Homework ;o)

Prove that $\approx$ is an equivalence-relation.

$$\frac{}{a \approx a}\ \approx\text{-atm} \qquad \frac{t_1 \approx s_1 \quad t_2 \approx s_2}{t_1\, t_2 \approx s_1\, s_2}\ \approx\text{-app}$$

$$\frac{t \approx s}{\lambda a.t \approx \lambda a.s}\ \approx\text{-lam}_1 \qquad \frac{t \approx (a\, b)\bullet s \quad a \# s}{\lambda a.t \approx \lambda b.s}\ \approx\text{-lam}_2$$

$$\frac{}{a \# b}\ \#\text{-atm} \qquad \frac{a \# t_1 \quad a \# t_2}{a \# t_1\, t_2}\ \#\text{-app}$$

$$\frac{}{a \# \lambda a.t}\ \#\text{-lam}_1 \qquad \frac{a \# t}{a \# \lambda b.t}\ \#\text{-lam}_2$$

assuming $a \neq b$

Nominal Techniques Course

Tuesday-Lecture

Christian Urban

University of Cambridge

Slides from…

www.cl.cam.ac.uk/∼cu200/

go under 'Recent Talks'

Recap from Yesterday

The proof that did not work:

case $a' \neq a$:

We have

$$\Gamma, a' : \tau', a : \tau_1 \vdash [t]_\alpha : \tau_2$$

$$\Downarrow$$

$$\Gamma, a' : \tau' \vdash [\lambda a.t]_\alpha : \tau_1 \to \tau_2$$

as desired.

case $a' = a$:

We only have

$$\Gamma, a : \tau_1 \vdash [t]_\alpha : \tau_2$$

$$\Downarrow$$

$$\Gamma \vdash [\lambda a.t]_\alpha : \tau_1 \to \tau_2$$

which is not what we want to prove.

Recap from Yesterday (ct.)

Introduced a permutation operation on atoms and λ-terms.

$$
\begin{aligned}
[]\bullet a &\stackrel{\text{def}}{=} a \\
((a_1\, a_2) :: \pi)\bullet a &\stackrel{\text{def}}{=} \begin{cases} a_1 & \text{if } \pi\bullet a = a_2 \\ a_2 & \text{if } \pi\bullet a = a_1 \\ \pi\bullet a & \text{otherwise} \end{cases} \\
\pi\bullet\,(t_1\, t_2) &\stackrel{\text{def}}{=} (\pi\bullet t_1)(\pi\bullet t_2) \\
\pi\bullet(\lambda a.t) &\stackrel{\text{def}}{=} \lambda(\pi\bullet a).(\pi\bullet t_2)
\end{aligned}
$$

This operation behaves much better than renaming substitution.

Recap from Yesterday (ct.)

A relation (or predicate) is **equivariant** provided it is preserved under permutations, that is, its validity is invariant under permutations. For example:

$$t_1 \approx t_2 \quad \text{if and only if} \quad \pi \bullet t_1 \approx \pi \bullet t_2$$

$$a \# t \quad \text{if and only if} \quad \pi \bullet a \# \pi \bullet t$$

$$\Gamma \vdash t : \tau \quad \text{if and only if} \quad \pi \bullet \Gamma \vdash \pi \bullet t : \pi \bullet \tau$$

Recap from Yesterday (ct.)

A relation (or predicate) is **equivariant** provided it is preserved under permutations, that i

permu

These two ideas are codified in **Nominal Logic** introduced by Andrew Pitts. This logic will be the topic for today.

t_1 ≈

$$a \# t \quad \text{if and only if} \quad \pi \bullet a \# \pi \bullet t$$

$$\Gamma \vdash t : \tau \quad \text{if and only if} \quad \pi \bullet \Gamma \vdash \pi \bullet t : \pi \bullet \tau$$

Remember my Criticism?

Barendregt-style Naming Convention roughly says:

> If lambda-terms $M_1,\ldots,M_n$ occur in a certain context, their bound variables are chosen to be different from the free variables.

or (my version)

> Close your eyes and hope everything goes well.

Nominal Logic

Nominal Logic is

- a many-sorted **first-order** logic, i.e.

  the usual inference rules and axioms for logic connectives ($\wedge$, $\vee$, $\Rightarrow$, $\forall$, $=$,...)

  \+ **nominal axioms** (permutations, equivariance,...)

- a **nominal theory** contains additional axioms about the domain at hand (e.g. capture-avoiding substitution for lambda-terms)

Terms

The usual first-order terms (sorted!):

- $x : S$, if x is a variable for sort S
- $f(t_1, \ldots, t_n)$, if $f : S^n \to S$ and $t_i : S_i$

plus

- $(a\,b)\bullet t$, if $a, b : A$ and $t : S$

plus

- $a.t : [A]S$, if $a : A$ and $t : S$

(If you really wanted, you could define abstractions using the terms above.)

Terms

The usual first-order terms (sorted!):

- $x : S$, if x is a variable for sort S
- $f(t_1, \ldots, t_n)$, if $f : S^n \rightarrow S$ and $t_i : S_i$

plus

- $(a\, b) \bullet t$

plus

- $a.t : [A$

(If you reall

abstractions

- sorts are partitioned into two kinds: sorts of atoms A and sorts of data D.
- so sorts are given by:

$$\begin{array}{cccl} S & ::= & A & \text{sorts of atoms} \\ & \mid & D & \text{sorts of data} \end{array}$$

Terms

The usual first-order terms (sorted!):

- $x : S$, if x is a variable for sort S
- $f(t_1, \ldots, t_n)$, if $f : S^n \to S$ and $t_i : S_i$

plus

- $(a\,b){\bullet}t$, if $a, b : A$ and $t : S$

plus

- $a.t : [A]S$, if $a : A$ and $t : S$

(If you really wanted, you could define abstractions using the terms above.)

Terms

The usual first-order terms (sorted!):

- $x : S$, if x is a variable for sort S
- $f(t_1, \ldots, t_n)$, if $f : S^n \to S$ and $t_i : S_i$

plus

- $(a$

$f : S_1 \times \ldots \times S_n \to S$ indicates that function symbol f has arguments sorts S_i and result sort S.

plus

- $a.t : [A]S$, if $a : A$ and $t : S$

(If you really wanted, you could define abstractions using the terms above.)

