a-logic

Murdoch J. Gabbay and Michael J. Gabbay

1 Introduction

Mathematics and computer science pervasively use logics formal languages for reasoning in. One of the most common is classical First-Order Logic with equality (FOL) [Bell and Machover, 1977, Gabbay and Günthner, 1986].

For example FOL is routinely used as a foundational tool to express Zermelo-Fraenkel set theory (**ZF sets**) [Johnstone, 1987, Machover, 1996] and thus as a (language to express) a foundation for classical mathematics. The implication is that first-order logic is a minimal and convenient logical framework for doing, well, triangles and numbers and other mathematical objects.

But is it so good for the mathematics of computation, i.e. for computer science? Perhaps not, since it cannot express substitution as an object-level operation on its own terms. The λ-calculus [Barendregt, 1984] can and does, in the sense that we can write $(\lambda a.s)t$, which β-reduces [Barendregt, 1984]:

$$(\lambda a.s)t \to s[a{\mapsto}t].$$

This is a pervasive phenomenon and theories of computation are frequently based on notions of substitution. (For this reason, some researchers refer informally to a general notion of '*movement*', which in the examples here is implemented by substitution.)

Another example is channel communication as appears for example in the π-calculus [Milner *et al.*, 1992]:

$$\overline{a}b \mid a\langle c\rangle.P \to P[c{\mapsto}b].$$

Now note that first-order logic cannot express this *directly*; consider a ternary **explicit substitution** term-former which we write $s[t{\mapsto}u]$. The intuition is that if t *is* a variable symbol, then $s[t{\mapsto}u]$ denotes the term obtained by replacing every instance of t in s by u.

This is ridiculous because even if t *were* a variable symbol, there is no way we could guarantee it would *still be* a variable symbol if that variable symbol is instantiated, as arises from the following consideration:

We want our logic to satisfy cut-elimination and thus we want to simplify a proof of the form

$$\dfrac{\dfrac{\Gamma \vdash P, \Delta}{\Gamma \vdash \forall x.\, P, \Delta} \qquad \dfrac{\Gamma, P[x{\mapsto}s] \vdash \Delta}{\Gamma, \forall x.\, P \vdash \Delta}}{\Gamma \vdash \Delta}\,(\mathbf{Cut})$$

to one without the cut. It suffices to demonstrate the **Substitution Lemma**:

Lemma. *If* $\Gamma \vdash \Delta$ *is derivable in our logic without cut, then* $\Gamma[x{\mapsto}s] \vdash \Delta[x{\mapsto}s]$ *is also derivable in our logic without cut, and with a no larger proof.*

(— 'no larger' according to whatever measure is useful for making an inductive argument work given the other transformations we make on the proof while eliminating cuts. The substitution action on Γ and Δ is element-wise. See Lemma 5.)

Likewise it is not possible to hypothesise binary term-formers λ and $\circ$ with axioms in first-order logic such that if a is a variable symbol then $(\lambda a.s) \circ u = s[a{\mapsto}u]$.

Yet there is a need to express the λ-calculus in logic, for any of several reasons:

- The untyped λ-calculus is a foundational system to rival ZF sets. Why should we be able to write down axioms for one in FOL, but not for the other?
- FOL has a well-developed and tractable meta-theory.
- Functions *are* useful.

Theoretical computer science as a field has reacted to these issues by working in higher-order frameworks, but this commits to certain very particular choices, e.g. committing to β-reduction as the only notion of substitution ('*movement*'), a particular discipline of types (which may prevent us from quantifying over all elements of the universe), and it may be expensive for the meta-theory (models are more complex, Interpolation need not hold, compactness fails, and so on).

So a first-order logic which can express explicit substitution as a term-former as described above, offers the prospect of combining the power of a functional theory, with the simplicity and flexibility of a first-order system.

In this paper we define a-logic, investigate its relation to First-Order Logic, prove some meta-theory for the logic as a formal system, and then

axiomatise the λ-calculus as a finite first-order theory. Finally, we consider axiomatisations of other theories.

The 'a' in 'a-logic' refers to the fact that the logic is for making statements about its variable symbols a, b, c. Any resemblance to a pronoun is coincidental.

2 a-logic

2.1 Terms, directions, predicates

We assume a countably infinite set x, y, z, a, b, n of **(term) variable symbols**, and a finite set of **term-formers** f to each of which is associated an **arity** $ar(f)$ which is a nonnegative number which may be 0.[1] If $ar(f) = 0$ we may call f a **constant**.

Terms are inductively generated as follows:

$$s, t ::= x \mid f(s, \ldots, s)$$

where f has $ar(f)$ arguments. If f is a constant we may write $f()$ as just f.

For reasons which will become clear later, we arbitrarily call some of the f **inevitably term-formers**. (For future reference, λ and application will be inevitably term-formers, but explicit substitution is not. We come back to this later.)

We write $s \equiv t$ for 's and t are syntactically identical terms'. We write $s \not\equiv t$ for 's and t are syntactically different terms'.

Assume some set of **predicate constant symbols** $p, q, r \ldots \in \mathbb{P}$, each with its arity $ar(p)$, $ar(q)$, $ar(r)$,... which is a nonnegative number which may be 0.

We assume distinguished predicate constants:

1. We call **at** the predicate **is an atom** (or **atom** for short), of arity 1.
2. We call $\bot$ **false** or **contradiction** of arity 0.
3. We call $=$ **equality**.

Write $p(t_1, \ldots, t_n)$ for a predicate constant applied to a list of terms of length n where we shall always assume $n = ar(p)$. Call this a **primitive proposition**.

Propositions are generated by the grammar

$$P ::= p(ts) \mid P \supset P \mid \forall x.\, P.$$

[1] Our reasons for using a as a variable symbol, not a constant, will become clear.

$\forall x.\, P$ is binding, nothing else is. Consistent with our convention for terms we write $P \equiv Q$ for 'P and Q are identical predicates' (up to α-equivalence).

Write VP for **the variables occurring in** P, defined by: $Vp(t_1, \ldots, t_n) = \bigcup Vt_i$, $V(P \supset Q) = VP \cup VQ$, and $V(\forall n.\, P) = VP \backslash \{n\}$.

Here, if S and T are sets then $S \setminus T$ is set of elements of S which are not elements of T. In particular $VP \backslash \{n\}$ denotes the set of elements of VP not equal to n. We use this notation without comment henceforth.

We may write $x \in P$ for $x \in VP$, we read it 'x **occurs in** P'. Then $x \notin P$ means 'x does not occur in P'. (We strictly equate predicates up to α-equivalence, we will *not* talk about 'occurring free' or 'occurring bound'.) Write $P[x \mapsto s]$ for P with every instance of the variable x replaced by s in the usual, capture-avoiding, manner.

Later, when we develop our notions of derivability and model, if we happen to know that **at** holds of a term, then we may choose a name like a, b, or n to represent it.

2.2 Contexts and judgements

A **(logical) context** Γ, Δ is a set of propositions. We way write $V\{P, Q\}$ for $VP \cup VQ$, and $V\Gamma$ for the natural meaning as a union; we may even mix, as in $V(\Gamma, P, \Gamma')$. We shall write $x \notin \Gamma, P, \Gamma'$ for $x \notin V(\Gamma, P, \Gamma')$, and so on.

A **judgment** is a pair of contexts which we write $\Gamma \vdash \Delta$. When a context is on the right-hand side of a judgement we call it a **cocontext**. The **valid** or **derivable** judgments are inductively defined by the following derivation rules (notation is defined below):

$$\frac{}{\Gamma, P \vdash P, \Delta}\,(\mathbf{Ax}) \qquad \frac{}{\Gamma, \bot \vdash \Delta}\,(\bot\mathbf{L})$$

$$\frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash P \supset Q, \Delta}\,(\supset \mathbf{R}) \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, P \supset Q \vdash \Delta}\,(\supset \mathbf{L})$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma \vdash \forall x.\, P, \Delta}\,(\forall\mathbf{R})^{(*)} \qquad \frac{\Gamma, P[x \mapsto s] \vdash \Delta}{\Gamma, \forall x.\, P \vdash \Delta}\,(\forall\mathbf{L}) \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma, P \vdash Q, \Delta}{\Gamma \vdash Q, \Delta}\,(\mathbf{Cut})$$

$$\frac{(s \text{ inevitably a term})}{\Gamma, \mathbf{at}\, s \vdash \Delta}\,(\mathbf{at}\,\mathbf{L}) \qquad \frac{\Gamma, \mathbf{at}\, a \vdash \Delta}{\Gamma \vdash \Delta}\,(\mathbf{Fresh})^{(*)}$$

$$\frac{}{\Gamma \vdash t = t, \Delta}\,(=\mathbf{R}) \qquad \frac{\Gamma, P[x \mapsto s'] \vdash \Delta}{\Gamma, s' = s, P[x \mapsto s] \vdash \Delta}\,(=\mathbf{L})$$

Here x and a are variables; bound by $\forall$, free otherwise. Conditions in

brackets are side-conditions whose validity can be decided just by examining syntax.

In particular the bracketed asterisks annotating ($\forall$**R**) and (**Fresh**) indicate that the rules are subject to the side-condition $x \notin \Gamma, \Delta$ or $a \notin \Gamma, \Delta$. (**Fresh**) is not mis-typed. It *eliminates* a variable. Comma, as in 'Γ, P', indicates set union. Note that this means that for example in ($\supset$ **R**), it may still be that $P \in \Gamma$ or indeed that $P \supset Q \in \Delta$.

We say that s is **inevitably a term** when it is of the form $f(t_1, \ldots, t_n)$ and f is an inevitable term-former. An example of a term which is not inevitably a term, is a term without any top-level term-former at all, namely y a variable symbol.

(Later on, we will encounter explicit substitution $s\langle t \mapsto u\rangle$ which is not inevitably a term, and λ-abstraction and application which are.)

Say a judgement $\Gamma \vdash \Delta$ is a **theorem** when a derivation exists concluding in $\Gamma \vdash \Delta$. Note that Δ may be empty in which case we write $\Gamma \vdash$; we may call Γ **inconsistent** when $\Gamma \vdash$ is derivable.

A subset of the logic is classical propositional logic, so we use standard sugar such as writing $\neg P$ for $P \supset \bot$, $P \vee Q$ for $(\neg P) \supset Q$, $\exists x.\, P$ for $\neg(\forall x.\, \neg P)$, $P \wedge Q$ for $\neg(P \supset (\neg Q))$, and in general we shall use other well-known shorthands for classical equivalences. We may also use 'derivation rules' such as ($\exists$**L**) and ($\neg$**L**), to represent several rules of the core system appropriately pasted together.

