One-and-a-halfth order terms: Curry-Howard and incomplete derivations

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

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

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 an 'incomplete derivation', with 'holes'.
This is natural in informal practice; the challenge is to formalise it. To this end we use a one-and-a-halfth order technique based on nominal terms, with *two* levels of variable. Predicates are types, derivations are terms — and the two levels of variable are respectively the assumptions and the 'holes' of an incomplete derivation.

1 Introduction

The Curry-Howard correspondence [US06,PCW05] connects logic with typed λ-calculus: predicates are types; derivations are terms; discharge is λ-abstraction; modus ponens is application; β-reduction is proof-normalisation. For example,[3]

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

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 λ-calculus supports 'forwards' reasoning, where we plug together complete derivations to form larger ones. However, we may wish to reason 'backwards': We start from an incomplete derivation of the desired conclusion and we work backwards to construct a derivation. Then we may have 'half a derivation', like as below left with a 'hole' called X:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\, X\\ \mathsf{B}\end{array} \qquad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,a \qquad \dfrac{\dfrac{\dfrac{[\mathsf{A}]^a \quad \begin{array}{c}\vdots\, X\\ \mathsf{A}\Rightarrow\mathsf{B}\end{array}}{\mathsf{B}} \qquad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,a \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}[\mathsf{A}]^a\\ \vdots\, X\\ \mathsf{A}\Rightarrow\mathsf{B}\end{array}}{\mathsf{B}} \qquad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,a$$

[3] We are grateful to Jojgov for the examples in his paper [Joj02]. We thank Iman Poernomo and two anonymous referees for helpful comments.

Here, λ-calculus syntax is less helpful. X corresponds with qa in the complete derivation, so (being straighforward about it) the incomplete derivation corresponds with '$\lambda a.((pa)X)$'. But X is under a λ-binder and should be *instantiated*; substituted for without avoiding capture. This is impossible within the λ-calculus. Most interesting logics are undecidable so theorem-proving is often interactive (like AUTOMATH [dB80] and its many descendents). This leads us to study calculi tailored to represent incomplete derivations.

In this paper we build on previous work by the first author and others on nominal techniques [GP01] and specifically nominal terms [UPG04] and one-and-a-halfth order logic [GM07]. These were designed specifically to study binding (in unification up to α-equivalence, and derivation-schema in first-order logic respectively). They feature two levels of variables, freshness conditions, and permutations; details are in this paper, and in the work cited above. In this paper we extend this pallette of ideas to represent binding in incomplete derivations. We are reasonably ambitious in our choice of logic for which to represent incomplete derivations: we will consider first-order predicate logic; this is a significantly more complex target than propositional logic, and it leads to quite a rich syntax.

In the style of Miller [Mil92], McBride's OLEG system [McB99], and a collection of λ-calculi by Bognar [Bog02], we can represent X by fa where f is a 'normal' variable, perhaps recording in a context f should be instantiated; $f \vdash \lambda a.((pa)fa)$. A problem from our point of view is, for example, that the representation of the incomplete derivation above left is identical to that of distinct incomplete derivations above centre and above right, in which X is refined.

Another approach is to extend the λ-calculus with hereditarily parameterised meta-variables (hereditarily, since the parameters may themselves have 'holes'). This path is taken by Jojgov [Joj02], and for a non-hereditary notion of parameters, by Severi and Poll [SP94], and Bloo *et al* [BKLN02].

Following one-and-a-halfth order logic [GM07] we propose an approach based on *nominal terms* [UPG04]. Nominal terms have *atoms* $a, b, c, \ldots$ and *unknowns* $X, Y, Z, \ldots$. Crucially, substitution of unknowns does not avoid capture by atoms, and we reason on what unknowns do not depend on, rather than using parameters to record what they might depend on ($a\#X$ versus fa; see 'freshness' below). The first author, Urban, Pitts, and Cheney amongst others have argued in favour this approach [UPG04,FG07,Mat07,Pit02,Che05].[4]

We further this argument and show that atoms, unknowns and freshness model assumptions, holes, and discharge in incomplete derivations.

Consider an example; definitions are in the body of the paper:

[4] In one-and-a-halfth order logic, unknowns populate predicates, and model predicate meta-variables in derivation schemas. Here, unknowns are used differently, to model holes in terms representing derivations. The common 'one-and-a-halfth order' idea — also present, implicitly, in Curry-style types for nominal terms [FG06] — is that atoms can be *variables*. This is different from [UPG04] where atoms populate a sort of atoms and are variable *symbols*.

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

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

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-halfth order Curry-Howard. Note that:

– *Assumptions* are represented by atoms. Types are predicates assumed.

– *Incomplete* parts of the derivation, or (using terminology from the theorem-proving community) *subgoals*, are represented by unknowns. 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'.