Terms

The usual first-order terms (sorted!):

- $x : S$, if x is a variable for sort S
- $f(t_1, \ldots, t_n)$, if $f : S^n \rightarrow S$ and $t_i : S_i$

plus

- $(a\,b)\bullet t$, if $a, b : A$ and $t : S$

plus

- $a.t : [A]S$, if $a : A$ and $t : S$

(If you really wanted, you could define abstractions using the terms above.)

Terms

The usual first-order terms (sorted):

Actually sorts are given by:

$$
\begin{array}{rcll}
S & ::= & A & \text{sorts of atoms} \\
  & \mid & D & \text{sorts of data} \\
  & \mid & [A]S & \text{sorts of atom-abstractions}
\end{array}
$$

- x
- f

plus

- $(a\,b)\bullet t$, if $a, b : A$ and $t : S$

plus

- $a.t : [A]S$, if $a : A$ and $t : S$

(If you really wanted, you could define abstractions using the terms above.)

Terms

The usual first-order terms (sorted!):

- $x : S$, if x is a variable for sort S
- $f(t_1, \ldots, t_n)$, if $f : S^n \to S$ and $t_i : S_i$

plus

- $(a\,b)\bullet t$, if $a, b : A$ and $t : S$

plus

- $a.t : [A]S$, if $a : A$ and $t : S$

(If you really wanted, you could define abstractions using the terms above.)

Formulae

Formulae are given by:

- $R(t_1, \ldots, t_n)$, if $R : S^n$ and $t_i : S_i$
- $\neg\varphi$, $\psi \wedge \varphi$, $\psi \vee \varphi$, $\psi \Rightarrow \varphi$, $\psi \Leftrightarrow \varphi$
- $(\forall x : S)\varphi$, $(\exists x : S)\varphi$, if x is of sort S
- $t_1 = t_2$, if $t_1, t_2 : S$

plus

- $a \# t$, if $a : A$ and $t : S$

(plus)

- one (which however can be defined in terms of the formulae given above)

Running Example: λ-Calculus

- Sorts: atom-sort Var, data-sort Trm
- Function-symbols:

$$\begin{aligned} var: &\quad Var \to Trm \\ app: &\quad Trm \times Trm \to Trm \\ lam: &\quad [Var]Trm \to Trm \end{aligned}$$

- Relation-symbol:

$$Subst: \ Trm \times Var \times Trm \times Trm$$

Running Example: λ-Calculus

- Sorts: atom-sort Var, data-sort Trm
- Function-symbols:

$$\begin{aligned} var: &\quad Var \rightarrow Trm \\ app: &\quad Trm \times Trm \rightarrow Trm \\ lam: &\quad [Var]Trm \rightarrow Trm \end{aligned}$$

- Relation-symbol:

$$Subst: \quad Trm \times Var \times Trm \times Trm$$

"$t_1[a := t_2] = t_3$"

Running Example: λ-Calculus

- Sorts: atom-sort Var, data-sort Trm
- Function-symbols:

$$\begin{aligned} var :&\quad Var \rightarrow Trm \\ app :&\quad Trm \times Trm \rightarrow Trm \end{aligned}$$

Alternative for $Subst$:
$Trm \times Var \times Trm \rightarrow Trm$

- Relation-symbol:

$$Subst : \quad Trm \times Var \times Trm \times Trm$$

"$t_1[a := t_2] = t_3$"

Running Example: λ-Calculus

- Sorts: atom-sort Var, data-sort Trm
- Function-symbols:

A formula we might like to prove:

$$(\forall a : Var)(\forall t_1, t_2, t_3 : Trm)$$
$$a \# t_2 \wedge Subst(t_1, a, t_2, t_3) \Rightarrow a \# t_3$$

- Relation-symbol:

$$Subst : \quad Trm \times Var \times Trm \times Trm$$

"$t_1[a := t_2] = t_3$"

Standard Rules and Axioms

for example:

- modus ponens

$$\frac{\psi \Rightarrow \varphi \quad \psi}{\varphi}$$

- excluded middle, symmetry of equality,...

$$\varphi \vee \neg\varphi \qquad\qquad t_1 = t_2 \Rightarrow t_2 = t_1$$

Standard Rules and Axioms

for example:

- modus ponens

Omitted!

$$\varphi$$

- excluded middle, symmetry of equality,...

$$\varphi \vee \neg\varphi \qquad t_1 = t_2 \Rightarrow t_2 = t_1$$

Nominal Axioms

14 axioms—we have to go through them one by one :o(

- 6 about swapping
- 4 about freshness
- 2 about equivariance
- 2 about abstractions

Nominal Axioms

14 axioms—we have to go through them one by one :o(

- 6 about swapping
- 4 about fre
- 2 about equ
- 2 about abs

Quandary: Andy axiomatised the notion of swapping; I prefer permutations and (apart from NL) we shall use permutations only. But I decided to stick with Andy's original presentation.

Properties of Swapping

- S1: $(\forall a : A)(\forall x : S) \;\; (a\,a) \bullet x = x$
- S2: $(\forall a, b : A)(\forall x : S) \;\; (a\,b) \bullet (a\,b) \bullet x = x$
- S3: $(\forall a, b : A) \;\; (a\,b) \bullet a = b$
- E1: $(\forall a, a' : A)(\forall b, b' : A')(\forall x : S)$
  $(a\,a') \bullet (b\,b') \bullet x = ((a\,a') \bullet b \;\, (a\,a') \bullet b') \bullet (a\,a') \bullet x$
- E3: $(\forall a, a' : A)(\forall \vec{t} : \vec{S})$
  $(a\,a') \bullet f(\vec{t}) = f((a\,a') \bullet \vec{t})$
- E5: $(\forall b, b' : A')(\forall a : A)(\forall x : S)$
  $(b\,b') \bullet a.x = ((b\,b') \bullet a).((b\,b') \bullet x)$

Properties of Swapping

- S1: $(\forall a : A)(\forall x : S) \;\; (a\,a)\bullet x = x$
- S2: $(\forall a, b : A)(\forall x : S) \;\; (a\,b)\bullet(a\,b)\bullet x = x$
- S3: $(\forall a,$
- E1: $(\forall a,$ ... $)$
  $(a\,a')\bullet(b$ ... $b')\bullet(a\,a')\bullet x$
- E3: $(\forall a, a' : A)(\forall \vec{t} : \vec{S})$
  $(a\,a')\bullet f(\vec{t}) = f((a\,a')\bullet\vec{t})$
- E5: $(\forall b, b' : A')(\forall a : A)(\forall x : S)$
  $(b\,b')\bullet a.x = ((b\,b')\bullet a).((b\,b')\bullet x)$

From those axioms we can get back the properties of permutations presented yesterday.