2.3 The first-order theory corresponding to a-logic

We now set up some notation. Uses of this notation will be scattered throughout the text; **all definitions relevant to theories are here**.

Let **FOL** be classical first-order logic with equality.

Say a triple of

- a finite set of **theory term-formers**, which may already be in the base logic (write a typical term-former f),
- a finite set of **theory atomic predicate symbols**, which may already be in the base logic (write a typical atomic predicate symbol ϕ),
- a possibly infinite set of **theory sentences** or **axioms**, which are predicates in the language of the base logic extended with any of the new term-formers and atomic predicate symbols,

is a **theory**. If the set of sentences is finite, call it a **finite theory**, otherwise call it an **infinite theory**.

When a theory has no, or a trivial, signature, we may equate it with the set of its axioms and name it using calligraphic letters, for example $\mathcal{T}$. Otherwise, we tend to give theories names in sans-serif font, for example Q or lambda. Theories named in sans-serif font will *always* be finite.

As a matter of notation we may write $\mathsf{Unc}{+}\mathsf{Cle}$ for the theory obtained by taking the unions of the theories' term-formers, predicate symbols, and sentences. We write $\neg\mathsf{Unc}$ for the theory obtained by negating all the theory sentences. Finally, if both Unc and Cle are finite, we may identify them with the predicate which is the logical conjunction of their senteces, and write things like $\mathsf{Unc} \supset \mathsf{Cle}$ or $\mathsf{Unc} \supset \neg\mathsf{Cle}$.

The rest of this subsection proves that a-logic corresponds to the following FOL theory alogic in a formal sense we make precise in a moment:

- There are some term-formers (e.g. f with arity n).
- There is one unary predicate symbol $\mathbf{at}$.
- There is an axiom
$$\exists a.\, \mathbf{at}\, a \qquad (\mathbf{folfresh})$$
- For each inevitable term-former f of arity n, there is an axiom:
$$\forall x_1, \ldots, x_n.\, \neg\mathbf{at}\, (f(x_1, \ldots, x_n)) \qquad (\mathbf{folat\,Lf})$$

Here is the characteristic property of inevitable term-formers:

LEMMA 1. $\mathsf{alogic},\ \mathbf{at}\, a,\ f(s_1, \ldots, s_n) = a$ *is inconsistent.*
(Formally: $\mathsf{alogic}, \mathbf{at}\, a, f(s_1, \ldots, s_n){=}a \vdash$ *.)*

Proof. We use $(\mathbf{folat\,Lf})$. ■

That is, 'in a-logic, terms which are inevitably terms, cannot be atoms'.

The intuition of an atom is a term which the logic identifies as being a variable symbol; the only way this can happen is if $\mathbf{at}\, t$ is derivable. A term is inevitably a term when the logic *always* identifies it as *not* being a variable symbol (and this happens when $\neg\mathbf{at}\, t$ is derivable).

LEMMA 2. $\Gamma \vdash \Delta$ *in a-logic if and only if* $\mathsf{alogic}, \Gamma \vdash \Delta$ *in FOL (with its signature extended with* $\mathbf{at}$ *and so on;* ***we shall not mention this again****).*

Proof. We can translate from a derivation in a-logic to one in FOL with alogic by transforming instances of $(\mathbf{Fresh})$ and $(\mathbf{at\,L})$ as shown (where for

simplicity we suppose f is unary):

$$\frac{\mathbf{at}\, a,\ \exists a.\, \mathbf{at}\, a,\ \Gamma \vdash \Delta}{\exists a.\, \mathbf{at}\, a,\ \Gamma \vdash \Delta}\,(\exists\mathbf{L}) \quad (a \notin \Gamma, \Delta)$$

$$\cfrac{\cfrac{\cfrac{}{\mathbf{at}\, f(s),\ \forall x.\, \neg\mathbf{at}\, f(x),\ \Gamma \vdash \Delta,\ \mathbf{at}\, f(s)}\,(\mathbf{Ax})}{\neg\mathbf{at}\, f(s),\ \mathbf{at}\, f(s),\ \forall x.\, \neg\mathbf{at}\, f(x),\ \Gamma \vdash \Delta}\,(\neg\mathbf{L})}{\mathbf{at}\, f(s),\ \forall x.\, \neg\mathbf{at}\, f(x),\ \Gamma \vdash \Delta}\,(\forall\mathbf{L})$$

■

So roughly speaking: (**Fresh**) just says that

an atom exists,

and (**at L**) says that

they cannot be directly described by (some) term-formers.

alogic is parametarised over the choice of which f are inevitable terms-formers, and we will always make that choice explicit.

COROLLARY 3. *a-logic is consistent ($\vdash$ is not derivable).*

Proof. FOL is sound and complete with respect to a standard notion of model [van Dalen, 1985, Machover, 1996], so $\Gamma \vdash \Delta$ in a FOL theory it suffices to show that the FOL theory has a model.

This is given by a two-element set $\{anatom, aterm\}$ (here *anatom* and *aterm* are merely suggestive names for two distinct elements, though in view of their uniqueness we might also call them *theatom* and *theterm*). Write [[-]] for the interpretation of a term or predicate.

$[\![f]\!]$ is the constant map to *aterm* no matter what the arity of f. $[\![\mathbf{at}\,]\!]$ is $\{anatom\}$. $[\![=]\!] = \{\langle anatom, anatom\rangle, \langle aterm, aterm\rangle\}$. Predicate constants can be interpreted arbitrarily. ■

Recall from the Introduction that we may call a variable symbol a of which we know **at** a an **atom**.

So is a-logic trivial, if it has such a simple model? Of course a model of pure FOL is the one-element set. The interest comes in provability and its properties, and in models of more complex predicates than '$\top$'. The same holds for a-logic. What we have demonstrated is that an a-logic theory is a special case of a first-order theory, so we can be optimistic that a-logic should behave meta-theoretically very much like FOL (since that is what it is).

2.4 Comments on the rules

This is a simple logic consisting of First-Order Logic with equality, augmented with a unary predicate **at** with two rules (**Fresh**) and (**at L**) as usual, though instead of left- and right-introduction rules we have a left - introduction and -elimination rule.

Recall from the last section that the distinguishing feature of an 'inevitable' term-former f is that (**at L**) may be used for a term of which it is the top-level term-former, or equivalently one for which the corresponding alogic has an axiom (**folat Lf**). The *need* for this becomes clear in §4.1. But why do we say 'inevitably a term'?

One of the important basic lemmas of First-Order Logic is the *Substitution Lemma* (see Lemma 5 for the a-logic version):

$$\text{If } \Gamma \vdash \Delta \text{ is derivable, then so is } \Gamma[a{\mapsto}s] \vdash \Delta[a{\mapsto}s].$$

This makes formal the idea that 'variables stand for unknown terms' and gives them a slightly dynamic character in the logic. Suppose $\mathbf{at}\, f(t_1, \ldots, t_n)$ gives rise to contradiction for certain f. Then we may also obtain a contradiction from $\mathbf{at}\, f(t_1[a{\mapsto}s], \ldots, t_n[a{\mapsto}s])$. The property of 'having top-level term-former f' is invariant under substitution. Therefore, (**at L**) is not incompatible with the Substitution Lemma after all.

We say 'inevitably' because substitutions do not affect the top-level term-former. We say 'a term' because the property of being a variable symbol is the canonical example of a property which is *not* inevitable (i.e. variables get substituted; that is what they are there for!).

Accordingly, there is no (**at R**) rule such as

$$\Gamma \vdash \mathbf{at}\, a, \Delta$$

if a *is* a variable symbol.

$\mathbf{at}\, \langle a, a\rangle \vdash$ is a theorem with a trivial proof by (**at L**). (We assume a pair term-former $\langle \text{-}, \text{-}\rangle$.)

In summary, $\mathbf{at}\, a$ in Γ represents a promise that a will never be instantiated, or at least, not to the wrong kind of term. We may call a variable symbol a of which we know $\mathbf{at}\, a$ an **atom**.

3 Meta-properties of a-logic

3.1 Cut-elimination

The proof-theory of a-logic is hardly more complex than that of FOL itself.

LEMMA 4 (Weakening). *If* $\Gamma \vdash \Delta$ *then* $\Gamma, \Gamma' \vdash \Delta', \Delta$.

Proof. It suffices to consider the derivation rules one-by-one and check that if we weaken the conclusion of an instance with Γ' and Δ', then we may weaken the hypotheses in the same way, and still have an instance of the derivation rule. We shall see that we also have to remember a systematic 'freshening' of the variables, but this is no problem.

This is quite obvious for (**Ax**), ($\supset$ **R**), ($\supset$ **L**), and ($\bot$**L**), because these rules make no comment on Γ and Δ so they 'might as well be as big as we like'.

($\forall$**R**) has a side-condition that $x \notin \Gamma, \Delta$ (we say a **is fresh**). If we weaken context and cocontext to Γ, Γ' and Δ, Δ' the instance of the rule will be invalid precisely when $x \in \Gamma'$ or $x \in \Delta'$.

However, we can use *any* fresh x in ($\forall$**R**), and we simply choose some x' *not* in Γ, Γ', Δ, or Δ' and rename x to x' systematically from then on as we move upwards, before weakening.[2]

The rest of the rules are just as easy: (**Fresh**) has a condition similar to ($\forall$**R**), which can be dealt with similarly. ■

Call a derivation **cut-free** when it does not use (**Cut**). Say it has **depth** n when the greatest number of deduction steps from the conclusion to an assumption is n. If S is a set of predicates write $S[y{\mapsto}t]$ for the set obtained by applying the substitution $[y{\mapsto}t]$ pointwise to S. We now prove the **substitution lemma**:

LEMMA 5. *If* $\Gamma \vdash \Delta$ *has a cut-free derivation of length* n *then* $\Gamma[y{\mapsto}t] \vdash \Delta[y{\mapsto}t]$ *has a cut-free derivation of length at most* n*.*

Proof. It suffices to consider the derivation rules one-by-one and check that if we apply $[y{\mapsto}t]$ to the conclusion, we may do the same to the hypotheses, and still have an instance of the derivation rule. As for Weakening above we may need to systematically accumulate some other transformations, but they are not a problem.

Rules (**Ax**), ($\supset$ **R**), ($\supset$ **L**), and ($\bot$**L**) are not a problem because they are propositional and do not care what terms the predicates mention.

($\forall$**R**) is problematic — what if $x \equiv y$ so that $[y{\mapsto}t]$ affects P in the hypotheses, and what if $x \in s$ and $y \in \Gamma, \Delta$ so that $x \in \Gamma[y{\mapsto}t], \Delta[y{\mapsto}t]$ (violating the side-condition)? As for weakening, these problems are solved simply by choosing a fresher x'.