This paper is 'just' about 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 have sort 'the sort of atoms'. A typing system [FG06] is not suitable; types corresponded to propositional logic with quantifiers whereas here, we want first-order logic and; we also 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.

Some words on what this paper is not: it is not proof-search [PR05,MS06]. We study binding in incomplete derivations, but not the act of stepping from one derivation to another. We also give no semantics to our syntax: There is no denotational semantics (Scott domains spring to mind; we would require an extended version, perhaps like FM (nominal) domain theory [SP05]). There is not even an operational semantics (reduction of derivations), though we do plan this for a later paper; see the Conclusions.

2 Terms, types, and Natural Deduction

2.1 Terms and types

We give definitions, then discuss examples in Remark 10 and Subsection 2.2.

Fix disjoint countably infinite sets of **atoms** $\mathbb{A}$ and **unknowns**. We let $a, b, c, d, \ldots$ range over atoms. We use a **permutative convention**; they range *permutatively* over atoms, so for example 'a and b' means 'a pair of two *distinct* atoms'. Similarly we let $X, Y, Z, \ldots$ range permutatively over unknowns.[5]

Fix **atomic type-formers** $\mathsf{P}, \mathsf{Q}, \mathsf{R}$, to each of which is associated an arity ar(-) which is a nonnegative integer (0, 1, 2, ...).

[5] When we write 'x' and 'y' we intend them to be distinct symbols; for example '$\lambda x.y$' is always taken to be different from '$\lambda x.x$'; likewise '$x = y$' is different syntax than '$x = x$'. This is distinct from the denotation of x being equal to that of y.

Definition 1 Let **types** be: $\phi, \psi, \xi \ ::= \ \bot \mid \phi \Rightarrow \phi \mid \mathsf{P}(\overbrace{a, \ \ldots}^{ar(\mathsf{P}) \text{ var'bles}}) \mid \forall a.\phi$.

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

We equate types up to $\forall$-bound atoms. We write $\equiv$ for syntactic identity of types. We write $\phi[a := b]$ for the usual capture-avoiding substitution action of b for a. Implication associates to the right; for example $\phi \Rightarrow \psi \Rightarrow \xi \equiv \phi \Rightarrow (\psi \Rightarrow \xi)$.

Intuitively, types are first-order logic with the trivial term-language (a logic whose terms are just variable symbols).

Definition 2 Define the **free atoms** of ϕ as standard by:

$$fa(\mathsf{P}(a,\ldots))=\{a,\ldots\} \quad fa(\phi{\Rightarrow}\psi)=fa(\phi)\cup fa(\psi) \quad fa(\bot)=\emptyset \quad fa(\forall a.\phi) = fa(\phi)\setminus\{a\}$$

Definition 3 Let **terms** be: $r, s, t, \ldots \ ::= \ a \mid X \mid \lambda a.r \mid r'r \mid \mathsf{xf}(r)$.

Following [GL08] we identify terms up to α-equivalence of a in $\lambda a.r$ provided that r mentions no unknowns.[6] We write $\equiv$ for syntactic equivalence of terms.

For example $\lambda a.a \equiv \lambda b.b$ and $\lambda a.X \not\equiv \lambda b.X$. 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.

Definition 4 A **type assignment** is a pair of the form $a : \phi$, 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 $\Gamma \cup \{a : \phi\}$. We use this convention later without comment. Intuitively, $a : \phi$ means 'a has type ϕ'; $a : *$ means 'a is a type variable'; $X : \phi$ means 'X has type ϕ'.

Remark 5 We use the same syntactic class (atoms) 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 atoms that can have types ($a : \phi \in \Gamma$), and atoms 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 significantly shorter.

Definition 6 Call a pair $a\#r$ of an atom 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.

[6] This allows us to rename bound atoms in 'normal' syntax — the part without unknowns — but it stops short of a nominal terms style α-equivalence with unknowns based on permutations. For our purposes in this paper, what we give ourselves is enough. See the Conclusions.

Definition 7 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**.

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

Definition 8 – 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 unknowns, 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 9 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 - $\vdash$ - is *not* derivable we write - $\nvdash$ -.
– 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**.

