have a solution, as the naming apart of bound variables and substitutions requires that any substitution for x will fail to obtain equality. The quantifier prefix constrains the behaviour of substitutions similarly to 'nominal style' freshness conditions. The existence of existential variables can be used to express 'incomplete derivations'. Again, the treatment of all variables is 'traditional', based on α-equivalence and capture-avoiding substitution.

Magnusson. ALF is a proof assistant based on Martin-Löf type theory [Mag93]. As far as we know this was the first to explicitly represent incomplete derivations. This is done using proof terms containing meta-variables ('place-holders' in ALF parlance). Each meta-variable is annotated with a 'local context', which is a list of variables upon which an expression to which the place-holder is refined, may depend. In addition, typing constraints on substituted expressions limit which expressions may be used to refine a meta-variable [Nor93].

The ALF approach has much in common with the Oleg approach of McBride [McB99], and with the approaches of Muñoz and Jojgov [Joj02].

Stoughton, Gunter and Rémy. Stoughton [Sto98] used a system capable of representing incomplete derivations for a programming language semantics animator, DOPS (Deterministic OPerational Semantics). DOPS uses a typed lambda calculus as a metalanguage, and neither non-determinism nor non-termination of the object language are

possible. Stoughton's representation of incomplete derivations uses this fact—an incomplete derivation tree is represented as a tree with branches missing coupled with a 'resumption'. The action of the resumption is uniquely determined by the evaluation relation of the object language in question, and extends the coupled tree with new branches, if possible, when executed.

Gunter and Rémy [GR93] appear to have first proposed using representations of incomplete derivations for animating programming language semantics. They provided a brief informal overview of their idea, but did not expand on it, nor did they implement a system [Sto98].

Other, distantly related work. We also briefly note contributions by Bognar [Bog02] and Hashimoto and Ohori [HO01]. Bognar's calculi treat terms with holes but their approach is essentially based on 'traditional' α-equivalence and λ-lifting, as can be seen for example in [Bog02, Definition 4.2.11]. Hashimoto and Ohori [HO01] admit capturing substitution and use a sophisticated type system to control it (and its interaction with capture-avoiding substitution which we referred to in the Introduction). There is no discussion of how suited these systems would be to a Curry-Howard correspondence.

6.2. Future work

This paper is part of a mathematical project to explore how multi-level systems — with capturing as well as capture-avoiding substitution — serve as a mathematical foundation for informal practice. In this paper we have seen how the ideas of two levels of variable, with freshness conditions, and permutations, can be used to as a formal syntax and operational semantics for a Curry-Howard for incomplete derivations.

We see four broad avenues for building on the system in this paper:

Types for multi-level calculi. The Lambda-Context Calculus [GL08, GL09] has levels of variables and an operational semantics, but it does not feature a nominal terms style α-equivalence based on permutations — in [GL08], the first author wrote about possible applications to modelling incomplete derivations. Two level λ-calculus has two levels of variable [GM09a]. This has λa and also a λX, substitution for X does not avoid capture by λa, and nominal terms style α-equivalence.

This paper would then be a rather powerful type system (more than Hindley-Milner for example) for the λX-free fragment of two level λ-calculus; we are reasonably confident this would extend to λX. That is for future work.

Implementation. We ask whether the ideas in this paper can be useful for the theory or practice of writing theorem provers. Can these ideas be refined and extended and applied to represent and program on intermediate proof-states in the kernel of a theorem-prover?

Nominal unification is known to be decidable [UPG04], and Fernández and Calvès have shown that nominal unification can be performed in polynomial time [CF07]. Further, Fernández and Gabbay [FG07b] provided sufficient conditions on terms for efficient, polynomial time, nominal rewriting. Thus, nominal terms have some good computational properties which we may be able to exploit. This need not be too large a job, since the kernel of a theorem-prover can be quite small (a few thousand lines of code).

Note that we have not developed the mathematics of our representation with efficiency in mind (for example, our term-reduction/derivation-normalisation system is small-step, and non-terminating). Improving the computational properties of our representation is future work. Note that this is not necessarily very difficult. For example the non-termination could be dealt with by introducing an explicit substitution. As is often the case, we trade mathematical simplicity for computational efficiency: this would add an extra term-former and lengthen some proofs, which is why we did not do it here (some further comment on this issue, follows below).