Properties of Freshness

- F1: $(\forall a, a' : A)(\forall x : S)$
  $a \mathrel{\#} x \wedge a' \mathrel{\#} x \Rightarrow (a\, a') \bullet x = x$
- F2: $(\forall a, a' : A) \;\; a \mathrel{\#} a' \Leftrightarrow \neg a = a'$
- F3: $(\forall a : A)(\forall b : A') \;\; a \mathrel{\#} b$ where $A \neq A'$
- F4: $(\forall x : S)(\exists a : A) \;\; a \mathrel{\#} x$

Properties of Freshness

- F1: $(\forall a, a' : A)(\forall x : S)$
  $a \# x \wedge a' \# x \Rightarrow (a\, a') \bullet x = x$
- F2: $(\forall a, a' : A)\ \ a \# a' \Leftrightarrow \neg a = a'$
- F3: $(\forall a : A$
- F4: $(\forall x : S$

We shall see in a minute our equality is really 'α-equivalence'.

So this axiom says (roughly): "if a and a' are not free in x, then the permutation preserves α-equivalence."

Properties of Freshness

- F1: $(\forall a, a' : A)(\forall x : S)$
  $a \# x \wedge a' \# x \Rightarrow (a\, a') \bullet x = x$
- F2: $(\forall a, a' : A)\ \ a \# a' \Leftrightarrow \neg a = a'$
- F3: $(\forall a : A)(\forall b : A')\ \ a \# b$ where $A \neq A'$
- F4: $(\forall x : S)(\exists a : A)\ \ a \# x$

Properties of Freshness

- F1: $(\forall a, a' : A)(\forall x : S)$
  $a \# x \wedge a' \# x \Rightarrow (a\, a') \bullet x = x$
- F2: $(\forall a, a' : A) \;\; a \# a' \Leftrightarrow \neg a = a'$
- F3: $(\forall a : A)(\forall b : A') \;\; a \# b$ where $A \neq A'$
- F4: $(\forall x : S)(\exists$

Remember the rule?

$$\frac{}{a \# b}\,\#\text{-atm} \quad \text{where } a \neq b$$

Properties of Freshness

- F1: $(\forall a, a' : A)(\forall x : S)$
  $a \# x \wedge a' \# x \Rightarrow (a\, a') \bullet x = x$
- F2: $(\forall a, a' : A) \;\; a \# a' \Leftrightarrow \neg a = a'$
- F3: $(\forall a : A)(\forall b : A') \;\; a \# b$ where $A \neq A'$
- F4: $(\forall x : S)(\exists a : A) \;\; a \# x$ ← !

Can we prove anything?

Remember the rule

$$\frac{a \neq b \quad a \# t}{a \# \lambda b.t} \#\text{-lam}_2$$

Let's try to prove

$$\neg a = b \wedge a \# x \Rightarrow a \# b.x$$

First two auxiliary facts...

Auxiliary Fact 1

Axioms E{1,3,5} ensure that we can push swappings all the way inside to the variables.

- E1: $(a\,a')\bullet(b\,b')\bullet x =$
  $((a\,a')\bullet b\ \ (a\,a')\bullet b')\bullet(a\,a')\bullet x$
- E3: $(a\,a')\bullet f(\vec{t}\,) = f((a\,a')\bullet\vec{t}\,)$
- E5: $(b\,b')\bullet a.x = ((b\,b')\bullet a).((b\,b')\bullet x)$

Lemma: $(\forall a, a' : A)(\forall\,\vec{x} : \vec{S})$
$(a\,a')\bullet t(\vec{x}) = t((a\,a')\bullet\,\vec{x})$

Proof: By induction on the structure of terms:
$t ::= x \mid f(\vec{t}\,) \mid b.t \mid (b\,b')\bullet t$.

Auxiliary Fact 1

Axioms E{1,3,5} ensure that we can push swappings all the way inside to the variables.

- E1: $(a\,a')\bullet(b\,b')\bullet x =$
  $((a\,a')\bullet b\ (a\,a')\bullet$
- E3: $(a\,a')\bullet f(\vec{t}) = f(($
- E5: $(b\,b')\bullet a.x = ((b\,b')\bullet a).((b\,b')\bullet x)$

all variables of t are assumed to be amongst $\vec{x}$

Lemma: $(\forall a, a' : A)(\forall\, \vec{x} : \vec{S})$
$$(a\,a')\bullet t(\vec{x}) = t((a\,a')\bullet \vec{x})$$

Proof: By induction on the structure of terms:
$$t ::= x \mid f(\vec{t}) \mid b.t \mid (b\,b')\bullet t.$$

Auxiliary Fact 1

Axioms E{1,3,5} ensure that we can push swappings all the way inside to the variables.

- E1: $(a\,a')\bullet(b\,b')\bullet x =$
  $((a\,a')\bullet b\;\;(a\,a')\bullet b')\bullet(a\,a')\bullet x$
- E3: $(a\,a')\bullet f(\vec{t}) = f((a\,a')\bullet\vec{t})$
- E5: $(b\,b')\bullet a.x = ((b\,b')\bullet a).((b\,b')\bullet x)$

Lemma: $(\forall a, a' : A)(\forall\,\vec{x} : \vec{S})$
$(a\,a')\bullet t(\vec{x}) = t((a\,a')\bullet\,\vec{x})$

Proof: By induction on the structure of terms:
$t ::= x \mid f(\vec{t}) \mid b.t \mid (b\,b')\bullet t.$

Auxiliary Fact 1

Axioms E{1,3,5} ensure that we can push swappings all the way inside to the variables.

- E1: $(a\,a')\bullet(b\,b')\bullet x =$
  $((a\,a')\bullet b\;(a\,a$ ...
- E3: $(a\,a')\bullet f(\vec{t}) = f$ ...
- E5: $(b\,b')\bullet a.x = ((b\,b')\bullet a).((b\,b')\bullet x)$

stands for t where all variables x_i are replaced by $(a\,a')\bullet x_i$

Lemma: $(\forall a, a' : A)(\forall\,\vec{x} : \vec{S})$
$$(a\,a')\bullet t(\vec{x}) = t((a\,a')\bullet\,\vec{x})$$

Proof: By induction on the structure of terms:
$$t ::= x \mid f(\vec{t}) \mid b.t \mid (b\,b')\bullet t.$$

Auxiliary Fact 1

Axioms E{1,3,5} ensure that we can push swappings all the way inside to the variables.