Remark 10 We compare the rules in Figures 1 and 2:

– Compare (**T**$\bot$**E**) with ($\bot$**E**). 'xf' stands for *ex falsum*. (**T**$\bot$**E**) corresponds with ($\bot$**E**) in a standard way. No surprises here.
– Compare (**T**$\Rightarrow$**I**) with ($\Rightarrow$**I**). (**T**$\Rightarrow$**I**) does not discharge $a : \phi$ because r may contain an unknown X. We intend X to be instantiated to t which (because instantiation need not avoid capture) may mention a; see Definition 16. We remember $a : \phi$ in the typing context so that we can use it to build t, if we like. We can mimic ($\Rightarrow$**I**) using (**T**$\Rightarrow$**I**) and (**Tfr**).
– (**Tfr**) is an explicit discharge rule. It connects $b\#r$, which we can read as 'b will discharged in the (possibly incomplete) derivation represented by r' with actual discharge of b; after discharge, we cannot use b to construct any further derivations. As we just argued above, in the presence of unknowns it is convenient to separate these two notions.
– Compare (**T**$\forall$**I**) with ($\forall$**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 unknowns and freshness assumptions on them.
– Compare (**a#b**), (**a#b′**), and (**a#b″**). (**a#b**) and (**a#b″**) are as in [UPG04]; distinct atoms are fresh. In (**a#b′**) we account for the *type* of b. For example:

$$a : \mathsf{P}(c),\ X : \mathsf{P}(c),\ c : *;\ a\#X \vdash a\#X$$
$$a{:}\mathsf{P}(c),\ X{:}\mathsf{P}(c),\ c{:}*;\ a\#X \nvdash c\#X \qquad a{:}\mathsf{P}(c),\ X{:}\mathsf{P}(c),\ c{:}*;\ a\#X \nvdash c\#a$$

2.2 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 '$\mathit{\Gamma};\emptyset\vdash r$' as '$\mathit{\Gamma}\vdash r$'. Write $\mathit{\Gamma}$ for $a:\mathsf{A},\ p:\mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C},\ q:\mathsf{A}\Rightarrow\mathsf{B}$:

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

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

Derivations of $\mathit{\Gamma}\vdash a\#\lambda a.((pa)qa)$ and $\mathit{\Gamma},\ X:\mathsf{B}\vdash a\#\lambda a.((pa)X)$ are elided.[7]

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

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

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

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

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

Derivations of freshnesses are elided.

[7] The $(\mathbf{Tfr})$ in the second derivation is ill-advised, in the sense that intuitively we cannot then instantiate X to qa; we might prefer to conclude the derivation with $(\mathbf{T{\Rightarrow}I})$. We leave this kind of choice to a derivation search algorithm, or we might favour a 'safe' variant of $(\mathbf{Tfr})$ which insists r be closed (that r mention no unknowns).

2.3 Natural Deduction

We outline Natural Deduction and prove forms of soundness and completeness.

Definition 11 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 12 *If $\Phi \vdash \psi$ and $\Phi \subseteq \Phi'$ then $\Phi' \vdash \psi$.*

Definition 13 Call $\Gamma; \Delta \vdash r$ **closed** when $\Delta = \emptyset$ and Γ mentions no unknowns. Recall that we write '$\Gamma; \emptyset \vdash r$' as '$\Gamma \vdash r$'.

Theorem 14 *Suppose $\Gamma \vdash r$ is closed and suppose $\Gamma \vdash r : \phi$ is derivable. Then $important(\Gamma \vdash r) \vdash \phi$ (Definition 9) is derivable in Natural Deduction. (Proof in the Appendix.)*

Theorem 15 *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 in the Appendix.)*

2.4 Admissible rules

Definition 16 Define a **substitution** action $r[X := t]$ by:

$$a[X := t] \equiv a \quad X[X := t] \equiv t \quad Y[X := t] \equiv Y \quad \mathsf{xf}(r)[X := t] \equiv \mathsf{xf}(r[X := t])$$
$$(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t]) \quad (r'r)[X := t] \equiv (r'[X := t])(r[X := t])$$

Write $unkn(\Gamma)$ for the unknowns mentioned in Γ. Figure 3 presents two kinds of weakening and a form of Cut.

Theorem 17 (**WeakX**) *is **admissible** (if the sequents above the line are derivable, so are those below).*

Proof. By a routine induction on derivations.

(**Weaka**) *states* that the atom is fresh for the incomplete parts in the derivation:

Theorem 18 (**Weaka**) *is an admissible rule. (Proof in the Appendix.)*

Lemma 19 states how instantiating unknowns is sound:

Lemma 19 *Suppose that:*

[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.

– $\Gamma, X : \psi; \Delta \vdash r : \phi$ *and* $\Gamma; \Delta \vdash t : \psi$.
– $\Gamma; \Delta' \vdash a\#t$ *for every* $a\#X \in \Delta$.

Then:

– $\Gamma; \Delta' \vdash r[X := t] : \phi$.
– $\Gamma, X : \psi; \Delta \vdash a\#r$ *implies* $\Gamma; \Delta' \vdash a\#r[X := t]$, *for every* a.

(Proof in the Appendix.)