In principle, nominal terms may be useful for implementing tactics within a theorem prover. For instance, many tactics lift subterms of λ-terms into a new context. This operation is complex due to the possibility of inadvertently capturing variables. Employing similar techniques to those presented in this paper may allow a more straightforward and less error-prone implementation of complex tactics. It may also be interesting to consider one-and-a-halfth level terms as a language for communicating incomplete derivations between systems.

Extensions of the system. It should not be hard to change the types and/or the terms of our calculus to suit other logical systems. For example we can follow Muñoz and try to enrich types in the direction of a dependent type theory, attempting to develop the typing rules from Figure 1 into a dependent type theory similar to that of Martin-Löf [BN90]. This would be distinct from a dependent type theory with elements of nominal techniques [SS04], which treats atoms (variables of level one) as variable symbols.

Likewise, we can try to extend the system to other logics, with λ-abstraction for level two variables (following the two level λ-calculus of [GM09a] or the lambda context calculus of [GL08, GL09]), or we can extend with explicit rules for instantiating level two variables and with constructs internalising other aspects of proof-search like instantiation and backtracking, following the example of McBride [McB99].

Finally, we mentioned goal-directed proof theory in the Introduction [GO00]. This field of research is devoted to the study of reasoning from conclusions towards assumptions, and to our knowledge no Curry-Howard correspondence has been developed for it.[13] Thus, it seems a good avenue for future work to apply the techniques used in this paper for a sequent system, to the goal-directed one.

Improving the system. Our calculus has no strong normalisation result; $(\lambda b.((\lambda a.c)a))b$ (which we also write $c[a \mapsto a][b \mapsto b]$, using the notation of Definition 5) can reduce indefinitely in the empty freshness context, because the β-redexes can 'skip over' one another, and the term is typable. For the same reason, strong normalisation is not preserved for the natural translation of the untyped λ-calculus to the (level 1) fragment of our language; consider the λ-term $(\lambda y.((\lambda x.z)x))y$.

There is a current of research devoted to creating calculi for computation which are computationally efficient and also are amenable to mathematical treatment; according to those criteria, it is clear that our calculus could be improved. Note however that even if computational efficiency is not a concern, we can be interested in elegant proofs (see below) and we can consider efficiency a separate concern, much as one might program in the λ-calculus but execute on an abstract machine. In particular, one can imagine a

[13] Dov Gabbay, private communication.

version of this calculus in which level 2 variables are annotated with parallel substitutions so that we can outlaw the 'skipping' behaviour mentioned above. This is future work.

So we should now take a critical look at the proofs in this paper, and this brings us to the main issue with this paper. The proofs are quite long, and rather complex: why?

1. We do not apply the familiar parallel reduction proof-method to prove confluence (attributed to Tait and Martin-Löf and documented, for example, in Barendregt's book [Bar84]). Instead, we use a proof-method based on a canonical form (Definition 57) and a diagram (Theorem 62). We developed this method for the λ-context calculus [GL08, GL09]. At the time we thought it was original but in fact it is close to one discovered by Takahashi [Tak95].[14]
   This proof-method has much to commend it; to quote Takahashi commenting on her application of it to prove the theorem of confluence of the untyped λ-calculus, it is "rigorous, direct, and perhaps the shortest among all the known proofs of the theorem". Our proof is longer, but this is just because it is applied to the more complex syntax of this paper.
   We consider the proof-method used in this paper to be a feature, not a bug.
   We now come to what we consider the main issue with our calculus:
2. The notion of α-equivalence, which is inherited from nominal terms [UPG04], is a source of complexity. First, α-equivalence depends on a freshness context and is not inherent to terms. This means that we cannot 'just quotient by α-equivalence', because the relation over which we would like to quotient is dependent on the freshness context. The obvious solution is to fix one the freshness context, but this is not possible because for confluence we may specifically wish to 'freshly extend it' (Definition 51) with extra fresh atoms. For similar reasons, in the typing rules we introduce a 'freshly extending' rule $(\mathbf{Tfr})$ (Figure 1).
   This combination of factors puts two non-syntax-directed rules in the theory of this paper, $(\mathbf{Tfr})$ and $(\mathbf{cong}\alpha)$, and we must argue up to fresh extensions of the freshness context. We trace most of the complexity of this paper to these factors (for example, the results in Subsection 5.1 are all devoted to controlling $(\mathbf{Tfr})$).