- E1: $(a\,a')\bullet(b\,b')\bullet x =$
  $((a\,a')\bullet b\ \ (a\,a')\bullet b')\bullet(a\,a')\bullet x$
- E3: $(a\,a')\bullet f(\vec{t}) = f((a\,a')\bullet\vec{t})$
- E5: $(b\,b')\bullet a.x = ((b\,b')\bullet a).((b\,b')\bullet x)$

Lemma: $(\forall a, a' : A)(\forall\,\vec{x} : \vec{S})$
$(a\,a')\bullet t(\vec{x}) = t((a\,a')\bullet\,\vec{x})$

Proof: By induction on the structure of terms:
$t ::= x \mid f(\vec{t}) \mid b.t \mid (b\,b')\bullet t.$

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \,\#\, \vec{x} \Rightarrow a \,\#\, t(\vec{x})$

Proof:
(1) $\exists a'.a' \,\#\, \vec{x} \wedge a' \,\#\, t(\vec{x})$ by F4

F4:
$(\forall x : S)(\exists a : A) \;\; a \,\#\, x$
'for everything, there exists always a fresh atom'
$a' \,\#\, \vec{x} \wedge a' \,\#\, t(\vec{x}) \Leftrightarrow a' \,\#\, (\vec{x}, t(\vec{x}))$

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'. a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\, a) \bullet x_i = x_i$ by F1

F1:
$a \# x \wedge a' \# x \Rightarrow (a\, a') \bullet x = x$

Auxiliary Fact 2

<u>Lemma</u>: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

<u>Proof</u>:

(1) $\exists a'.a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\,a) \bullet x_i = x_i$ by F1

(3) from $a' \# t(\vec{x})$ (1)
$(a'\,a) \bullet a' \# (a'\,a) \bullet t(\vec{x})$ by (in a minute)

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'.a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\,a){\bullet}x_i = x_i$ by F1

(3) from $a' \# t(\vec{x})$ (1)
$(a'\,a){\bullet}a' \# (a'\,a){\bullet}t(\vec{x})$ by (in a minute)

S3:
$(a\,b){\bullet}a = b$

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'.a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\,a) \bullet x_i = x_i$ by F1

(3′) from $a' \# t(\vec{x})$ (1)
$a \# (a'\,a) \bullet t(\vec{x})$ by (in a minute), S3

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'.a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\,a)\bullet x_i = x_i$ by F1

(3') from $a' \# t(\vec{x})$ (1)
$a \# (a'\,a)\bullet t(\vec{x})$ by (in a minute), S3

Auxiliary Fact 1:

$(a'\,a)\bullet t(\vec{x}) = t((a'\,a)\bullet \vec{x})$

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'. a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a' \, a) \bullet x_i = x_i$ by F1

(3") from $a' \# t(\vec{x})$ (1)
$a \# t((a' \, a) \bullet \vec{x})$ by (in a minute), S3, Aux. F. 1

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'.a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\, a) \bullet x_i = x_i$ by F1

(3") from $a' \# t(\vec{x})$ (1)
$a \# t((a'\, a) \bullet \vec{x})$ by (in a minute), S3, Aux. F. 1

(4) from (3") and (2)
$a \# t(\vec{x})$ by equality

Auxiliary Fact 2

Lemma: $(\forall a : A)(\forall \vec{x} : \vec{S})$
$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

Proof:

(1) $\exists a'.a' \# \vec{x} \wedge a' \# t(\vec{x})$ by F4

(2) from $a \# \vec{x}$ (assumption) and $a' \# \vec{x}$ (1)
$(a'\,a) \bullet x_i = x_i$ by F1

(3") from $a' \# t(\vec{x})$ (1)
$a \# t((a'\,a) \bullet \vec{x})$ by (in a minute), S3, Aux. F. 1

(4) from (3") and (2)
$a \# t(\vec{x})$ by equality

Done.

Now the Proof

Lemma: $\neg a = b \wedge a \# x \Rightarrow a \# b.x$

Now the Proof

Lemma: $\neg a = b \wedge a \# x \Rightarrow a \# b.x$

Proof:

(1) from $\neg a = b$ (assumption)
$a \# b$ by F2

F2:
$a \# a' \Leftrightarrow \neg a = a'$

Now the Proof

Lemma: $\neg a = b \wedge a \# x \Rightarrow a \# b.x$

Proof:

(1) from $\neg a = b$ (assumption)
$a \# b$ by F2

(2) from (1) and $a \# x$ (assumption)
$a \# b.x$ by Aux. F. 2

Auxiliary Fact 2:

$a \# \vec{x} \Rightarrow a \# t(\vec{x})$

we set $\vec{x}$ to b, x
and $t(\vec{x})$ to $b.x$

Now the Proof

<u>Lemma</u>: $\neg a = b \wedge a \# x \Rightarrow a \# b.x$

<u>Proof</u>:

(1) from $\neg a = b$ (assumption)
$a \# b$ by F2

(2) from (1) and $a \# x$ (assumption)
$a \# b.x$ by Aux. F. 2

Done.

Last Four Axioms

- E2: $(\forall a, a' : A)(\forall b : A')(\forall x : S)$
  $b \# x \Rightarrow (a\,a')\bullet b \# (a\,a')\bullet x$
- E4: $(\forall a, a' : A)(\forall \vec{x} : \vec{S})$
  $R(\vec{x}) \Rightarrow R((a\,a')\bullet \vec{x})$
- A1: $(\forall a, a' : A)(\forall x, x' : S)$
  $a.x = a'.x' \Leftrightarrow$
  $(a = a' \wedge x = x') \vee (a \# x' \wedge x = (a\,a')\bullet x')$
- A2: $(\forall y : [A]S)(\exists a : A)(\exists x : S)$
  $y = a.x$

Last Four Axioms

- E2: $(\forall a, a' : A)(\forall b : A')(\forall x : S)$
  $b \# x \Rightarrow (a\, a') \bullet b \# (a\, a') \bullet x$

- E4: $(\forall a, a' : A)(\forall \vec{x} : \vec{S})$
  $R(\vec{x}) \Rightarrow R((a\, a') \bullet \vec{x})$

- A1: $(\forall a, a' : A)(\forall x, x' : S)$
  $a.x = a'.x'$ …
  $(a = a' \wedge x =$ … $\bullet x')$

- A2: $(\forall y : [A]S$ …
  $y = a.x$ …

We require that **all** predicates (and hence formulae) are equivariant. It is **impossible** to formulate a non-equivariant predicate in Nominal Logic.