[2] What if we choose an x' which is introduced by some other ($\forall$**R**) further up in the proof? Valid answers are: *you have the proof in your hand; check, and make sure you don't* or *don't look ahead, but freshen them up if and when you come to them.* This is a form of α-equivalence at the level of proofs which is common to the theory of proofs of any logic with binding.

In ($\forall\mathbf{L}$) it suffices to observe that by α-equivalence on formulae we may assume $x \not\equiv y$ and $x \notin t$ so that

$$(\forall x.\, P)[y{\mapsto}t] \equiv \forall x.\, (P[y{\mapsto}t]) \qquad P[x{\mapsto}s][y{\mapsto}t] \equiv P[y{\mapsto}t][x{\mapsto}s[y{\mapsto}t]]$$

and after the substitution we still have a valid instance of ($\forall\mathbf{L}$) where s is replaced by $s[y{\mapsto}t]$.

In (**at L**) it suffices to observe that if s is not a variable, neither is $s[y{\mapsto}t]$.

(**Fresh**) is not a problem unless ('by accident') $a \equiv y$; if we apply $[y{\mapsto}t]$ naïvely then after the substitution we no longer have an instance of the rule. The answer is, as for ($\forall\mathbf{R}$), to rename a to some fresher a' and then apply the substitution. ■

LEMMA 6. (**Fresh**) *followed by any other rule* $(*)$*, may be commuted to* $(*)$ *followed by* (**Fresh**)*.*

As a corollary, all instances of (**Fresh**) *in a derivation may be commuted downwards to the head of the proof.*

Proof. We verify that derivations may be transformed as follows:

$$\dfrac{\Gamma \vdash P, \Delta \qquad \dfrac{\Gamma, P, \mathbf{at}\, a \vdash \Delta}{\Gamma, P \vdash \Delta}\,(\mathbf{Fresh})}{\Gamma \vdash \Delta}\,(\mathbf{Cut}) \quad\Longrightarrow\quad \dfrac{\dfrac{\Gamma, \mathbf{at}\, a \vdash P, \Delta \quad \Gamma, \mathbf{at}\, a, P \vdash \Delta}{\Gamma, \mathbf{at}\, a \vdash \Delta}\,(\mathbf{Cut})}{\Gamma \vdash \Delta}\,(\mathbf{Fresh})$$

This is easy, we just need Weakening to add $\mathbf{at}\, a$ to the derivation in the top left. Similarly:

$$\dfrac{\dfrac{\Gamma, P[x{\mapsto}s], \mathbf{at}\, b \vdash \Delta}{\Gamma, P[x{\mapsto}s] \vdash \Delta}\,(\mathbf{Fresh})}{\Gamma, \forall x.\, s \vdash \Delta}\,(\forall\mathbf{L}) \quad\Longrightarrow\quad \dfrac{\dfrac{\Gamma, \mathbf{at}\, b, P[x{\mapsto}s] \vdash \Delta}{\Gamma, \mathbf{at}\, b, \forall x.\, P \vdash \Delta}\,(\forall\mathbf{L})}{\Gamma \vdash \Delta}\,(\mathbf{Fresh})$$

This is easy, since if $b \notin \Gamma, P[x{\mapsto}s], \Delta$ then certainly $b \notin \Gamma, \forall x.\, P, \Delta$.

$$\dfrac{\dfrac{\Gamma,\ \mathbf{at}\, a \vdash P,\ \Delta}{\Gamma \vdash P,\ \Delta}\,(\mathbf{Fresh})}{\Gamma \vdash \forall y.\, P,\ \Delta}\,(\forall\mathbf{R}) \quad\Longrightarrow\quad \dfrac{\dfrac{\Gamma,\ \mathbf{at}\, a \vdash P,\ \Delta}{\Gamma,\ \mathbf{at}\, a \vdash \forall y.\, P,\ \Delta}\,(\forall\mathbf{R})}{\Gamma \vdash \forall y.\, P,\ \Delta}\,(\mathbf{Fresh})$$

By the side-condition on (**Fresh**) we can assume $a \not\equiv y$ and $a \notin \Gamma, \Delta, P$. By the side-condition on ($\forall\mathbf{R}$) we also know $y \notin \Gamma, \Delta$. Therefore, $y \notin \Gamma, \Delta, \mathbf{at}\, a, P$ and $a \notin \Gamma, \Delta, \forall y.\, P$. Therefore in the transformed derivation above, the instances of (**Fresh**) and ($\forall\mathbf{R}$) are still valid.

We consider only one more case:

$$\dfrac{\dfrac{\Gamma,\ \mathbf{at}\,a,\ P[x{\mapsto}s'] \vdash \Delta}{\Gamma,\ P[x{\mapsto}s'] \vdash \Delta}\,(\mathbf{Fresh})}{\Gamma,\ P[x{\mapsto}s],\ s' = s \vdash \Delta}\,(=\mathbf{L}) \Longrightarrow \dfrac{\dfrac{\Gamma,\ \mathbf{at}\,a,\ P[x{\mapsto}s'] \vdash \Delta}{\Gamma,\ \mathbf{at}\,a,\ s' = s,\ P[x{\mapsto}s] \vdash \Delta}\,(=\mathbf{L})}{\Gamma,\ s' = s,\ P[x{\mapsto}s] \vdash \Delta}\,(\mathbf{Fresh})$$

We need only verify that $a \notin s$. If this is not the case, we rename a in the derivation of $\Gamma,\ \mathbf{at}\,a,\ P[x{\mapsto}s'] \vdash \Delta$ first. ■

THEOREM 7 (Cut elimination). *If $\Gamma \vdash \Delta$ has a derivation then it has a cut-free derivation.*

Proof. We work by induction on the pair of numbers $(cuts, depth)$ where $cuts$ is the number of instances of (**Cut**) in the derivation, and $depth$ is its depth — lexicographically ordered. The essential cases are just as for first-order logic (since **at** has no right rule, there can be no essential case for it); we use the previous lemmas in standard ways [van Dalen, 1985, Machover, 1996].

The commutation cases are also as for first-order logic except we worry about commuting (**Fresh**) from between a left- and a right-introduction. It suffices to verify that (**Fresh**) may be commuted *down* through all the other rules. This follows by the previous lemma. ■

3.2 Interpolation

The following result is known as Craig's Interpolation Theorem for First-Order Logic ([D.M.Gabbay, 2005, Theorem 5.65, page 169], [G.Boolos, 1989, Section 23]):

THEOREM 8. *In classical first-order logic with equality, if $P \supset Q$ is provable then there is some I mentioning only predicate symbols and term-formers mentioned in both P and Q, and possibly also $=$, such that $P \supset I$ and $I \supset Q$ are provable.*

We call I the **the interpolant** and say it is in the **common sublanguage** of P and Q. Because $=$ may appear in the interpolant even if it does not occur in P and Q, we call such an atomic predicate a **logical symbol**.

An easy corollary is:

THEOREM 9. *In classical a-logic with equality, if $P \supset Q$ then there is an interpolant I, such that $P \supset I$ and $I \supset Q$ are provable; $=$ and* **at** *are logical symbols.*

More specifically, if P or Q mention **at** *(but not necessarily both) it may still be that I mentions* **at**, *but if neither P nor Q mention* **at** *then the interpolant will not.*

Proof. In §2.3 we characterised a-logic as a theory alogic in FOL. Therefore $P \supset Q$ is equivalent to $P \wedge \mathsf{alogic} \supset Q$ and $P \supset (\mathsf{alogic} \supset Q)$. By Interpolation, interpolants exist.

We now observe that alogic mentions **at** and an unspecified collection of term-formers. By partitioning these axioms judiciously between the left- and right-hand sides of the implication to minimise the term-formers which appear on both sides, we can obtain the result of the theorem. ■

3.3 Models

Since alogic is a FOL theory and FOL has Soundness and Completeness, to give a theory of models it suffices to recall the theory of models for FOL, which is first-order classical logic with equality.

Here we give a *very brief* recap:

A **model Q** of a theory Q is:

- An **underlying set** $|Q|$, which we just write $\mathbf{Q}$.
- For each n-ary function-symbol f, a function $[\![f]\!] : \mathbf{Q}^n \to \mathbf{Q}$.
- For each atomic predicate symbol ϕ, a function $[\![\phi]\!] : \mathbf{Q}^n \to \mathit{Bool}$, where $\mathit{Bool} = \{\mathit{True}, \mathit{False}\}$ is a two-element set representing truth values. We may treat $[\![\phi]\!]$ as a set, writing $[\![\phi]\!] \subseteq \mathbf{Q}^n$.

(Recall that we are working, for simplicity, with an unsorted logic, so there is only one underlying set to consider.)

We can either insist that equality be identity in the model, or treat it as just another one of the atomic predicate symbols, in which case the model should also satisfy

> If $(p, q) \in [\![=]\!]$ and $\langle p_1, \ldots, p, \ldots, p_n\rangle \in [\![\phi]\!]$, then $\langle p_1, \ldots, q, \ldots, p_n\rangle \in [\![\phi]\!]$.

We use this latter approach in this paper.

The interpretation of atomic predicates can be extended to predicates in a standard way. Write σ for functions from variable symbols to elements of $\mathbf{Q}$, and call these **valuations**.

- $[\![\bot]\!]\sigma$ is false.
- $[\![P \supset Q]\!]\sigma$ is true when *if* $[\![Q]\!]\sigma$ is true, then $[\![P]\!]\sigma$ is true.
- $[\![\forall x.\, P]\!]\sigma$ is true when $[\![P]\!]([x{\mapsto}p]\sigma)$ is true, for every $p \in \mathbf{Q}$. Here $[x{\mapsto}p]\sigma$ represents the valuation which acts like σ, only x maps to p.

Say $[\![\mathbf{Q}]\!] \models P$ when for all valuations, $[\![P]\!]\sigma$ is true. Say $\Gamma \models_{[\![\mathbf{Q}]\!]} \Delta$ when $[\![\mathbf{Q}]\!] \models \bigwedge_i G_i \supset \bigvee_j D_j$, where $\Gamma = \{G_1, \ldots\}$ and $\Delta = \{D_1, \ldots\}$ and we interpret empty conjunction as $\top \equiv \bot \supset \bot$ and empty disjunction as $\bot$.

Then a standard theorem is **Soundness and Completeness** for FOL:

THEOREM 10. *$\Gamma \vdash \Delta$ if and only if $\Gamma \models_{[\![\mathbf{Q}]\!]} \Delta$ for all models $[\![\mathbf{Q}]\!]$.*

4 Substitution

We now give a *theory of substitution*.