Cut in natural deduction is no more than 'plugging the conclusion of one derivation into the assumption(s) of another'. However, now assumptions may be holes in incomplete derivations and we can 'plug' in a capturing manner. The rule (**Cut**) specifies that operation, and from Lemma 19 we have:

Theorem 20 (**Cut**) *is an admissible rule.*

3 Derivation-search (sketch)

If by 'backwards' reasoning we mean 'reasoning from conclusion towards assumptions' then the machinery so far is sufficient. To mix 'forwards' with 'backwards' reasoning we may need a little more; consider an incomplete derivation of $\mathsf{A},\ \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} \vdash \mathsf{B}$ (cf. (2) in Subsection 2.2):

$$\dfrac{\begin{matrix}\dfrac{\mathsf{A} \qquad \dfrac{\forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c))}{\mathsf{A} \Rightarrow \mathsf{P}(c)}}{\mathsf{P}(c)} \\ \vdots \end{matrix} \qquad\qquad \begin{matrix}(\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} \\ \vdots\end{matrix}}{\mathsf{B}} \tag{3}$$

Our syntax does not represent this as a single term-in-context because the 'hole' is not at the leaf of the derivation. We can represent this incomplete derivation as a set of sequents, all sharing the same typing and freshness context. Following theorem-provers and unification algorithms we present this as a set of *goals*, in rewriting style; the rewrites below can easily be converted into derivation trees:

Definition 21 Let Ξ range over finite sets of typings $r : \phi$, $a : *$, and freshnesses $a\#r$. We may call $A \in \Xi$ a **goal** and we may call Ξ a **goal set**.

$A \in \Xi$ has intuition 'we know A' — not 'we want to prove A' — but if A mentions an unknown X then what we know is incomplete and we would like to complete it, i.e. prove it. To derive ϕ from $\Gamma; \Delta$ we start rewriting from $X : \phi, \Gamma, \Delta$ for X not appearing in Γ or Δ, and we try to instantiate X. We can declare success when we arrive at a goal state of the form $\Xi,\ r : \phi$ such that $\Gamma; \Delta \vdash r : \phi$.

For example to prove B from $\mathsf{A},\ \forall a.(\mathsf{A} \Rightarrow \mathsf{P}(a)),\ (\forall a.\mathsf{P}(a)) \Rightarrow \mathsf{B}$ we can start with $X : \mathsf{B},\ a : \mathsf{A},\ p : \forall a.(\mathsf{A} \Rightarrow \mathsf{P}(a)),\ q : (\forall a.\mathsf{P}(a)) \Rightarrow \mathsf{B}$ and rewrite as follows (we may drop types to save space):

$$
\begin{aligned}
&X : \mathsf{B},\ a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} \\
&\qquad \xrightarrow{(\mathbf{Weaka})} X : \mathsf{B},\ a,\ p,\ q,\ c : * \\
&\qquad \xrightarrow{(\mathbf{T\forall E})} X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc : \mathsf{A} \Rightarrow \mathsf{P}(c) \\
&\qquad \xrightarrow{(\mathbf{T{\Rightarrow}E})} X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc,\ pca : \mathsf{P}(c) \\
&\qquad \xrightarrow{(\mathbf{T\forall I})} X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc,\ pca,\ \lambda c.(pca) : \forall c.\mathsf{P}(c) \\
&\qquad \xrightarrow{(\mathbf{T{\Rightarrow}E})} X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc,\ pca,\ \lambda c.(pca),\ q(\lambda c.(pca)) : B \\
&\qquad \xrightarrow{(\mathbf{Cut})} q(\lambda c.(pca)) : B,\ a,\ p,\ q,\ c,\ pc,\ pca,\ \lambda c.(pca)
\end{aligned}
$$

We read off $q(\lambda c.(pca))$ as our result. (3) is represented by the third line above: $X : \mathsf{B},\ a,\ p,\ q,\ pc,\ pca : \mathsf{P}(c)$.

The following series of rewrites generates the derivation in (1) from the Introduction, also discussed in Subsection 2.2 ($\longrightarrow^*$ is multiple rewrites):

$$
\begin{aligned}
&X : \mathsf{A} \Rightarrow \mathsf{C},\ p : \mathsf{A} \Rightarrow \mathsf{B},\ q : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C} \\
&\qquad \xrightarrow{(\mathbf{Weaka})} X : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a : \mathsf{A},\ a\#X \\
&\qquad \xrightarrow{(\mathbf{WeakX})} X : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ a\#X,\ X' : \mathsf{C} \\
&\qquad \xrightarrow{(\mathbf{T{\Rightarrow}I})} X : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ a\#X,\ X',\ \lambda a.X' : \mathsf{A} \Rightarrow \mathsf{C} \\
&\qquad \xrightarrow{(\mathbf{Cut})} \lambda a.X' : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ X' \\
&\qquad \xrightarrow{(\mathbf{T{\Rightarrow}E})}{}^{*} \lambda a.X' : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ X',\ pa : \mathsf{B},\ qa : B \Rightarrow C,\ (pa)qa : C \\
&\qquad \xrightarrow{(\mathbf{Cut})} \lambda a.((pa)qa) : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ pa,\ qa,\ (pa)qa
\end{aligned}
$$

4 Conclusions

We have seen how nominal terms, with a typing system, can model 'incomplete derivations' in first-order logic. We use a 'one-and-a-halfth order' syntax, building on ideas from nominal terms and one-and-a-halfth order logic: atoms model variable symbols and can be quantified (we use atoms to model both type and term variables); unknowns model 'holes' in the derivation. This directly reflects informal practice, in which holes in incomplete derivations are instantiated (substituted with capture).