Last Four Axioms

- E2: $(\forall a, a' : A)(\forall b : A')(\forall x : S)$
  $b \# x \Rightarrow (a\, a') \bullet b \# (a\, a') \bullet x$
- E4: $(\forall a, a' : A)(\forall \vec{x} : \vec{S})$
  $R(\vec{x}) \Rightarrow R((a\, a') \bullet \vec{x})$
- A1: $(\forall a, a' : A)(\forall x, x' : S)$
  $a.x = a'.x' \Leftrightarrow$
  $(a = a' \wedge x = x') \vee (a \# x' \wedge x = (a\, a') \bullet x')$

Remember the two rules?

$$\frac{t \approx s}{\lambda a.t \approx \lambda a.s} \approx\text{-lam}_1 \qquad \frac{a \neq b \quad t \approx (a\, b) \bullet s \quad a \# s}{\lambda a.t \approx \lambda b.s} \approx\text{-lam}_2$$

Last Four Axioms

- E2: $(\forall a, a' : A)(\forall b : A')(\forall x : S)$
  $b \# x \Rightarrow (a\, a')\bullet b \# (a\, a')\bullet x$
- E4: $(\forall a, a' : A)(\forall \vec{x} : \vec{S})$
  $R(\vec{x}) \Rightarrow R((a\, a')\bullet \vec{x})$
- A1: $(\forall a, a' : A)(\forall x, x' : S)$
  $a.x = a'.x' \Leftrightarrow$
  $(a = a' \wedge x = x') \vee (a \# x' \wedge x = (a\, a')\bullet x')$
- A2: $(\forall y : [A]S)(\exists a : A)(\exists x : S)$
  $y = a.x$

Says that all elements of $[A]S$ must be abstractions.

Another Little Proof

<u>Lemma</u>: $(\forall a : A)(\forall x : S)\ a \# a.x$

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

F4:

$(\forall x : S)(\exists a : A)\ \ a \# x$

'for everything, there exists always a fresh atom'

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

Previous Lemma:
$\neg a' = a \wedge a' \# x \Rightarrow a' \# a.x$

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

(3) from (2)
$(a'\, a)\bullet a' \# (a'\, a)\bullet a.x$ by E2

E2:

$b \# x \Rightarrow (a\, a')\bullet b \# (a\, a')\bullet x$

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

(3) from (2)
$(a'\,a)\bullet a' \# (a'\,a)\bullet a.x$ by E2

S3:
$(a\,b)\bullet a = b$

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

(3') from (2)
$a \# (a'\, a)\bullet a.x$ by E2,S3

E5:
$(a'\, a)\bullet a.x = (a'\, a)\bullet a.(a'\, a)\bullet x$

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

(3") from (2)
$a \# a'.(a'\ a) \bullet x$ by E2,S3,E5,S3

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$

> A1:
>
> $a.x = a'.x' \Leftrightarrow$
> $(\ldots) \vee (a \# x' \wedge x = (a\, a') \bullet x')$

(2) from (1)

$a' \# a.x$

(3") from (2)

$a \# a'.(a'\, a) \bullet x$ by E2,S3,E5,S3

(4) now

$a'.(a'\, a) \bullet x = a.x$ by A1

provided: $(a'\, a) \bullet x = (a'\, a) \bullet x$

$a' \# x$

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

(3'') from (2)
$a \# a'.(a'\,a) \bullet x$ by E2,S3,E5,S3

(4) now
$a'.(a'\,a) \bullet x = a.x$ by A1
provided: $(a'\,a) \bullet x = (a'\,a) \bullet x$ by reflexivity
$a' \# x$ by (1)

Another Little Proof

Lemma: $(\forall a : A)(\forall x : S)\ a \# a.x$

Proof:

(1) $\exists a' \# x \wedge a' \# a$ by F4

(2) from (1)
$a' \# a.x$ by Previous Lemma

(3") from (2)
$a \# a'.(a'\,a)\bullet x$ by E2,S3,E5,S3

(4) now
$a'.(a'\,a)\bullet x = a.x$ by A1
provided: $(a'\,a)\bullet x = (a'\,a)\bullet x$ by reflexivity
$a' \# x$ by (1)

(5) So $a \# a.x$. Done.

OK, we are not yet quite there

- Sorts: atom-sort Var, data-sort Trm
- Function-symbols:

$$\begin{array}{rl} var: & Var \to Trm \\ app: & Trm \times Trm \to Trm \\ lam: & [Var]Trm \to Trm \end{array}$$

- Relation-symbol:

$$Subst: \; Trm \times Var \times Trm \times Trm$$

We need to specify a theory for the λ-calculus!

Theory for λ-Calculus

- $(\forall a : Var)(\forall t_1, t_2 : Trm)$
  $var(a) \neq app(t_1, t_2)$
- $(\forall a : Var)(\forall a : [Var]Trm)$
  $var(a) \neq lam(s)$
- $(\forall t_1, t_2 : Trm)(\forall a : [Var]Trm)$
  $app(t_1, t_2) \neq lam(s)$
- $(\forall t : Trm)$

$$\begin{array}{ll} & (\exists a : Var)\ t = var(a) \\ \vee & (\exists t_1, t_2 : Trm)\ t = app(t_1, t_2) \\ \vee & (\exists s : [Var]Trm)\ t = lam(s) \end{array}$$

Theory for λ-Calculus (ct.)

- $(\forall a, a : Var)\ var(a) = var(a') \Rightarrow a = a'$
- $(\forall t_1, t_2, s_1, s_2 : Trm)$
  $app(t_1, t_2) = app(s_1, s_2) \Rightarrow t_1 = s_1 \wedge t_2 = s_2$
- $(\forall s_1, s_2 : [Var]Trm)$
  $lam(s_1) = lam(s_2) \Rightarrow s_1 = s_2$
- $(\forall \vec{x} : \vec{S})$

$$\begin{array}{ll} & (\forall a : Var)\ \varphi(var(a), \vec{x}) \\ \wedge & (\forall t_1, t_2 : Trm)\ \varphi(t_2, \vec{x}) \wedge \varphi(t_1, \vec{x}) \\ & \qquad \Rightarrow \varphi(app(t_1, t_2), \vec{x}) \\ \wedge & (\exists a : Var)\ a \# \vec{x} \wedge (\forall t : Trm)\ \varphi(t, \vec{x}) \\ & \qquad \Rightarrow \varphi(lam(a.t), \vec{x}) \\ \Rightarrow & (\forall t : Trm)\ \varphi(t, \vec{x}) \end{array}$$

Theory for λ-Calculus (ct.)