4.1 Explicit substitution

Let sub have a ternary term-former *sub* which we sugar to $s\langle u{\mapsto}t\rangle$. Being a ternary term-former, $s\langle u{\mapsto}t\rangle$ is valid syntax whether or not u is a variable symbol — however, our axioms allow us to prove nothing about $s\langle u{\mapsto}t\rangle$ unless $\mathbf{at}\, u$ is known.

There are no new predicate symbols.

As a matter of notation, write

$$a\#u \quad \text{is sugar for} \quad \mathbf{at}\, a \ \wedge\ \forall x.\, u\langle a{\mapsto}x\rangle = u.$$

Intuitively we can read $a\#u$ as 'a is fresh for u' or 'a does not occur (free) in u'. Note that this is not a syntactic judgement; for example a does not occur in the syntax of x, but if we know nothing about x, intuitively $a\#x$ may or may not hold.

Then, sentences are:

$$\mathbf{at}\, a \supset u\langle a{\mapsto}a\rangle = u \qquad \mathbf{at}\, a \supset a\langle a{\mapsto}x\rangle = x \qquad \mathbf{at}\, a \wedge \mathbf{at}\, b \supset (a \neq b \Leftrightarrow a\#b)$$
$$\mathbf{at}\, a \wedge b\#u \supset u\langle a{\mapsto}b\rangle\langle b{\mapsto}y\rangle = u\langle a{\mapsto}y\rangle \qquad \mathbf{at}\, a \wedge a\#x \supset a\#u\langle a{\mapsto}x\rangle$$
$$\mathbf{at}\, b \wedge a\#b \wedge a\#y \supset u\langle a{\mapsto}x\rangle\langle b{\mapsto}y\rangle = u\langle b{\mapsto}y\rangle\langle a{\mapsto}x\langle b{\mapsto}y\rangle\rangle$$

(Note that $a\#u$ implies $\mathbf{at}\, a$.) Here a, b, x, y, z, and u are all variable symbols which are universally quantified. For example, the first axiom is in fact $\forall a, u.\, \mathbf{at}\, a \supset u\langle a{\mapsto}a\rangle = u$.

We do *not* introduce a sentence

$$\forall x, y, z.\, \neg\mathbf{at}\, (x\langle y{\mapsto}z\rangle) \qquad (\mathbf{folat}\, \mathbf{L})$$

nor the corresponding deduction rule $(\mathbf{at}\, \mathbf{L})$ for *sub*. This is to avoid the following derivation being valid:

$$\dfrac{\dfrac{\dfrac{}{\mathbf{at}\, a\langle a{\mapsto}a\rangle \vdash}\,(\mathbf{at}\, \mathbf{L})}{\mathbf{at}\, a \vdash}\,(\text{the first axiom})}{\vdash}\,(\mathbf{Fresh})$$

LEMMA 11. sub+alogic *is consistent.*

Proof. As in Lemma 2 it suffices to give a FOL model, call it $\mathcal{N}$. We recall the model from Lemma 2, but here $\mathcal{N}$ is (intuitively) all atoms with no terms.

The underlying set is $\mathbb{N}$ where $\mathbb{N}$ is the **natural numbers** $0, 1, \ldots$. $[\![=]\!] = \bigcup_{n\in\mathbb{N}}\{\langle n, n\rangle\}$ is the usual graph of equality on the underlying set. $[\![\mathbf{at}\,]\!] = \mathbb{N}$.

Clearly $[\![\exists a.\, \mathbf{at}\, a]\!]$ is 'true'; choose $[\![a]\!] = 1$.

Write p, q, r for arbitrary elements of our model (**we tend to do the same later** when we consider other models). By abuse of notation, henceforth we write $p\langle q{\mapsto}r\rangle$ instead of $[\![sub]\!](p, q, r)$, and $\mathbf{at}\, p$ instead of $p \in [\![\mathbf{at}\,]\!]$, and so on, without comment. **We also use this kind of shorthand later.**

$[\![sub]\!]$ is defined by:

$$n\langle n{\mapsto}m\rangle = m \qquad n\langle n'{\mapsto}m\rangle = n \quad (n' \neq n)$$

It remains to verify that the derivation rules are valid of these interpretations and that they make the interpretations of the axioms valid. This is easy:

- We check in the definition above that $p\langle n{\mapsto}n\rangle = p$ for all p and all n ($\mathbf{at}\, p$ holds always).
- We similarly check that $n\langle n{\mapsto}p\rangle = p$ always.
- We check that $n\#m$ holds precisely when $n \neq m$.
- The other equalities follow by checking of cases.

■

We can easily extend the model $\mathcal{N}$ with 'terms'; for example, introduce an element $aterm$ such that $aterm \notin \mathbf{at}$, $aterm\langle n{\mapsto}p\rangle = aterm$, and $p\langle aterm{\mapsto}q\rangle = p$. It is not hard to check that the axioms are still satisfied.

To see more complex models, the reader can either verify that 'ordinary syntax' with 'ordinary substitution' (e.g. the syntax of first-order logic with capture-avoiding substitution of terms for variables) is also a model, or they can wait for later when we use models of ECA to construct models of lambda (see below for terminology).

We conclude with some simple lemmas:

LEMMA 12.

1. *$a\#u$ if and only if for some $b\#u$, $u\langle a{\mapsto}b\rangle = u$. (So to test freshness, it suffices to try freshening the variable and checking for equality.)*

2. If $\mathbf{at}\, a$ *and* $b\#x$ *and* $b\#y$*, then* $b\#x\langle a\mapsto y\rangle$*. (A useful case.)*

Proof.

1. Suppose $a\#u$. Then $u\langle a\mapsto b\rangle = u$ by definition. Conversely suppose $u\langle a\mapsto b\rangle$ for some $b\#u$. Then by an axiom, $u\langle a\mapsto b\rangle\langle b\mapsto y\rangle = u\langle a\mapsto y\rangle$ for any y.
2. By an axiom $x\langle a\mapsto y\rangle = x\langle a\mapsto b\rangle\langle b\mapsto y\rangle$. Then $b\#x\langle a\mapsto b\rangle\langle b\mapsto y\rangle$ follows by another axiom.

■

LEMMA 13. $\mathsf{sub} + \mathsf{alogic}$, $\mathbf{at}\, a$, $\mathbf{at}\, b \not\vdash a \neq b$. *(Two atoms with different names are not necessarily distinct.)*

Proof. Consider a valuation on a and b in the model above, making them equal. The result follows by soundness and completeness of FOL. ■

LEMMA 14. *Any model of* sub *must have at least two elements.*

Proof. By (**Fresh**), the underlying set is non-empty, since it contains an element a such that $\mathbf{at}\, a$. Now suppose this is the only element. It is easy to check that $a\#a$, which contradicts the right-to-left direction of the third axiom of sub ($\mathbf{at}\, a \wedge \mathbf{at}\, b \wedge a\#b \supset a \neq b$). ■

$\exists x, y.\, x \neq y$ alone is sufficient, in the presence of the other axioms, to give the right-to-left direction of the third axiom of sub.

The reader familiar with Nominal Techniques [Gabbay and Pitts, 2001] might expect an axiom along the lines of

$$(\mathbf{Fresh}\#) \qquad \forall x.\, \exists a.\, a\#x$$

('no x can mention all atoms'), or even something with the meaning of 'no x can mention more than finitely many atoms' [Gabbay and Pitts, 2001, (Fresh), page 8] (call this **finite support**). This would certainly be useful.

The problem is, (**Fresh**#) only allows us to choose a fresh for x, not necessarily $x_1, \ldots, x_n$ for arbitrarily large n. Unless the language of terms has something like pairing — e.g. a binary term-former of which we can deduce $a\#(x, y) \Leftrightarrow a\#x \wedge a\#y$ — then the only option is to introduce an axiom scheme over all n. Rather than do that (and make our theories infinite), we wait for a more specific theory (lambda for example) and then add made-to-measure axioms with the effect of (**Fresh**#). As for insisting on finite support, we do not bother; it would make some things easier, but for our purposes we have enough.

4.2 'Provably closed'

Meta-level substitution and explicit substitution are quite different. For example $x[y{\mapsto}z] \equiv x$ but $x\langle y{\mapsto}z\rangle = x$ is not derivable in general. This fact is clearly brought out by the following definition, which will be useful in a moment:

In sub, write

$$\bullet s \quad \text{for} \quad \forall a.\, \mathbf{at}\, a \supset a\#s$$

where $a \notin s$. Read $\bullet s$ as 's **is (provably) closed**'. By abuse of terminology, we may use 's **is open**' to mean 's is not provably closed'.

It is important to note that $\bullet s$ does not at all imply that the term s is *actually* closed (viewed as syntax). For example, $\bullet x \vdash \bullet x$ but x *is* a variable. Similarly, if $\bullet x$ then $\forall a.\, \mathbf{at}\, a \supset \bullet(a\langle a{\mapsto}x\rangle)$, but $a\langle a{\mapsto}x\rangle$ clearly has free variables a and x.

However, in an underlying model $\bullet p$ indicates precisely the notion of closure given by 'is invariant under the explicit substitution'. For example recalling $\mathcal{N}$ the model of sub used in the proof of Lemma 11, it is easy to prove

LEMMA 15. *$\mathcal{N}$ has no closed elements. Recalling also the extension of $\mathcal{N}$ with aterm, the element aterm* is *closed.*

Proof. By routine concrete calculations which we omit. ■

Note also that there are no binders in terms, in the sense of a term-former which quotients the syntax of terms by α-equivalence. The *only* binder anywhere in this paper, is $\forall$ which is inherited from FOL and exists at the level of predicates (and the few times we mention 'real' λ-abstraction if we want to build a 'real' function).

(Recall from the end of §2.2 that when Γ entails the empty set we write $\Gamma \vdash$ and say Γ is inconsistent.)

So if a term can contain variable symbols and *still* be 'closed', what does 'closed' mean? Here are two examples of things we can derive:

LEMMA 16. *In* sub*...*

- $\mathbf{at}\, a, \bullet a \vdash$ *is derivable (atoms cannot be closed).*
- $\mathbf{at}\, a, \bullet x \vdash \bullet a\langle a{\mapsto}x\rangle$ *is derivable (if x is closed then, well, x is closed).*

Proof. In the following derivation, we omit sub.

$$\frac{\mathbf{at}\, a,\ a\#a \vdash \qquad \mathbf{at}\, a \vdash \mathbf{at}\, a}{\forall b.\, \mathbf{at}\, b \supset b\#a,\ \mathbf{at}\, a \vdash}\,(\forall\mathbf{L}), (\supset\mathbf{L})$$

$\mathbf{at}\, a,\ a\#a \vdash \bot$ follows easily from sub.