Since this paper was written, and as a response to these issues, we have investigated *permissive* nominal terms and also *semantic* nominal terms [DGM09, GM09c, GM09b]. These do away with freshness contexts and are based on the idea of infinite and *coinfinite* sets of atoms (a set is coinfinite when its sets complement is infinite). Because sets are coinfinite, it is always guaranteed that there is a fresh atom, so that we can 'just quotient' by α-equivalence and there is no need for freshly extended contexts, nor for rules to handle these things. Developing this further is future research.

[14] Takahashi's proof still defines a parallel reduction relation, whereas the proof we give uses just multiple β-reduction, but the underlying ideas of the two proofs are very similar.

References

[ACTZ07] Andrea Asperti, Claudo Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. *Crafting a Proof Assistant*, pages 18–32. Lecture Notes in Computer Science. Springer, 2007.

[Bar84] Henk P. Barendregt. *The Lambda Calculus: its Syntax and Semantics (revised ed.)*. North-Holland, 1984.

[BN90] Jan M. Smith Bengt Nordstrom, Kent Petersson. *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science*. Clarendon Press, Oxford, 1990. Also online at `http://www.cs.chalmers.se/Cs/Research/Logic/book/`.

[Bog02] Mirna Bognar. *Contexts in Lambda Calculus*. PhD thesis, Vrije Universiteit Amsterdam, 2002.

[Bru96] Norbert Brunner. 75 years of independence proofs by Fraenkel-Mostowski permutation models. *Mathematica Japonica*, 43:177–199, 1996.

[CF07] Christophe Calvès and Maribel Fernández. Implementing nominal unification. *Electronic Notes in Theoretical Computer Science*, 176(1):25–37, 2007.

[CU03] J. Cheney and C. Urban. System description: Alpha-Prolog, a fresh approach to logic programming modulo alpha-equivalence. In *UNIF'03*, pages 15–19. Universidad Politecnica de Valencia, 2003.

[CU04] James Cheney and Christian Urban. Alpha-prolog: A logic programming language with names, binding and alpha-equivalence. In Bart Demoen and Vladimir Lifschitz, editors, *Proc. of the 20th Int'l Conf. on Logic Programming (ICLP 2004)*, number 3132 in Lecture Notes in Computer Science, pages 269–283. Springer, 2004.

[dB80] N.G. de Bruijn. A survey of the project AUTOMATH. In Hindley and Seldin, editors, *To H.B.Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[DGM09] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. In *CILC, 24th Italian Conference on Computational Logic*, 2009.

[FG07a] Maribel Fernández and Murdoch J. Gabbay. Curry-style types for nominal terms. In *Types for Proofs and Programs (proceedings of TYPES'06)*, volume 4502 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2007.

[FG07b] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting (journal version). *Information and Computation*, 205(6):917–965, 2007.

[Gab07] Murdoch J. Gabbay. Fresh Logic. *Journal of Applied Logic*, 5(2):356–387, June 2007.

[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.

[GL08] Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus. *Electronic Notes in Theoretical Computer Science*, 196:19–35, 2008.

[GL09] Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus (extended version). *Information and computation*, 2009.

[GM93] Michael J. C. Gordon and Thomas F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[GM07] 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 *Lecture Notes in Computer Science*, pages 162–176. Springer, 2007.

[GM08a] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.

[GM08b] Murdoch J. Gabbay and Dominic P. Mulligan. One-and-a-halfth Order Terms: Curry-Howard for Incomplete Derivations. In *Proceedings of 15th Workshop on Logic, Language and Information in Computation (WoLLIC 2008)*, volume 5110 of *Lecture Notes in Artificial Intelligence*, pages 180–194, 2008.