- $(\forall a, a : Var)\ var(a) = var(a') \Rightarrow a = a'$
- $(\forall t_1, t$ …
  $app(t$ … $\wedge\ t_2 = s_2$
- $(\forall s_1,$ …
  $lam(s_1) = lam(s_2) \Rightarrow s_1 = s_2$

In order to establish a property φ for all lambda-terms, we need to prove it (in the λ-case) for only **one** fresh atom. . . ?!

- $(\forall \vec{x} : \vec{S})$

$$
\begin{array}{ll}
 & (\forall a : Var)\ \varphi(var(a), \vec{x}) \\
\wedge & (\forall t_1, t_2 : Trm)\ \varphi(t_2, \vec{x}) \wedge \varphi(t_1, \vec{x}) \\
 & \qquad\qquad \Rightarrow \varphi(app(t_1, t_2), \vec{x}) \\
\wedge & (\exists a : Var)\ a \# \vec{x} \wedge (\forall t : Trm)\ \varphi(t, \vec{x}) \\
 & \qquad\qquad \Rightarrow \varphi(lam(a.t), \vec{x}) \\
\Rightarrow & (\forall t : Trm)\ \varphi(t, \vec{x})
\end{array}
$$

Some /Any-Property

In Nominal Logic we can prove the following property:

$$(\exists a : A)\ a \# \vec{x} \wedge \varphi(a, \vec{x})$$

$$\Leftrightarrow (\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$$

which asserts that if(f) a formula φ holds for **one** fresh name, then it holds for **every** fresh name.

Some /Any-Property

In Nominal Logic we can prove the following property:

$$(\exists a : A)\ a \# \vec{x} \wedge \varphi(a, \vec{x})$$
$$\Leftrightarrow (\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$$

which asserts that if(f) a formula φ holds for **one** fresh name, then it holds for **every** fresh name.

Proof: $\Rightarrow$ We have:

$$a \# \vec{x},\ a' \# \vec{x} \text{ and } \varphi(a, \vec{x})$$

we need to prove $\varphi(a', \vec{x})$.

From $\varphi(a, \vec{x})$ we have for any swapping (by a fact we prove in a minute—we already assumed it, i.e. equivariance, for predicates):

$$\begin{array}{ll} & \varphi(a, \vec{x}) \\ \Leftrightarrow & \varphi((a\,a')\bullet a, (a\,a')\bullet \vec{x}) \\ \Leftrightarrow & \varphi(a', (a\,a')\bullet \vec{x}) \end{array}$$

Since $a \# \vec{x}$ and $a' \# \vec{x}$

$$\Leftrightarrow \quad \varphi(a', \vec{x})$$

In N… …ving prope…

which … …ds for **one** f… …fresh name.

Proof: $\Rightarrow$ We have:

$$a \# \vec{x},\ a' \# \vec{x} \text{ and } \varphi(a, \vec{x})$$

we need to prove $\varphi(a', \vec{x})$.

Some /Any-Property

In Nominal Logic we can prove the following property:

$$(\exists a : A)\ a \# \vec{x} \wedge \varphi(a, \vec{x})$$
$$\Leftrightarrow (\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$$

which asserts that if(f) a formula φ holds for **one** fresh name, then it holds for **every** fresh name.

Proof: $\Leftarrow$ We have: $(\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$;
we need to prove there exists an a such that:

$$a \# \vec{x} \quad \text{and} \quad \varphi(a, \vec{x})$$

Some/Any-Property

In Nominal Logic we can prove the following property:

We get one such a by axiom F4, which asserts there is always a fresh a.

$$(\exists a$$

$$\Leftrightarrow (\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$$

which asserts that if(f) a formula φ holds for **one** fresh name, then it holds for **every** fresh name.

Proof: $\Leftarrow$ We have: $(\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$;
we need to prove there exists an a such that:

$$a \# \vec{x} \quad \text{and} \quad \varphi(a, \vec{x})$$

Some /Any-Property

In Nominal Logic we can prove the following property:

$$(\exists a : A)\ a \# \vec{x} \wedge \varphi(a, \vec{x})$$
$$\Leftrightarrow (\forall a : A)\ a \# \vec{x} \Rightarrow \varphi(a, \vec{x})$$

which asserts that if(f) a formula φ holds for **one** fresh name, then it holds for **every** fresh name.

Weakening for λ-Calc.

Yes, but in the weakening proof the problem was that we had to prove the property for **all** atoms ($\notin \text{dom}(\Gamma)$), not just **fresh** ones.

Weakening for λ-Calc.

Yes, but in the weakening proof the problem was that we had to prove the property for **all** atoms ($\not\in \mathrm{dom}(\Gamma)$), not just **fresh** ones.

We knew (for the premise):

1. $\varphi(\Gamma, a : \tau_1; [t]_\alpha; \tau_2)$
2. $a \not\in \mathrm{dom}(\Gamma)$

We had to prove:

$\varphi(\Gamma, a' : \tau'; [\lambda a.t]_\alpha; \tau_2)$

for **all** τ' and $a' \not\in \mathrm{dom}(\Gamma)$.

Weakening for λ-Calc.

Yes, but in the weakening proof the problem was that we had to prove the property for **all** atoms ($\not\in \text{dom}(\Gamma)$), not just **fresh** ones.

We knew (for the premise):

1. $\varphi(\Gamma, a : \tau_1; [t]_\alpha; \tau_2)$
2. $a \not\in \text{dom}(\Gamma)$

We had to prove:

$\varphi(\Gamma, a' : \tau'; [\lambda a.t]_\alpha; \tau_2)$

for **all** τ' and $a' \not\in \text{dom}(\Gamma)$.

The problematic case $a = a'$ will now go through smoothly because $a \# lam(a.t)$.

What about Substitution?

We only need to specify the λ-case for a fresh name:

- $Subst(t_1, a, t_2, t_3) \Leftrightarrow$

$$
\begin{aligned}
& t_1 = var(a) \wedge t_2 = t_3 \\
\vee \quad & (\exists a' : Var) \\
& \quad t_1 = var(a') \wedge a \neq a' \wedge t_3 = var(a') \\
\vee \quad & (\exists s_1, s_1', s_2, s_2' : Trm) \\
& \quad t_1 = app(s_1, s_2) \wedge t_3 = app(s_1', s_2') \wedge \\
& \quad Subst(s_1, a, t_2, s_1') \wedge Subst(s_2, a, t, s_2') \\
\vee \quad & (\exists a' : Var)(\exists s_1, s_2 : Trm) \\
& \quad t_1 = lam(a'.s_1) \wedge t_3 = lam(a'.s_2) \wedge \\
& \quad a' \# t_2 \wedge Subst(s_1, a, t_2, s_2)
\end{aligned}
$$

What about Substitution?