We have tested our system on examples. We have shown the fragment without unknowns is sound and complete with respect to 'normal' derivations (Subsection 2.3). We have shown instantiating unknowns is sound, and explored what weakening means in the presence of the two levels of variable (Subsection 2.4).

This paper is part of a larger project which we expect to be a fruitful source of research. In roughly decreasing order of certainty, we envisage the following:

Curry-Howard supposes normalisation of derivations — this translates to an operational semantics for terms (Definition 3). This has to be more than 'remove all β-reducts' because, for example, the β-reduct in $(\lambda a.X)Y$ cannot be reduced. To address this, an investigation into *two-and-a-halfth order λ-calculus* is ongoing. 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-and-a-halfth order λ-calculus; we are reasonably confident this would extend to λX.

The rewrite system alluded to in Section 3 can be viewed as an independent system and studied. On that topic, we can ask whether the ideas in this paper can be useful for the theory or practice of writing theorem provers. Perhaps the representation itself will be useful, but nominal unification is known to be decidable [UPG04]; thus, nominal terms have some good computational properties which we may be able to exploit. Given the scale and complexity of modern theorem-provers, answers to such questions may take some time to emerge — but the situation is also far from hopeless, since in the first instance only the prover's 'kernel' is involved.

Indeed, we can simplify the types to propositional logic (simple types; we drop the predicate part) and attempt to develop the rewrite system into a unification algorithm *à la* Huet [Hue02]. We can also try to enrich types in the direction of a dependent type theory and attempt to develop the typing rules from Figure 1 into a dependent type theory *på samma* Martin-Löf [NPS90]. This would be distinct from a dependent type theory with elements of nominal techniques [SS04], which treats atoms as variable symbols.

References

BKLN02. Roel Bloo, Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. Parameters in pure type systems. In *LATIN*, pages 371–385. Springer, 2002.

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

Che05. James Cheney. Nominal logic and abstract syntax. *SIGACT News (logic column 14)*, 36(4):47–69, 2005.

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.

FG06. Maribel Fernández and Murdoch J. Gabbay. Curry-style types for nominal rewriting. TYPES'06, 2006.

FG07. M. Fernández and M. J. Gabbay. Nominal rewriting. *Information and Computation*, 205:917–965, 2007.

GL08. Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus. *ENTCS*, 196:19–35, 2008.

GM07. Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. *Journal of Logic and Computation*, November 2007. Available online.

GP01. Murdoch J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2001.

Hue02. Gérard Huet. Higher order unification 30 years later. In *TPHOL 2002*, number 2410 in LNCS, pages 3–12, 2002.

Joj02. Gueorgui I. Jojgov. Holes with binding power. In *TYPES*, volume 2646 of *LNCS*, pages 162–181. Springer, 2002.

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.

MS06. Dale Miller and Alexis Saurin. A game semantics for proof search: preliminary results. In *MFPS XXI*, volume 155 of *ENTCS*, pages 543–563, 2006.

NPS90. B. Nordstrom, K. Petersson, and J. M. Smith. *Programming in Martin-Lof's Type Theory*, volume 7 of *Int'l Series of Monographs on Computer Science.* Clarendon Press, Oxford, 1990. Also online.

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. 2005.

Pit02. Andrew M. Pitts. Equivariant syntax and semantics. In *ICALP*, pages 32–36. Springer-Verlag, 2002.

PR05. David J. Pym and Eike Ritter. A games semantics for reductive logic and proof-search. In *GALOP (Games for Logic and Programming Languages)*, pages 107–123, 2005.

SP94. Paula Severi and Erik Poll. Pure type systems with definitions. In *LFCS*, pages 316–328. Springer, 1994.

SP05. Mark R. Shinwell and Andrew M. Pitts. On a monadic semantics for freshness. *Theoretical Computer Science*, 342(1):28–55, 2005.

SS04. U. Schöpp and I. Stark. A Dependent Type Theory with Names and Binding. In *CSL*, volume 3210 of *LNCS*, pages 235–249, 2004.