[GM09a] Murdoch J. Gabbay and Dominic P. Mulligan. Two-and-a-halfth Order Lambda-calculus. *Electronic Notes in Theoretical Computer Science*, 2009. To appear.

[GM09b] Murdoch J. Gabbay and Dominic P. Mulligan. Semantic nominal terms. In *TAASN*, 2009.

[GM09c] Murdoch J. Gabbay and Dominic P. Mulligan. Universal algebra over lambda-terms and nominal terms: the connection in logic between nominal techniques and higher-order variables. *LFMTP'09: Proceedings of the Fourth International Workshop on Logical Frameworks and Meta-Languages*, 64–73, 2009.

[GO00] Dov Gabbay and Nicola Olivetti. *Goal Directed Algorithmic Proof Theory*, volume 21 of *Applied Logic Series*. Kluwer Academic, 2000.

[GP01] 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, 2001.

[GR93] Carl A. Gunter and Didier Rémy. A proof theoretic assessment of runtime type errors. Technical Report 11261-921230-43TM, AT&T Bell Laboratories, 1993.

[HO01] Masatomo Hashimoto and Atsushi Ohori. A typed context calculus. *Theoretical Computer Science*, 266(1-2):249–272, 2001.

[Joj02] Gueorgui I. Jojgov. Holes with binding power. In *TYPES*, volume 2646 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2002.

[Mag93] Lena Magnusson. Refinement and local undo in the interactive proof assistant ALF. In *Informal Proceedings of the 1993 Workshop on Types for Proofs and Programs*, pages 237–253, 1993.

[Mat07] Aad Mathijssen. *Logical Calculi for Reasoning with Binding*. PhD thesis, Technische Universiteit Eindhoven, 2007.

[McB99] Conor McBride. *Dependently Typed Functional Programs and their Proofs*. PhD thesis, University of Edinburgh, 1999.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14(4):321–358, 1992.

[MP93] James McKinna and Randy Pollack. Pure Type Systems formalized. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications (TLCA'93)*, number 664 in LNCS, pages 289–305. Springer-Verlag, March 1993.

[Muñ96] César Muñoz. Proof representation in type theory: State of the art. In *Proceedings of the 22nd Latin American Conference on Informatics (CLEI Panel 96)*, 1996.

[Muñ97] César Muñoz. *A Calculus of Explicit Substitutions for Incomplete Proof Representation in Type Theory*. PhD thesis, INRIA Rocqencourt, Projet Coq, 1997.

[Nor93] Bengt Nordström. The ALF proof editor. In *Proceedings of the Workshop on Types for Proofs and Programs (TYPES 1993)*, pages 253–266, 1993.

[Pau89] Lawrence C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[PCW05] Iman Hafiz Poernomo, John Newsome Crossley, and Martin Wirsing. *Adapting Proofs-as-Programs: The Curry–Howard Protocol*. Number XII in Monographs in Computer Science. Springer, 2005.

[SP05] Mark R. Shinwell and Andrew M. Pitts. Fresh objective Caml user manual. Technical Report UCAM-CL-TR-621, University of Cambridge, 2005.

[SS04] Ulrich Schöpp and Ian Stark. A Dependent Type Theory with Names and Binding. In *CSL*, volume 3210 of *Lecture Notes in Computer Science*, pages 235–249, 2004.

[Sto98] Alley Stoughton. An operational semantics framework supporting the incremental construction of derivation trees. In *Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II)*, volume 10 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science B. V., 1998.

[Tak95] Makato Takahashi. Parallel reductions in lambda-calculus. *Information and Computation*, 1(118):120–127, 1995.

[Tru94] John Truss. Permutations and the axiom of choice. In H.D.Macpherson R.Kaye, editor, *Automorphisms of first order structures*, pages 131–152. OUP, 1994.

[UPG04] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.

[US06] Pawel Urzyczyn and Morten Sørensen. *Lectures on the Curry-Howard isomorphism*, volume 149 of *Studies in Logic*. Elsevier, 2006.