We only need to specify the λ-case for a fresh name:

This awfully looks like the definition one often finds for **raw** lambda-terms—which is **not** total.

$$
\begin{array}{rcll}
a\,[a := s] & \stackrel{\text{def}}{=} & s & \\
b\,[a := s] & \stackrel{\text{def}}{=} & b & \text{for } a \neq b \\
(t_1\, t_2)\,[a := s] & \stackrel{\text{def}}{=} & (t_1[a := s])(t_2[a := s]) & \\
(\lambda b.t)\,[a := s] & \stackrel{\text{def}}{=} & \lambda b.(t[a := s]) & \\
\end{array}
$$

for b not occurring freely in s

$$a' \# t_2 \wedge Subst(s_1, a, t_2, s_2)$$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(var) $(\forall a')(\exists t_3) Subst(var(a'), a, t_2, t_3)$

Induction:

1) $(\forall a : Var)\ \varphi(var(a), \vec{x})$
2) $(\forall t_1, t_2 : Trm)\ \varphi(t_2, \vec{x}) \wedge \varphi(t_1, \vec{x}) \Rightarrow \varphi(app(t_1, t_2), \vec{x})$
3) $(\exists a : Var)\ a \# \vec{x} \wedge (\forall t : Trm)\ \varphi(t, \vec{x}) \Rightarrow \varphi(lam(a.t), \vec{x})$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(var) $(\forall a')(\exists t_3) Subst(var(a'), a, t_2, t_3)$

case $a \neq a'$: then $t_3 = t_2$

$a = a'$: then $t_3 = var(a)$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(var) $(\forall a')(\exists t_3) Subst(var(a'), a, t_2, t_3)$

case $a \neq a'$: then $t_3 = t_2$

$a = a'$: then $t_3 = var(a)$

(app) ...

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(lam)

$$(\exists a')\ a' \# (a, t_2) \wedge (\forall t)(\exists t_3) Subst(t, a, t_2, t_3) \Rightarrow (\exists t_3) Subst(lam(a'.t), a, t_2, t_3)$$

Induction:

1) $(\forall a : Var)\ \varphi(var(a), \vec{x})$
2) $(\forall t_1, t_2 : Trm)\ \varphi(t_2, \vec{x}) \wedge \varphi(t_1, \vec{x}) \Rightarrow \varphi(app(t_1, t_2), \vec{x})$
3) $(\exists a : Var)\ a \# \vec{x} \wedge (\forall t : Trm)\ \varphi(t, \vec{x}) \Rightarrow \varphi(lam(a.t), \vec{x})$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(lam)

$$(\exists a')\ a' \# (a, t_2) \wedge (\forall t)(\exists t_3) Subst(t, a, t_2, t_3)$$
$$\Rightarrow (\exists t_3) Subst(lam(a'.t), a, t_2, t_3)$$

$(\exists a')\ a' \# (a, t_2)$ by F4

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(lam)

$$(\exists a')\ a' \# (a, t_2) \wedge (\forall t)(\exists t_3) Subst(t, a, t_2, t_3) \Rightarrow (\exists t_3) Subst(lam(a'.t), a, t_2, t_3)$$

$(\exists a')\ a' \# (a, t_2)$ by F4

We need to prove:

$$Subst(t, a, t_2, t_3) \Rightarrow (\exists t_3) Subst(lam(a'.t), a, t_2, t_3)$$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(lam)

$$(\exists a')\ a' \# (a, t_2) \wedge (\forall t)(\exists t_3) Subst(t, a, t_2, t_3) \Rightarrow (\exists t_3) Subst(lam(a'.t), a, t_2, t_3)$$

$(\exists a')\ a' \# (a, t_2)$ by F4

We need to prove:

$$Subst(t, a, t_2, t_3) \Rightarrow Subst(l'm(a'.t), a, t_2, l'm(a'.t_3))$$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
S

Proof: By in

(lam)
$(\exists a')\ a' \#$

$(\exists a')\ a' \#$

Substitution:

$$Subst(l'm(a'.t), a, t_2, l'm(a'.t_3)) \Leftrightarrow$$
$$\vdots$$
$$\vee \quad (\exists b : Var)(\exists s_1, s_2 : Trm)$$
$$lam(a'.t) = lam(b.s_1) \wedge$$
$$lam(a'.t_3) = lam(b.s_2) \wedge$$
$$b \# t_2 \wedge Subst(s_1, a, t_2, s_2)$$

We need to prove:

$$Subst(t, a, t_2, t_3) \Rightarrow Subst(l'm(a'.t), a, t_2, l'm(a'.t_3))$$

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(lam)

$$(\exists a')\ a' \# (a, t_2) \wedge (\forall t)(\exists t_3) Subst(t, a, t_2, t_3)$$
$$\Rightarrow (\exists t_3) Subst(lam(a'.t), a, t_2, t_3)$$

$(\exists a')\ a' \# (a, t_2)$ by F4

We need to prove:

$$Subst(t, a, t_2, t_3) \Rightarrow Subst(l'm(a'.t), a, t_2, l'm(a'.t_3))$$

Which means:

$a' \# t_2$ and $Subst(t, a, t_2, t_3)$ both by ass.

Still, $Subst$ is Total!

Lemma: $(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

Proof: By induction $(\exists t_3)\ Subst(t_1, a, t_2, t_3)$

(lam)

$(\exists a')$ ... $t_3)$

$(\exists a')$ ... F4

We can even strengthen this lemma and prove
$(\forall a : Var)(\forall t_1, t_2 : Trm)(\exists ! t_3 : Trm)$
$Subst(t_1, a, t_2, t_3)$

We need to prove:

$$Subst(t, a, t_2, t_3) \Rightarrow Subst(l'm(a'.t), a, t_2, l'm(a'.t_3))$$

Which means:

$a' \# t_2$ and $Subst(t, a, t_2, t_3)$ both by ass.

Remember this Property?