UPG04. C. Urban, A. 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 Technical appendix

We will use the following fact without comment:

Lemma 22 *If $\Gamma \vdash r$ is closed (so Γ mentions no unknowns and the freshness context is empty) and $\Gamma \vdash r$ is typable, then r mentions no unknowns.*

Proof (of Theorem 14). By induction on the derivation of $\Gamma \vdash r : \phi$.

- The case of (**Tax**). Suppose $\Gamma, a : \phi \vdash a : \phi$.
  It is easy to calculate that $important(\Gamma, a : \phi \vdash a) = \{\phi\}$; then $\phi \vdash \phi$ is a fact (see Footnote 8).
- The case of (**T⇒I**). Suppose $\Gamma, a : \phi \vdash \lambda a.s : \phi \Rightarrow \psi$ and $\Gamma, a : \phi \vdash s : \psi$.
  By inductive hypothesis $important(\Gamma, a : \phi \vdash s) \vdash \psi$. We use (⇒**I**).
  If $important(\Gamma, a : \phi \vdash \lambda a.s) = important(\Gamma, a : \phi \vdash s) \setminus \{\phi\}$ then we discharge ϕ.
  If $important(\Gamma, a : \phi \vdash \lambda a.s) = important(\Gamma, a : \phi \vdash s)$ then we discharge ϕ zero times. There are no other possibilities.

– The case of (**T⇒E**). Suppose $\Gamma \vdash r'r : \psi$ and $\Gamma \vdash r' : \phi \Rightarrow \psi$ and $\Gamma \vdash r : \phi$. By inductive hypothesis $\mathit{important}(\Gamma \vdash r') \vdash \phi \Rightarrow \psi$ and $\mathit{important}(\Gamma \vdash r) \vdash \phi$. By Lemma 12 and (**T⇒E**),

$$\mathit{important}(\Gamma \vdash r') \cup \mathit{important}(\Gamma \vdash r) \vdash \psi.$$

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,

$$\mathit{important}(\Gamma \vdash r'r) = \mathit{important}(\Gamma \vdash r') \cup \mathit{important}(\Gamma \vdash r).$$

The result follows.

– The case of (**T∀I**). Suppose $\Gamma, a : * \vdash \lambda a.r : \forall a.\phi$, where

$$\Gamma, a : * \vdash r : \phi \quad \text{and} \quad a \notin \mathit{fa}(\mathit{important}(\Gamma, a : * \vdash r)).$$

By inductive hypothesis $\mathit{important}(\Gamma, a : * \vdash r) \vdash \phi$. By (∀**I**),

$$\mathit{important}(\Gamma, a : * \vdash r) \vdash \forall a.\phi.$$

– The case of (**T∀E**). Suppose $\Gamma, b : * \vdash rb : \phi[a := b]$ and $\Gamma, b : * \vdash r : \forall a.\phi$. By inductive hypothesis $\mathit{important}(\Gamma, b : * \vdash r) \vdash \forall a.\phi$. By (∀**E**),

$$\mathit{important}(\Gamma, b : * \vdash r) \vdash \phi[a := b].$$

By reasoning similar to the case of (**T⇒E**) we can calculate that

$$\mathit{important}(\Gamma, b : * \vdash rb) = \mathit{important}(\Gamma, b : * \vdash r) \cup \mathit{important}(\Gamma, b : * \vdash b : *).$$

Now it is a fact that $\mathit{important}(\Gamma, b : * \vdash b : *) = \emptyset$. The result follows.

– The case of (**Tfr**). Suppose $\Gamma \vdash r : \phi$, and

$$\Gamma, A \vdash r : \phi \text{ and } \Gamma, A \vdash b\#r \text{ where } A \in \{b : \phi, b : *\}.$$

By inductive hypothesis $\mathit{important}(\Gamma, A \vdash r) \vdash \phi$.
If $A = b : *$ then

$$\mathit{important}(\Gamma, A \vdash r) = \mathit{important}(\Gamma \vdash r)$$

and the result follows immediately. If $A = b : \psi$ then since $\Gamma, A \vdash b\#r$ again

$$\mathit{important}(\Gamma, A \vdash r) = \mathit{important}(\Gamma \vdash r).$$

The result follows.

By abuse of appendices, we place the proof of Theorem 18 *before* that of Theorem 15. There is no circularity in the proofs and it is convenient for brevity; the special case of Theorem 18 when Γ mentions no unknowns, is needed to prove Theorem 15.

Proof (of Theorem 18). By induction on derivations. Only the case of (**T∀I**) is of interest.