A. ZFA equivariance

Equivariance is well-studied in model theory [Tru94, Bru96]; in a sentence, equivariance states that truth is invariant under permuting atoms, where an 'atom' is an element with no structure, aside from being equal to itself. Variable symbols are an example of something 'in nature' which can naturally be viewed as atomic.

Permutations were used in computer science already in [MP93]. In [GP01] equivariance was noted as a meta-mathematical fact of Fraenkel-Mostowski set theory and used to deduce the some/any property of the 'NEW'-quantifier and thus derive new inductive reasoning principles for syntax-with-binding. In [Gab07] equivariance was used in a discursive proof (that is, in the informal but rigorous mathematical argument of a published paper) to 'rename variable symbols' without losing the inductive hypothesis. In [GM08a, Appendix A], equivariance was explicitly stated and proved as a meta-mathematical principle of Zermelo-Fraenkel set theory with atoms (a more general, and more standard, foundational system than Fraenkel-Mostowski set theory, which does not have its 'finite support' property; see [GP01] for details).

We will now briefly recap on [GM08a, Appendix A], but instead of axiomatic set theory we will use a 'lighter' argument based on von Neumann cumulative hierarchies.

Fix a collection of **atoms** (recall the sets of level one variables from Subsection 2.1). Write this set $\mathbb{A}$. Define a **cumulative hierarchy** by:

$$\mathcal{U}_0 = \mathbb{A} \qquad \mathcal{U}_\alpha = \bigcup_{\beta<\alpha} \mathcal{P}(\mathcal{U}_\beta) \cup \mathcal{U}_\beta \qquad \mathcal{U} = \bigcup_\alpha \mathcal{U}_\alpha$$

Here $\mathcal{P}$ denotes the powerset and α and β range over ordinals. Intuitively $\mathcal{U}$ is a collection of well-founded (possibly infinitely-branching) trees, which is constructed by transfinite induction starting from trees consisting of a single node labelled with an element of $\mathbb{A}$. This is the standard model of Zermelo-Fraenkel set theory with atoms. It is a sufficently rich structure to build all of usual mathematics, and it has been argued that this is the foundational mathematical universe most commonly used in 'practical mathematics' (rather than Zermelo-Fraenkel set theory, without atoms).

A permutation Π is a bijection on $\mathbb{A}$ (we do not care if $\{a \mid \Pi(a) \neq a\}$ is finite or not). We define a **permutation action** on $\mathcal{U}$ by:

$$\Pi \cdot a = \Pi(a) \qquad \Pi \cdot U = \{\Pi \cdot u \mid u \in U\}$$

Here $U \in \mathcal{U} \setminus \mathbb{A}$ and $a \in \mathbb{A}$.

Now let terms T and predicates Φ be inductively defined by:

$$T ::= \mathcal{X} \mid \mathbb{A}$$
$$\Phi ::= \Phi \Rightarrow \Phi \mid \bot \mid \forall \mathcal{X}.\Phi \mid T = T \mid T \in T$$

Here $\mathcal{X}$ are variable symbols. This is the language of first-order logic. This happens to be very similar to the language of types investigated in this paper (because we wanted to model an *interesting* logic), but Φ here is intended to be a model of the informal but rigorous mathematical arguments written in English in this paper. In other words, the Φ model the statements of lemmas and theorems in this paper, and the theory of truth for Φ over $\mathcal{U}$ is sufficiently rich to soundly capture the mathematics in those lemmas and theorems.

It then suffices to observe the following:

Theorem 63. *$\Phi(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ is true of $\mathcal{U}$, if and only if $\Phi(\Pi \cdot \mathcal{X}_1, \ldots, \Pi \cdot \mathcal{X}_n)$ is true.*

Proof. By induction on Φ. It suffices to observe that:

- $u \in u'$ if and only if $\Pi \cdot u \in \Pi \cdot u'$
- $u = u'$ if and only if $\Pi \cdot u = \Pi \cdot u'$.
- $\Pi \cdot \mathbb{A} = \mathbb{A}$.

□

When in the body of this paper we permute variable symbols, we use Theorem 63 to retain the inductive hypothesis.