For the second part, we can use either

$$\mathbf{at}\, a \supset a\langle a{\mapsto}x\rangle = x \qquad \text{or} \qquad \mathbf{at}\, a \wedge a\#x \supset a\#u\langle a{\mapsto}x\rangle$$

— we omit the derivations. ■

Note that $aterm_i$ from the model used in Lemma 11 are closed, and they the only closed element of the model (all the other elements are atoms).

Say a model of sub is **non-trivial** when there exist p and q such that $\bullet p$ and $\bullet q$ and $p \neq q$. The model used in Lemma 11 demonstrates that sub has at least one nontrivial model. Using Soundness and Completeness of FOL we have:

LEMMA 17. $\mathsf{sub} + \mathsf{alogic}, \bullet x, \bullet y \nvdash x = y$.

So let us take stock. We have constructed a logic with a theory sub which 'internalises the meta-level' enough to express substitution as an explicit term-former. We can now exploit this to directly and finitely axiomatise the untyped λ-calculus as a first-order theory, and explore in what sense that axiomatisation is correct.

5 λ-calculus

5.1 The theory lambda

Assume sub, the theory of explicit substitution above.

We now construct lambda, an axiomatisation of the extensional λ-calculus. The signature has binary term-formers **application** and **lambda**, write them st and $\lambda s.t$, and axioms:

(α)	$\forall a, b, x, y.\, \mathbf{at}\, a \wedge b\#x \supset \lambda a.x = \lambda b.x\langle a{\mapsto}b\rangle$
(β)	$\forall a, x, y.\, \mathbf{at}\, a \wedge \bullet y \supset (\lambda a.x)y = x\langle a{\mapsto}y\rangle$
(ξ)	$\forall x, y.\, \bullet x \wedge \bullet y \supset (\forall z.\, \bullet z \supset xz = yz) \supset x = y$
$(\sigma\lambda)$	$\forall a, b, x, y.\, \mathbf{at}\, b \wedge a\#y \supset (\lambda a.x)\langle b{\mapsto}y\rangle = \lambda a.(x\langle b{\mapsto}y\rangle)$
$(\sigma\mathbf{app})$	$\forall a, x, y, z.\, \mathbf{at}\, a \supset (xy)\langle a{\mapsto}z\rangle = (x\langle a{\mapsto}z\rangle)(y\langle a{\mapsto}z\rangle)$
$(\#\mathbf{app})$	$\forall x, y.\, \exists a.\, a\#x \wedge a\#y \wedge \forall b.\, (b\#axy \Leftrightarrow (b\#a \wedge b\#x \wedge b\#y))$

We explore in what precise sense this can be said to be an 'axiomatisation of the extensional λ-calculus' in the rest of this section.

Note that lambda is built on top of sub. So let us suppose that sub *is* 'substitution' and that $a\#x$ *is* 'a does not occur (free) in x' — whatever those sentences mean. In that case we can say:

- (α) is α-conversion.

- (β) is β-conversion. The condition that it be applied to a provably closed term means that a term like $(\lambda a.a)b$ (for a and b atoms) will *not* reduce; as if it were waiting for b to 'become' something, which it can do if an explicit substitution arrives from a surrounding term. Compare this with [Fernández *et al.*, 2005], where a reduction strategy is considered which only allows reductions for *closed* terms; in their case the motivation is efficient reduction.

- (ξ) is a form of extensionality, restricted to the world of provably closed terms.

- $(\sigma\lambda)$ and $(\sigma\mathbf{app})$ are the usual rules for moving a substitution inside a term (the action of substitution on atoms is specified in sub).

- $(\#\mathbf{app})$ says 'there are infinitely many atoms and no x can mention them all'. See Lemma 18.

Interesting points about this axiomatisation are:

- λ is a binary term-former — *just like application.* The special status of the first argument is mediated through $\mathbf{at}$.

- This is a finitely axiomatised FOL theory; quantification over all elements is $\forall x.blah$; quantification over all *atoms* (i.e. object-level unknowns) is $\forall x.\mathbf{at}\, x \supset blah$.

- The underlying domain need not be an inductive type. It is some set with interpretations for the signature, that is, λ, explicit substitution, and $\mathbf{at}$.

- The theory includes an *explicit* term-former for substitution, substitution is *not* emulated by applying a λ-abstraction to a term.

- The condition $\bullet y$ in (β), and similar conditions in (ξ), give terms a two-level structure: provably closed terms are the 'real' λ-terms, because we can β-reduce on them.

  This gives open terms something of a flavour of an *internal metalanguage* for talking about terms.

- Accordingly, we can think intuitively as follows:

  1. Unknown λ-terms are variables.
  2. Unknown λ-term *variables* are atoms (terms of which we can prove $\mathbf{at}$).

3. Known λ-terms (the 'real' λ-terms) are provably closed (terms of which we can prove $\bullet$).

- Explicit substitution acts on u whether or not u is provably closed (and so is part of the internal meta-language).
- The axioms of lambda represent just one choice, out of many possible systems.

  For example, rules

  $$(\beta') \quad \forall a, x, y.\, \mathbf{at}\, a \wedge \neg \mathbf{at}\, y \supset (\lambda a.x)y = x\langle a \mapsto y\rangle$$
  $$(\xi') \quad \forall x, y.\, \neg \mathbf{at}\, x \wedge \neg \mathbf{at}\, y \supset (\forall z.\, \bullet z \supset xz = yz) \supset x = y$$

  are quite reasonable; they just equate more terms of the internal meta-language.
- The following axiom is *wrong* and *inconsistent*:

  $$(\beta_{\mathbf{FALSE}}) \qquad \forall a, x, y.\, \mathbf{at}\, a \supset (\lambda a.x)y = x\langle a \mapsto y\rangle$$

  Using it, it is not hard to show that $\mathbf{at}\, a \vdash a = (\lambda a.a)a$. However, $\neg \mathbf{at}\, ((\lambda a.a)a)$ (because the top-level term-former is application), so $\mathbf{at}\, a \vdash \bot$. Using (**Fresh**) it is easy to derive $\vdash$, i.e. the system becomes inconsistent and has no models.

Write lambda as shorthand for lambda + sub + alogic. We conclude this subsection with a question, whose answer also throws light on the significance of (#**app**):

How many atoms are there, and how fresh can they be?

LEMMA 18.

- lambda $\vdash \exists a, b.\, \mathbf{at}\, a \wedge \mathbf{at}\, b \wedge a\#b$. *('There exist two distinct atoms'.)*
- lambda $\vdash \exists a, b, c.\, \mathbf{at}\, a \wedge \mathbf{at}\, b \wedge \mathbf{at}\, c \wedge a\#b \wedge b\#c \wedge a\#c$. *('There exist three distinct atoms'.)*
- *'There exist n distinct atoms' (we do not write out the predicate in full).*
- lambda $\vdash \exists a.\, a\#x_1 \wedge \cdots \wedge a\#x_n$. *('for any x_1 to x_n, there is an a fresh for them all'.)*

Proof.

- By (**Fresh**) there exists some atom a. We then use (#**app**), taking $x = a = y$. (We cannot use (#**app**) directly, since without (**Fresh**) the empty model is a model of lambda.)
- We use (#**app**) again, taking $x = ba = y$ and using the previous part.
- We just consider the case $n = 4$. We use (#**app**), taking $x = cba = y$.
- For the case $n = 1$ we use (#**app**) with $x = x_1 = y$, to obtain $a\#x_1$. For $n = 2$ we use (#**app**) again with $x = ax_1$ and $y = x_2$ to obtain $b\#a, x_1, x_2$. For $n = 3$ we use (#**app**) with $x = bx_1x_2$ and $y = x_3$ to obtain some $c\#b, x_1, x_2, x_3$, and so on.

■

5.2 Moving from theories of lambda to λ-models

We now show how provably closed terms in lambda correspond to elements of models of the extensional λ-calculus in a more traditional sense (thus giving our axioms some justification).

Let ECA (Extensional Combinatory Algebras) be the FOL theory with signature:

- A binary term-former **application** (we write it infix and invisible, as the name suggests).
- Constants (0-ary term-formers) s and k. Write **i** for **sk**.

— and axioms:

$$\forall x, y.\, \mathsf{k}xy = x \qquad \mathsf{s}xyz = (xy)(xz) \qquad \forall x, y.\, (\forall z.\, xz = yz) \supset x = y.$$

(We do not need extensionality, but the extensional theory is marginally easier to work with.)

An **extensional λ-model** is a FOL model of ECA. This is a traditional notion of model for the λ-calculus [Salibra, 2003b, Selinger, 1997].

We now show how to build a λ-model out of a theory in lambda+sub+alogic. Take any model $\mathbf{Q}$ of lambda+sub+alogic.

Let Λ be the set of all provably closed elements of (the underlying set of) $\mathbf{Q}$. That is, $\Lambda = \{p \in \mathbf{Q} \mid \bullet p\}$.

THEOREM 19. *Λ is an extensional λ-model.*

Proof. We know by Lemma 18 that infinitely many atoms exist in $\mathbf{Q}$, so pick three different ones a, b, c (**different** here means that $a\#b$, $b\#c$, and $a\#c$ are all derivable). Set $\mathsf{s} \equiv \lambda abc.(ab)(ac)$ and $\mathsf{k} \equiv \lambda ab.a$. Here we abuse

notation and use the same notation for elements which are atoms in the model, as for terms of which we can prove **at** — **we do this without comment henceforth**.

We check that:

- $\bullet\lambda abc.(ab)(ac)$ and $\bullet\lambda ab.a$ are derivable.
- $\lambda abc.(ab)(ac) = \lambda a'b'c'.(a'b')(a'c')$ is derivable, for any other choice of three different atoms.
- Likewise $\lambda ab.a = \lambda a'b'.a'$ is derivable.

The first part is from Lemma 20 which follows. The second two parts are by using (α), and the other axioms.

We must verify that $\mathsf{s}pqr = (pq)(pr)$. We verify this using (β) and the rules of sub. Similarly, we can verify $\mathsf{k}pq = p$.

Finally we must verify that with these definitions,

$$\forall p, q.\, (\forall r.\, pr = qr) \supset p = q.$$

This follows directly from (ξ). ■

LEMMA 20. *Assume* $\mathsf{lambda} + \mathsf{sub} + \mathsf{alogic}$ *is in the context. Then:*

$$\mathbf{at}\, a \vdash a\#\lambda a.x \qquad \textit{and} \qquad b\#x,\ \mathbf{at}\, a \vdash b\#\lambda a.x.$$

Proof.