– Suppose $\Gamma, a:*; \Delta \vdash r : \phi$ and $a \notin fa(important(\Gamma, a:*; \Delta \vdash r))$. By inductive hypothesis $\Gamma, a:*, B; \Delta, b\#\mathcal{X} \vdash r : \phi$ where $\mathcal{X} = unkn(\Gamma)$ and $b \notin \Gamma, a:*$. It is not hard to calculate that $\Gamma, a:*, B; \Delta, b\#\mathcal{X} \vdash b\#r$ and so

$$a \notin fa(important(\Gamma, a:*, B; \Delta, b\#\mathcal{X} \vdash r)).$$

The result follows.

Proof (of Theorem 15). 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 atom 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 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 $a:\phi \in \Gamma$ for no a then let $\Gamma' = \Gamma, a:\phi$ for some a not appearing in Γ. By Theorem 18 (see the comment preceding this proof) $\Gamma' \vdash r : \phi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$.
By $(\mathbf{T}{\Rightarrow}\mathbf{I})$ 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.

– The case $(\Rightarrow\mathbf{E})$. Suppose $\Phi \vdash \psi$ and $\Phi \vdash \phi \Rightarrow \psi$ and $\Phi \vdash \phi$.
By inductive hypothesis there exist:

- $\Gamma' \vdash r'$ such that $important(\Gamma' \vdash r') \subseteq \Phi$ and $\Gamma' \vdash r' : \phi \Rightarrow \psi$.
- $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$.

Without loss of generality we may assume that $\Gamma \cup \Gamma'$ satisfies our uniqueness condition; we rename atoms to make this true if necessary.
We use $(\mathbf{T}{\Rightarrow}\mathbf{E})$ and the fact that $important(\Gamma \cup \Gamma' \vdash r'r) \subseteq \Phi$.

– The case $(\forall\mathbf{I})$. Suppose $\Phi \vdash \forall a.\phi$ where $a \notin fa(\Phi)$ and $\Phi \vdash \phi$.
By inductive hypothesis there exists $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$. Since $a \notin fa(\Phi)$ we know that $a \notin fa(important(\Gamma \vdash r))$.
If $a:* \in \Gamma$ then let $\Gamma' = \Gamma$. If $a:\xi \in \Gamma$ for some type ξ then we are in the pathological situation that $a \notin fa(\phi)$ and $a:\xi \in \Gamma$ 'by mistake'; we rename a. If $a:* \notin \Gamma$ then let $\Gamma' = \Gamma, a:*$. By Theorem 18 $\Gamma' \vdash r : \phi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$.
We use $(\mathbf{T}\forall\mathbf{I})$ and the fact that $important(\Gamma' \vdash r) = important(\Gamma' \vdash \lambda a.r)$.

– The case $(\forall\mathbf{E})$. Suppose $\Phi \vdash \phi[a:=b]$ and $\Phi \vdash \forall a.\phi$.
By inductive hypothesis there are Γ and r such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \forall a.\phi$.
If $b:* \in \Gamma$ then let $\Gamma' = \Gamma$. If $b:* \notin \Gamma$ then let $\Gamma' = \Gamma, b:*$. By Theorem 18 $\Gamma' \vdash r : \forall a.\phi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$.

[9] Here x ranges over all atoms, not necessarily permutatively, so perhaps $x = a$.

We use (**T**$\forall$**E**) and the fact that

$$important(\Gamma' \vdash r) = important(\Gamma' \vdash rb).$$

Lemma 23 *Suppose that:*

– $\Gamma, X:\psi; \Delta \vdash r:\phi$ *and* $\Gamma; \Delta \vdash t:\psi$.
– $\Gamma; \Delta' \vdash a\#t$ *for every* $a\#X \in \Delta$.

Then $\quad important(\Gamma; \Delta' \vdash r[X := t]) \subseteq important(\Gamma, X:\psi; \Delta \vdash r)$.

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

– The case of a. Easy.
– The case of X. We calculate:

$$important(\Gamma, X:\psi; \Delta \vdash X) = \{\phi \mid a:\phi \in \Gamma,\ a\#X \notin \Delta\}$$

By assumption $important(\Gamma; \Delta' \vdash t) \subseteq \{\phi \mid a:\phi \in \Gamma,\ a\#X \notin \Delta\}$.

Other cases are easier.

Proof (of Lemma 19). By induction on the derivation of $\Gamma, X:\psi; \Delta \vdash r:\phi$. The first part is routine, we consider only two cases:

– The case (**a**$\#$**b**$'$). Suppose $\Gamma, a:*, b:\phi', X:\psi; \Delta \vdash a\#b$ is derived by (**a**$\#$**b**$'$). Then $a \notin fa(\phi')$ and it follows that

$$\Gamma, a:*, b:\phi'; \Delta' \vdash a\#b.$$

Since $b[X := t] \equiv b$, we are done.