- Fix some u and use Lemma 18 to choose $b\#a, x, u$. By axioms, $\lambda a.x = \lambda b.(x\langle a{\mapsto}b\rangle)$ and
$$(\lambda a.x)\langle a{\mapsto}u\rangle = \lambda b.(x\langle a{\mapsto}b\rangle\langle a{\mapsto}u\rangle) = \lambda b.(x\langle a{\mapsto}b\rangle) = \lambda a.x.$$
- Suppose $\mathbf{at}\, a$ and $b\#x$. If $b = a$ we use the previous part. Otherwise, $b\#a$ by an axiom, we use the last part of Lemma 18 to obtain a fresh c and reason as follows:
$$(\lambda a.x)\langle b{\mapsto}u\rangle = (\lambda c.(x\langle a{\mapsto}c\rangle))\langle b{\mapsto}u\rangle = \lambda c.(x\langle a{\mapsto}c\rangle\langle b{\mapsto}u\rangle) = \lambda c.(x\langle a{\mapsto}c\rangle) = \lambda a.x$$
For the last step, we use the second part of Lemma 12.

■

So informally described, we have observed that elements corresponding to s and k live inside a model of lambda and are provably closed. Application is already in lambda, so we can just take the provably closed elements and quotient by provable equality to get an extensional λ-model.

5.3 Moving back

In the previous subsection we showed how to obtain a model of ECA from a model of lambda. Now we consider going back.

Given a model $\mathbf{Q}$ of ECA there are two issues with obtaining a model of lambda:

- How to add atoms and explicit substitution.
- How to interpret λ.

These problems can be handled in a relatively standard way using a suitable quotient of a free term model over the elements of $\mathbf{Q}$. Extend $\mathbf{Q}$ formally with a disjoint countably infinite set $a, b \in \mathbb{A}$ and close formally under application. We obtain a language $\mathcal{L}$ generated by the grammar

$$l ::= (a \in \mathbb{A}) \mid (q \in \mathbf{Q}) \mid l \cdot l.$$

We may drop $\cdot$ and the bracket; $\cdot$ associates to the left.

Quotient $\mathcal{L}$ by

$$(\mathsf{s})l_1l_2l_3 = (l_1l_3)(l_2l_3), \quad (\mathsf{k})l_1l_3 = l_1, \quad \text{and} \quad (p)(q) = (pq)$$

provided that l_3 does not mention any atoms.

(The final rule says that if p and q are from $\mathbf{Q}$ then (p) applied to (q) in $\mathcal{L}$ is related to the result of applying p to q in $\mathbf{Q}$, injected into $\mathcal{L}$.)

Call this quotiented set $\mathbf{Q}'$. We shall be **lax** about the difference between l, an element of $\mathcal{L}$, and its equivalence class, an element of $\mathbf{Q}'$.

Here is a simple result:

LEMMA 21.

1. *If l does not mention atoms, then $l = (p)$ for some $p \in \mathbf{Q}$.*
2. *$p \neq q$ in $\mathbf{Q}$ if and only if $(p) \neq (q)$ in $\mathbf{Q}'$.*
3. *As a corollary of the last part, $(\mathsf{s}) \neq (\mathsf{k})$ in $\mathbf{Q}'$.*

Proof. For the first part, we notice that in the absence of atoms we may use the final rule above to concatenate the elements of l.

For the second part, we notice that the rules for deducing $l = l'$ simulate the equalities satisfied in $\mathbf{Q}$ anyway. ■

Consider $\mathbb{A}$ as a subset of $\mathbf{Q}'$ by mapping a to (a). We may be **lax** and write a when we mean (a), and more generally we may write p for (p). So for example s in the equation below should be (s).

Let $[\![\mathbf{at}]\!] = \mathbb{A} \subseteq \mathbf{Q}'$.

Define an interpretation of explicit substitution as follows:

- $l\langle a{\mapsto}u\rangle$ is (the equivalence class of) the string obtained by formally replacing every a in l by u.

  It is not hard to show that this is well-defined, i.e. that:

  $$(\mathsf{s}l_1l_2l_3)\langle a{\mapsto}u\rangle = ((l_1l_2)(l_1l_3))\langle a{\mapsto}u\rangle \qquad (\mathsf{k}l_1l_2)\langle a{\mapsto}u\rangle = l_1\langle a{\mapsto}u\rangle$$
  $$((p)(q))\langle a{\mapsto}u\rangle = (pq)\langle a{\mapsto}u\rangle.$$

  For example, for the last part we simply observe that both sides are equal to (pq).

- $p\langle q{\mapsto}u\rangle = p$ for $q \notin \mathbb{A}$ (this is just a convenient default value).

LEMMA 22.

- *$a\#ll'$ if and only if $a\#l$ and $a\#l'$.*
- *$a\#(b)$ if $a \neq b$.*
- *$a\#(a)$ is not the case.*

Therefore, $a\#l$ if and only if a occurs in l,

As a corollary, $\bullet l$ if and only if l does not mention atoms, and $\bullet l$ if and only if $l = (p)$ for some $p \in \mathbf{Q}$.

Proof.

- Direct from the definition of explicit substitution, observing that

  $$(ll')\langle a{\mapsto}l''\rangle = (l\langle a{\mapsto}l''\rangle)(l'\langle a{\mapsto}l''\rangle).$$

- Direct from the definition.
- Direct from the observation that $a\langle a{\mapsto}(\mathsf{s})\rangle = (\mathsf{s})$ and $a\langle a{\mapsto}(\mathsf{k})\rangle = (\mathsf{k})$, and $(\mathsf{s}) \neq (\mathsf{k})$ by a previous part of this result.

For the last part, observe that elements of $\mathbf{Q}'$ have a strict inductive structure as lists, and by induction on structure we conclude that $a\#l$ if and only if a occurs in l.

For the corollary, we use the results just proved and the first part of Lemma 21. ■

LEMMA 23. *Using the interpretations above, $\mathbf{Q}'$ is a model of* sub.

Proof. We just verify that the axioms of sub are valid for the interpretations given. This is quite long and uses the previous parts of this lemma and Lemma 21, but it is also routine, since the interpretation of substition *really is* substitution. ■

We now interpret λ in a standard way for combinatory algebra:

- $\lambda a.a = \mathsf{skk}$.
- $\lambda a.l = \mathsf{k}l$ if $a\#l$.
- Otherwise, $\lambda a.(ll') = \mathsf{s}(\lambda a.l)(\lambda a.l')$.

See [Barendregt, 1984, Definition 7.1.5] or [Selinger, 1997, Subsection 2.2.2] for a futher treatment.

LEMMA 24. *$a\#\lambda a.l$, and $a\#\lambda b.l$ if $a\#l$.*

Proof. By a simple induction on the structure of l. For example, $a\#\mathsf{skk}$ so $a\#\lambda a.a$. ■

We must now verify that this interpretation satisfies the axioms of lambda:

THEOREM 25. *With the interpretations above, $\mathbf{Q}'$ is a model of* alogic + sub + lambda. *That is:*

1. *$\lambda a.l = \lambda b.(l\langle a \mapsto b\rangle)$ if $b\#l$.*
2. *$(\lambda a.l)l' = l\langle a \mapsto l'\rangle$ if $\bullet l'$.*
3. *If $\bullet l$ and $\bullet l'$ then if for all l'' such that $\bullet l''$, $ll'' = l'l''$, then $l = l'$.*
4. *$(\lambda a.l)\langle b \mapsto m\rangle = \lambda a.(l\langle b \mapsto m\rangle)$ if $a\#m$.*
5. *$(ll')\langle b \mapsto m\rangle = (l\langle b \mapsto m\rangle)(l'\langle b \mapsto m\rangle)$.*
6. *For any l and l', there exists some $a\#l, l'$ such that for all b, $b\#all'$ if and only if $b\#a, l, l'$.*

Here we use $a\#l, l'$ as shorthand for $a\#l$ and $a\#l'$, and similarly for $b\#a, l, l'$.

Proof.

1. If $a\#l$ then a does not occur in l so $l\langle a \mapsto b\rangle$ *is* l with b replaced by a. The result follows by induction on l.
2. Suppose l' mentions no atoms. We work by induction on l.
   - $(\lambda a.a)l' = \mathsf{skk}l' = (\mathsf{k}l')(\mathsf{k}l') = l'$ (since l' mentions no atoms; we do not mention this again).
   - If $a\#l$ then $(\lambda a.l)l' = \mathsf{k}ll' = l'$.

- $(\lambda a.l_1 l_2)l' = ((\lambda a.l_1)l')((\lambda a.l_2)l')$. We now use the inductive hypothesis.

■

Recall that we may write lambda when we mean alogic+sub+lambda.

So, lambda really does axiomatise the λ-calculus. We conclude with two more simple observations:

Say a model of lambda is **non-trivial** when there exist two distinct provably closed elements.

THEOREM 26. lambda *has a non-trivial model.*

Proof. ECA has non-trivial models by standard results of the λ-calculus [Barendregt, 1984]. We use the theorem above and Lemma 21. ■

THEOREM 27. *If we travel from* $\mathbf{Q}$ *an extensional* λ*-model to* $\mathbf{Q}'$ *a model of* lambda *and back again as described above, we return to* Q*.*

Proof. By construction Q maps into the model of lambda, with $p \mapsto (p)$. By Lemma 22 this map surjects onto the provably closed elements, and by Lemma 21 the map is an injection. The result follows, since mapping back from $\mathbf{Q}'$ to $\mathbf{Q}$ is just taking closed elements. ■

The converse does not hold: if we travel from $\mathbf{Q}'$ to $\mathbf{Q}$ and back to obtain $\mathbf{Q}''$, it need not be the case that $\mathbf{Q}''$ is identical to $\mathbf{Q}'$; fewer equalities may hold in $\mathbf{Q}''$. For example, consider $\mathbf{Q}'$ equal to terms up to provable equality with

$$\mathsf{lambda},\ \forall a, b.\, \mathbf{at}\, a \wedge \mathbf{at}\, b \supset (\lambda a.a)b = b$$

(and the rest of lambda). This identity does not hold in $\mathbf{Q}''$.

We can say that "the map from $\mathbf{Q}'$ to $\mathbf{Q}$ throws away the *internal meta-language*" (mentioned in §5).

So models of lambda are, in a suitably formal sense, models of the λ-calculus, of perhaps a slightly new kind, since open terms are given a direct semantics too.

6 Other theories

We now return to our example theories.

6.1 Meta-level substitution

Until now we have dealt with two notions of substitution: the **meta-level** substitution $[b \mapsto t]$ and the **object-level** (explicit) substitution $\langle a \mapsto s \rangle$.

How do they interact? As follows:[3]

$$u\langle a{\mapsto}s\rangle[b{\mapsto}t] \equiv u[b{\mapsto}t]\langle a{\mapsto}s[b{\mapsto}t]\rangle.$$

Note that meta-level substitution is an operation on terms, whereas the explicit substitution is a term-former with a theory of equality. This is why $[b{\mapsto}t]$ can distribute down under $\langle a{\mapsto}s\rangle$, without avoiding capture. (The first author builds a λ-calculus of meta-variables based on this idea in [Gabbay, 2005].)

We can internalise this. Extend sub with a binary predicate $\leq$ and axioms:

$$\mathbf{at}\, a \wedge \mathbf{at}\, b \wedge a{\leq}b \wedge b{\leq}a \;\supset\; a{=}b$$
$$\mathbf{at}\, a \wedge \mathbf{at}\, b \wedge \mathbf{at}\, c \wedge a{\leq}b \wedge b{\leq}c \;\supset\; a{\leq}c \qquad \mathbf{at}\, a \supset a{\leq}a.$$

Write $a < b$ for $a \leq b \wedge a \neq b$. Modify $(\sigma\lambda)$ as follows:

$$(\sigma\lambda) \qquad \mathbf{at}\, b \wedge a\#b \wedge a\#y \wedge b \leq a \;\supset\; u\langle a{\mapsto}x\rangle\langle b{\mapsto}y\rangle = s\langle b{\mapsto}y\rangle\langle a{\mapsto}x\langle b{\mapsto}y\rangle\rangle$$

$$(\sigma\lambda') \qquad \mathbf{at}\, b \wedge \mathbf{at}\, b \wedge a < b \;\supset\; u\langle a{\mapsto}x\rangle\langle b{\mapsto}y\rangle = s\langle b{\mapsto}y\rangle\langle a{\mapsto}x\langle b{\mapsto}y\rangle\rangle$$

This endows $a < b$ with the intuition 'b is a meta-variable with respect to a', in the sense that *the characteristic of a meta-variable* is that it may be replaced by any term, possibly capturing names in the term under surrounding bindings.

6.2 'Substitution instance of'

Return to sub as originally presented. Substitution $\langle t{\mapsto}u\rangle$ is not a term, only substitution *applied* to a term, that is $s\langle t{\mapsto}u\rangle$, is a term (and even then, only meaningfully so if $\mathbf{at}\, t$.

Assume term-formers for pairs and lists and use standard notations (s,t), **Nil**, and $hd :: tl$. Introduce a binary term-former $\cdot$ and axioms such that:

$$s \cdot [] = s \qquad s \cdot (a,u) :: tl = s\langle a{\mapsto}u\rangle \cdot tl$$

(here a must be an atom).

Then a list of pairs whose first elements are atoms behaves like an internal notion of substitution. It is easy to detect when a term is a substitution; let σ, σ',... vary over them.

We can then write predicates such as:

[3] In the equation we slipped in a simplification: $a[b{\mapsto}t] \equiv a$.

1. 'σ and σ' are extensionally equal':

$$Eq(\sigma, \sigma') = \forall x.\, x \cdot \sigma = x \cdot \sigma'$$

(since a-logic is untyped, we may wish in practice to restrict equality to some subset of all terms, e.g. those not of the form σ'').

We can then write 's is a substitution instance of t' as $\exists \sigma.\, s = t \cdot \sigma$. Similarly we can write $\sigma \leq \sigma'$ when $\exists \sigma''.\, Eq(\sigma, \sigma'\sigma'')$ (where $\sigma'\sigma''$ denotes list concatenation), and thus 'σ is a (most general) unifier of s and t', and so on.

2. The discussion above extends to predicates in a weaker sense. We can write $a\#P$ to mean $\forall u.\, P\langle a \mapsto u\rangle \Leftrightarrow P$ where here $P\langle a \mapsto u\rangle$ is that predicate obtained by replacing every term s in P by $s\langle a \mapsto u\rangle$. We can express $\bullet P$ as $\forall a.\, \mathbf{at}\, a \supset a\#P$, and so on.

6.3 Logic Programming.

As a-logic gives **at** proof-rules, and can express properties of unification using explicit substitutions, it is interesting to consider logic programming in it. We give a sketch of how this might proceed. Define **a-hereditarily harrop formulae** by

$$G ::= \top \mid p(ts) \mid G \wedge G \mid \mathbf{at}\, t \mid \exists x.G \mid \forall x.G \mid G \vee G \mid D \supset G$$
$$D ::= \top \mid p(ts) \mid D \wedge D \mid G \supset D \mid \mathbf{at}\, t \mid \forall x.D$$

Write $\mathcal{G}$ and $\mathcal{D}$ for the sets of all G and D respectively. Say a derivation is **uniform** when every subderivation concluding a non-atomic goal ends in a right rule or (**FreshL**). Say $(\mathcal{G}, \mathcal{D})$ is a **logic programming language** when for any $G \in \mathcal{G}$ and $\Delta \subseteq \mathcal{D}$, if $\Delta \vdash G$ is derivable then a uniform derivation exists.

THEOREM 28. *a-hereditarily harrop formulae are an abstract logic programming language.*

Proof. We establish commutation rules for those right rules down through those left rules, which can appear in a proof of $\Delta \vdash G$. Most of the proof is as for FOL [Miller *et al.*, 1991]; then by Lemma 6 (**Fresh**) commutes down through any left rule, and it is easily verified that (**at L**) commutes up through any right rule. ■

7 Conclusions and related work

Think of an atom as a 'generic element'; certainly, it is distinguished from other kinds of elements (just like the reader can distinguish an 'arbitrary

triangle' from other kinds of triangles) — but only if we use **at**. This brings out a motivation for this work, from Kit Fine's book 'reasoning with arbitrary elements' [Fine, 1985]. Fine's 'generic elements' are not quite the atoms in a-logic; for a start he does not insist that generic elements exist 'naked' in the semantics, only that there be some notions of dependence and specialisation. However, we were inspired by the idea of a semantics *in which generic elements are first-class members of the denotation.* To us, 'generic element' suggests 'variable', and 'denotation' suggests 'underlying set'.

We were influenced by the logic of generic judgements by Dale Miller and Alwen Tiu [Miller and Tiu, 2003]. This uses higher-order techniques in logic-programming (with a 'definitional' equality [Schroeder-Heister, 1993]) suited to logic-programming but not as readily to model theory; we suspect it is unsuited for our applications. Their system exhibits a quantifier $\nabla x.Px$ which, roughly, reads 'generate a new variable symbol and assert P applied to x, where P has higher type'. While Miller and Tiu seem to insist on there being no semantic content to this (being logic-programmers, they treat the logic as a computational system and it is perfectly reasonable to request it supply a fresh variable symbol; the meaning of this operation is given by the resulting notions of derivability and proof-search, semantics in the sense of maps to sets is irrelevant). The first author was curious to know what ∇ actually *means.* This paper does not study the mathematical relation of a-logic with the logic of generic judgements (but we will come back to ∇ in more detail in a later paper) — however, the underlying idea suggested by what a semantics for ∇ *could* be, is clearly visible in the intuition we suggest for **at** that it identifies an element as being a variable symbol.

We have explored the semantics of a-logic, but not treated them in this paper (except via the FOL theory alogic and standard FOL model theory). We would like to point out that a nice way of *obtaining* a semantics is via FOL model theory, by doing to a-logic itself what we did to the λ-calculus with lambda: build a signature for the syntax of a-logic in a suitable extension of sub, so that the syntax of a-logic can be interpreted in that theory. Write down axioms for implication and universal quantification (which we can do because we have explicit substitution); much as we did for lambda. Prove that formal derivability in a-logic, coincides with the entailment between interpretations in the theory. Then a model of the theory in a-logic, is a model of a-logic. The peculiarity of this notion of model is that variable symbols exist in the underlying set and are represented in the underlying model by atoms.

This paper has considered the λ-calculus as a finitely-axiomatised object-level theory in first-order logic with a finite signature. We were aware of

Salibra's work on Lambda Abstraction Algebras,[4] [Lusin and Salibra, 2004, Salibra, 2003a] which are an axiomatisation of the λ-calculus as an *algebraic* theory (our axiomatisation is not algebraic because of all the hypotheses such as $\mathbf{at}\, a$, $\bullet x$, and $b\#x$), albeit with a infinite signature (roughly, Salibra assumes a unary term-former λa, one for each of a countably infinite set of names a) and infinitely many algebraic equalities (indexed over the a). The parallel with our binary term-former λ (using **at** in axioms to make sure that the first element is an atom) is obvious. However, Salibra can state that names are infinite as a fact of the model, he uses the infinite signature and an appropriate infinite axiom scheme, whereas we have internalised names into the object level and so must introduce a single but non-algebraic axiom (#**app**) to do the same job. In addition, we must introduce conditions $\bullet y$ to avoid inconsistencies (see *wrong* and *inconsistent* on page 19), giving, as we commented there, our open λ-terms the flavour of an *internal meta-language*.

Work by van Benthem and others on Dynamic Logic [van Benthem, 1997] is related to ours. The idea is to treat valuations σ (§3.3) as possible worlds, with an accessibility relation for each variable corresponding to changing the value of σ at x. Substitution of terms for variables becomes movement in a Kripke structure, and universal and existential quantifiers become modalities.

Similar ideas appear in work by Venema [Venema, 1996] and others. Kripke structures are a very general and powerful framework, and a-logic slots nicely into this picture as a logic with a predicate **at** which internalises whether the current world assigns a term to a variable (this is to our knowledge a new idea).

Taking the broader view, there are roughly two ways of enriching FOL with structure — add a new predicate with deduction rules directly, or view the structure as a Kripke frame and add modalities and suitable axioms. In this view, a-logic and Dynamic logic pursue similar goals by different means. This does raise the question of what other aspects of syntax can be internalised in a-logic style.

Michael Beeson created Otter2 [Beeson, 2004, Beeson, 2001] a first-order theorem-prover which extends the term-language of Otter (a first-order theorem-prover [Kalman, 2001]) with λ-terms. He calls the underlying logic 'Lambda-Logic'; which we can think of as a single-sorted FOL with λ-abstraction in the language of terms. His motivation was good proof-

[4] If the reader has not heard of these, but knows about Cylindric Algebras [L. Henkin, , Burris and Sankappanavar, 1981], then they can think of Lambda Abstraction Algebras as infinite-dimensional cylindric algebras for the λ-calculus, and not be too far off the mark.

search principles (from first-order logic) combined with expressivity (from the stronger term-language). Variables cannot be positively identified as such within the logic (which is what **at** does), but a scheme of infinitely many axioms asserts equalities such as $Ap(\lambda x.t, s) = t[x \mapsto s]$ where here Ap is a distinguished binary term-former for application and $t[x \mapsto s]$ is the result of *actually* performing a capture-avoiding substitution of x for s in t.