– The case (**a**$\#\lambda$**b**). Suppose $\Gamma, X:\psi; \Delta \vdash a\#r$ and $\Gamma, X:\psi; \Delta \vdash a\#\lambda b.r$ is derived by (**a**$\#\lambda$**b**). By inductive hypothesis $\Gamma; \Delta' \vdash a\#r[X := t]$ and so

$$\Gamma; \Delta \vdash a\#\lambda b.(r[X := t]).$$

Since $\lambda b.(r[X := t]) \equiv (\lambda b.r)[X := t]$, we are done.

For the second part we consider some cases:

– The case (**Tax**). Suppose $Y:\xi \in \Gamma$ and $\Gamma, X:\psi; \Delta \vdash Y:\xi$ is derived by (**Tax**). Then $\Gamma; \Delta' \vdash Y:\xi$. Since $Y \equiv Y[X := t]$, we are done.
Similarly for $\Gamma, X:\psi; \Delta \vdash a:\xi$.
Suppose $\Gamma, X:\psi; \Delta \vdash X:\psi$ is derived by (**Tax**). By assumption $\Gamma; \Delta' \vdash t:\psi$. Since $t \equiv X[X := t]$, we are done.

– The case (**T**$\Rightarrow$**I**). Since $(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t])$. The cases of (**T**$\Rightarrow$**E**) and (**T**$\forall$**E**) are similar.

– The case (**T**$\forall$**I**). Since $(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t])$ and from Lemma 23.

– The case (**Tfr**).
Suppose $\Gamma,\ A,\ X{:}\psi; \Delta \vdash r:\phi$ because $\Gamma,\ A,\ X{:}\psi;\ \Delta,\ b\#\mathcal{X} \vdash r:\phi$ and suppose $\Gamma,\ A,\ X{:}\psi;\ \Delta, b\#\mathcal{X} \vdash b\#r$, where $b \notin \Delta$ and A is $b:*$ or $b:\xi$ for some ξ.
By inductive hypothesis and some calculations, $\Gamma,\ A;\ \Delta',\ b\#\mathcal{X}' \vdash r[X := t]:\phi$ and $\Gamma,\ A;\ \Delta',\ b\#\mathcal{X}' \vdash b\#r[X := t]$, for a suitable $\mathcal{X}'$ and Δ' where $b \notin \Delta'$. The result follows.

$$\frac{(A \in \{a:\phi,\ a:*,\ X:\phi\})}{\Gamma,\ A;\ \Delta \vdash A}\,(\mathbf{Tax}) \qquad \frac{\Gamma;\Delta \vdash r:\bot}{\Gamma;\Delta \vdash \mathsf{xf}(r):\phi}\,(\mathbf{T\bot E})$$

$$\frac{\Gamma,\ a:\phi;\ \Delta \vdash r:\psi}{\Gamma,\ a:\phi;\ \Delta \vdash \lambda a.r:\phi \Rightarrow \psi}\,(\mathbf{T{\Rightarrow}I}) \qquad \frac{\Gamma;\ \Delta \vdash r':\phi \Rightarrow \psi \quad \Gamma;\ \Delta \vdash r:\phi}{\Gamma;\ \Delta \vdash r'r:\psi}\,(\mathbf{T{\Rightarrow}E})$$

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

$$\frac{\Gamma,\ b:*;\ \Delta \vdash r:\forall a.\phi}{\Gamma,\ b:*;\ \Delta \vdash rb:\phi[a:=b]}\,(\mathbf{T\forall E})$$

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

$$\frac{}{\Gamma,\ a:\phi,\ b:\phi;\Delta \vdash a\#b}\,(\mathbf{a\#b}) \qquad \frac{}{\Gamma,\ a:*,\ b:*;\Delta \vdash a\#b}\,(\mathbf{a\#b''})$$

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

$$\frac{(a \notin fa(\phi))}{\Gamma,\ a:*,\ b:\phi;\Delta \vdash a\#b}\,(\mathbf{a\#b'}) \qquad \frac{\Gamma;\Delta \vdash a\#r' \quad \Gamma;\Delta \vdash a\#r}{\Gamma;\Delta \vdash a\#r'r}\,(\mathbf{a\#app}) \qquad \frac{\Gamma;\Delta \vdash a\#r}{\Gamma;\Delta \vdash a\#\mathsf{xf}(r)}\,(\mathbf{a\#xf})$$

Fig. 1. Typing and freshness derivation rules

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

Fig. 2. Natural Deduction style derivation rules

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

$$\frac{\Gamma, X{:}\psi;\Delta, a_1\#X,\ldots,a_n\#X \vdash r:\phi \quad \Gamma;\Delta \vdash t{:}\psi \quad \Gamma;\Delta \vdash a_i\#t\ (1{\le}i{\le}n) \quad (X{\notin}\Delta)}{\Gamma;\Delta \vdash r[X:=t]:\phi}\,(\mathbf{Cut})$$

Fig. 3. Admissible rules