We can think of the term-language of Lambda-Logic as being λ-terms up to α-equivalence, but not up to β-equivalence. This treatment of higher-orders brings many advantages. The argument (as we interpret it) is that β-reduction is just *too powerful* to put into the language of terms (it is Turing-complete, after all, so unification and matching are undecidable), but still we want functions to practice computer science, so we put in functions but leave β-reduction as an equality to be asserted, rather than being a literal quotient on terms. (Compare this with, say, Isabelle/HOL [Paulson, 2001], which practices higher-order unification.)

In this terminology the theory lambda (§5.1) does the same as Lambda-Logic, only we do not even quotient by α-equivalence. Consequently, λ can be treated as a binary term-former and the whole affair becomes just one theory in FOL enriched with **at** . It may or may not be the case that lambda is well-suited to theorem-proving (it does not delegate α-equivalence to the meta-level, and this might well be a great nuisance), but it would be possible to express the axioms of lambda in *any* theorem-prover with first-order logic, as we have demonstrated.

a-logic draws on Nominal Logic [Pitts, 2003, Gabbay and Cheney, 2004] and to some extent FreshML [FreshML, , Shinwell *et al.*, 2003]. The terminology 'atom' is directly drawn from that work, as is the notion of 'fresh' $\#$. However, in Nominal Logic atoms populate a distinct type, and $\#$ is defined in terms of permutations using the NEW quantifier $\mathcal{N}$ [Gabbay and Pitts, 2001], rather than substitutions using $\forall$. The 'nominal way' gives atoms the character of a collection of atomic constants representing precisely themselves up to identity, rather than a subset of the universe which are atoms representing generic elements. Consistently with the 'atomic' nature of atoms, Nominal Logic and FreshML are designed for *deep embeddings* of syntax in an ambient logic, so that the syntax is an inductive datatype [Azurat and Prasetya, 2002, Wildmoser and Nipkow, 2004]. Thus, terms like $(1+1)$ are equal in the logic when they are provably equal as syntax, and not when they are equal in some denotational sense (so $1+1$ is not equal to 2). Any *other* notions of equality, such as arithmetic equality or β-convertibility, are expressed (probably inductively) as relations. In our implementation, equality is a mix of the two; denotational on closed elements, and syntactic (except for the action of the explicit substitution)

on open ones.

It is important to realise that Nominal Unification [Urban *et al.*, 2004] and Nominal Rewriting [Maribel Fernández, 2004] introduces a two-level hierarchy of variables, 'unknowns' X and 'atoms' a, which is *not* present in Nominal Logic [Pitts, 2003]. This corresponds to the 'meta-level substitution' we considered in this paper in §6.1, though there are technical differences (only substitution, no permutations) and the axiomatisation corresponds more closely to the New Calculus of Contexts [Gabbay, 2005].

We have shown how the λ-calculus can be expressed in first-order logic. Compare this with logic expressed in the λ-calculus. For example Beeson (having extended terms of FOL with λ-abstraction to obtain Lambda Logic) uses the λ-terms to express an object-level logic [Beeson, 1998] (we could do the same here, starting from lambda). Barendregt and others have pursued the idea of taking λ-calculus as *the* foundation, and expressing logic in it (and avoiding inconsistency while doing so, which turns out to be hard) [H. Barendregt, 1998b, H. Barendregt, 1998a] and [Barendregt, 1984, p.560]. Finally, many systems are based directly on higher-order logic [Leivant, 1994, Paulson, 1989, Prawitz, 1964], which simply quotients terms by β-reduction and gains much power as a result (we cannot discuss the trade-offs here [Leivant, 1994, Beeson, 2004]).

To avoid inconsistency in lambda, we restrict β-reduction to st where t is provably closed. We note that this idea is similar to work by Fernández et el [Fernández *et al.*, 2005] (they use a similar condition for a different purpose, to develop efficient reduction strategies and eliminate the need for α-conversion).

This paper is one of very many which try to bring the advantages of the λ-calculus to FOL, and more generally to reconcile them (in new and interesting ways!). We do this in a general way, which we demonstrate would be applicable also to other systems with binders, and we do this working directly in FOL using standard tools.

BIBLIOGRAPHY

[Azurat and Prasetya, 2002] A. Azurat and I.S.W.B. Prasetya. A survey on embedding programming logics in a theorem prover. Technical Report 7, Institute of information and computing sciences, Utrecht University, 2002.

[Barendregt, 1984] H. P. Barendregt. *The Lambda Calculus: its Syntax and Semantics (revised ed.)*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[Beeson, 1998] M. Beeson. Unification in lambda-calculi with if-then-else. In *CADE-15: Proceedings of the 15th International Conference on Automated Deduction*, pages 103–118, 1998.

[Beeson, 2001] M. Beeson. A second-order theorem prover applied to circumscription. *IJCAR*, pages 318–324, 2001.

[Beeson, 2004] M. Beeson. Lambda logic. *IJCAR*, pages 460–474, 2004.

[Bell and Machover, 1977] J. Bell and M. Machover. *A course in mathematical logic*. North-Holland, 1977.

[Burris and Sankappanavar, 1981] S. Burris and H. Sankappanavar. *A Course in Universal Algebra*. Springer, 1981. Also available online.

[D.M.Gabbay, 2005] L.Maksimova D.M.Gabbay. *Interpolation and Definability: Modal and Intuitionistic Logics*. Clarendon Press, 2005.

[Fernández *et al.*, 2005] Fernández, Mackie, and Sinot. Closed reductions: Explicit substitutions without α-conversion. *Mathematical Structures in Computer Science*, 15(2):343–381, 2005.

[Fine, 1985] K. Fine. *Reasoning with Arbitrary Objects*. Blackwell, 1985.

[FreshML,] FreshML. FreshML homepage, . `http://www.freshml.org`.

[Gabbay and Cheney, 2004] M. J. Gabbay and J. Cheney. A sequent calculus for nominal logic. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 139–148, 2004.

[Gabbay and Günthner, 1986] D. Gabbay and F. Günthner, editors. *Handbook of philosophical logic*. Number 166 in Synthese Library. D.Reidel Publishing company, 1986.

[Gabbay and Pitts, 2001] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[Gabbay, 2005] M. J. Gabbay. A new calculus of contexts. In *PPDP*, 2005.

[G.Boolos, 1989] R. Jeffrey G. Boolos. *Computability and Logic*. Cambridge University Press, 1989.

[H. Barendregt, 1998a] M. Bunder H. Barendregt, W. Dekkers. Completeness of the propositions-as-types interpretation of intuitionistic logic into illative combinatory logic. *J. Symbolic Logic*, 3:869–890, 1998.

[H. Barendregt, 1998b] M. Bunder H. Barendregt, W. Dekkers. Completeness of two systems of illative combinatory logic for first-order propositional and predicate calculus. *Archive für Mathematische Logik*, 37:327–341, 1998.

[Johnstone, 1987] P. T. Johnstone. *Notes on logic and set theory*. Cambridge University Press, 1987.

[Kalman, 2001] J. A. Kalman. *Automated Reasoning with Otter*. Rinton Press, 2001.

[L. Henkin,] A. Tarski L. Henkin, J. D. Monk. *Cylindric algebras*. North Holland. Part I (1971), Part II (1985).

[Leivant, 1994] D. Leivant. Higher order logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2, pages 229–322. 1994.

[Lusin and Salibra, 2004] S. Lusin and A. Salibra. The lattice of lambda theories. *Journal of Logic and Computation*, 14 n.3:373–394, 2004.

[Machover, 1996] M. Machover. *Set Theory, Logic and their Limitations*. Cambridge University Press, 1996.

[Maribel Fernández, 2004] I. Mackie M. Fernández, Murdoch J. Gabbay. Nominal rewriting. Submitted, January 2004.

[Miller and Tiu, 2003] D. Miller and A. Tiu. A proof theory for generic judgments: An extended abstract. In *Proceedings of LICS 2003*, pages 118–127. IEEE, June 2003.

[Miller *et al.*, 1991] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[Milner *et al.*, 1992] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, September 1992.

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

[Paulson, 2001] L. Paulson. *The Isabelle reference manual*. Cambridge University Computer Laboratory, February 2001.

[Pitts, 2003] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.

[Prawitz, 1964] D. Prawitz. Completeness and hauptsatz for second-order logic. *Theoria*, 33:246–258, 1964.

[Salibra, 2003a] A. Salibra. Lambda calculus: models and theories. In G. Scollo F. Spoto and A. Nijhol, editors, *Proceedings of the Third AMAST Workshop on Algebraic Methods in Language Processing (AMiLP-2003)*, number 21 in TWLT Proceedings, pages 39–54, University of Twente, 2003. Invited Lecture.

[Salibra, 2003b] A. Salibra. Topological incompleteness and order incompleteness of the lambda calculus. *ACM Trans. Comput. Logic*, 4(3):379–401, 2003.

[Schroeder-Heister, 1993] P. Schroeder-Heister. Definitional reflection and the completion. In *Extensions of Logic Programming*, volume 798 of *Springer Lecture Notes in Artificial Intelligence*, pages 333–347, 1993.

[Selinger, 1997] P. Selinger. *Functionality, polymorphism, and concurrency: a mathematical investigation of programming paradigms.* PhD thesis, University of Pennsylvania, June 1997.

[Shinwell *et al.*, 2003] M. R. Shinwell, A. M. Pitts, and M. J. Gabbay. FreshML: Programming with binders made simple. In *Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), Uppsala, Sweden*, pages 263–274. ACM Press, August 2003.

[Urban *et al.*, 2004] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323:473–497, 2004.

[van Benthem, 1997] J. van Benthem. Modal foundations for predicate logic. *Logic Journal of the IGPL*, 5(2):259–286, 1997.

[van Dalen, 1985] D. van Dalen. Intuitionistic logic. In Gabbay and Günthner, editors, *Handbook of Philosophical Logic vol III*, volume 166 of *Synthèse*, chapter 4. D. Reidel.

[Venema, 1996] Y. Venema. A modal logic of substitution and quantification. *Logic Colloquium '92*, pages 293–309, 1996.

[Wildmoser and Nipkow, 2004] M. Wildmoser and T. Nipkow. Certifying machine code safety: Shallow versus deep embedding. In *TPHOLs*, volume 3223 of *LNCS*, pages 305–320. Springer, 2004.