We wanted to prove the following property for substitution:

$$(\forall a : Var)(\forall t_1, t_2, t_3 : Trm)$$
$$a \# t_2 \wedge Subst(t_1, a, t_2, t_3) \Rightarrow a \# t_3$$

We need to show:

$$(\exists a')\ a' \# (a, t_2) \wedge$$
$$(\forall t)\ \varphi(t, a, t_2) \Rightarrow \varphi(lam(a'.t), a, t_2)$$

the IH $\varphi(t, a, t_2)$ being:

$$(\forall t_3)\ a \# t_2 \wedge Subst(t_1, a, t_2, t_3) \Rightarrow a \# t_3$$

Remember this Property?

We wanted to prove the following property for substitution:

$$(\forall a : Var)(\forall t_1, t_2, t_3 : Trm)$$
$$a \# t_2 \wedge Subst(t_1, a, t_2, t_3) \Rightarrow a \# t_3$$

We need to show:

$$(\exists a')\ a' \# (a, t_2) \wedge$$
$$(\forall t)\ \varphi(t, a, t_2) \Rightarrow \varphi(lam(a'.t), a, t_2)$$

Which means from $a \# t_2$, $a' \# (a, t_2)$ and

$Subst(lam(a'.t), a, t_2, t_3)$

we need to show $a \# t_3$ using

$(\forall t_3)\, a \# t_2 \wedge Subst(t, a, t_2, t_3) \Rightarrow a \# t_3$

Remember this Property?

We wanted to prove the following property for subs

We know (something to be proved ;o) that if

$$Subst(lam(a'.t), a, t_2, t_3)$$

holds, then there exists an s such that

$$Subst(lam(a'.t), a, t_2, lam(a'.s))$$

holds.

We r

$(\exists a'$

$$(\forall t)\ \varphi(t, a, t_2) \Rightarrow \varphi(lam(a'.t), a, t_2)$$

Which means from $a \# t_2$, $a' \# (a, t_2)$ and

$Subst(lam(a'.t), a, t_2, t_3)$

we need to show $a \# t_3$ using

$(\forall t_3)\ a \# t_2 \wedge Subst(t, a, t_2, t_3) \Rightarrow a \# t_3$

Remember this Property?

We wanted to prove the following property for substitution:

$$(\forall a : Var)(\forall t_1, t_2, t_3 : Trm)$$
$$a \# t_2 \wedge Subst(t_1, a, t_2, t_3) \Rightarrow a \# t_3$$

We need to show:

$$(\exists a')\ a' \# (a, t_2) \wedge$$
$$(\forall t)\ \varphi(t, a, t_2) \Rightarrow \varphi(lam(a'.t), a, t_2)$$

Which means from $a \# t_2$, $a' \# (a, t_2)$ and

$$Subst(lam(a'.t), a, t_2, lam(a'.s))$$

we need to show $a \# lam(a'.s)$ using

$$(\forall t_3)\, a \# t_2 \wedge Subst(t, a, t_2, t_3) \Rightarrow a \# t_3$$

By $a \# a'$

Remember this Property?

We wanted to prove the following property for substitution:

$$(\forall a : Var)(\forall t_1, t_2, t_3 : Trm)$$
$$a \# t_2 \wedge Subst(t_1, a, t_2, t_3) \Rightarrow a \# t_3$$

We need to show:

$$(\exists a')\ a' \# (a, t_2) \wedge$$
$$(\forall t)\ \varphi(t, a, t_2) \Rightarrow \varphi(lam(a'.t), a, t_2)$$

Which means from $a \# t_2$, $a' \# (a, t_2)$ and

$$Subst(lam(a'.t), a, t_2, lam(a'.s))$$

we need to show $a \# s$ using

$$(\forall t_3)\, a \# t_2 \wedge Subst(t, a, t_2, t_3) \Rightarrow a \# t_3$$

Remember this Property?

We ... for

su...

Since $a' \# t_2$ we can turn this into

$$Subst(t, a, t_2, s)$$

instantiate the implication to s and get

$$a \# s$$

... $\# t_3$

We need to show:

$$(\exists a')\ a' \# (a, t_2) \wedge$$
$$(\forall t)\ \varphi(t, a, t_2) \Rightarrow \varphi(lam(a'.t), a, t_2)$$

Which means from $a \# t_2$, $a' \# (a, t_2)$ and

$$Subst(lam(a'.t), a, t_2, lam(a'.s))$$

we need to show $a \# s$ using

$$(\forall t_3)\ a \# t_2 \wedge Subst(t, a, t_2, t_3) \Rightarrow a \# t_3$$

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\, a') \bullet \vec{x})$$

Proof: By Induction.

- Predicates

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

Proof: By Induction.

- Predicates

$$\begin{array}{ll} & R(\vec{x}) \\ \Rightarrow & R((a\,a')\bullet\vec{x}) \end{array} \qquad \text{by E4}$$

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a') \bullet \vec{x})$$

Proof: By Induction.

- Predicates

$$
\begin{array}{lll}
 & R(\vec{x}) & \\
\Rightarrow & R((a\,a') \bullet \vec{x}) & \text{by E4} \\
\Rightarrow & R((a\,a') \bullet (a\,a') \bullet \vec{x}) & \text{by E4}
\end{array}
$$

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

<u>Proof:</u> By Induction.

- Predicates

$$\begin{array}{lll} & R(\vec{x}) & \\ \Rightarrow & R((a\,a')\bullet\vec{x}) & \text{by E4} \\ \Rightarrow & R((a\,a')\bullet(a\,a')\bullet\vec{x}) & \text{by E4} \\ \Rightarrow & R(\vec{x}) & \text{by S2} \end{array}$$

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

Proof: By Induction.

- Predicates

$$\begin{array}{lll} & R(\vec{x}) & \\ \Rightarrow & R((a\,a')\bullet\vec{x}) & \text{by E4} \\ \Rightarrow & R((a\,a')\bullet(a\,a')\bullet\vec{x}) & \text{by E4} \\ \Rightarrow & R(\vec{x}) & \text{by S2} \end{array}$$

- $=, \wedge, \vee, \Rightarrow, \#, \ldots$ easy

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

Proof: By Induction.

- Quantifiers (be careful)

  $$(\exists a : A)\ \varphi(a, \vec{x}) \Leftrightarrow (\exists a : A)\ \varphi(a, (a\,a')\bullet\vec{x})$$

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

Proof: By Induction.

- Quantifiers (be careful)

$$(\exists a : A)\ \varphi(a, \vec{x}) \Leftrightarrow (\exists a : A)\ \varphi(a, (a\,a')\bullet\vec{x})$$

$$\begin{aligned}(\exists a : A)\ a \# a' &\Leftrightarrow (\exists a : A)\ a \# (a\,a')\bullet a' \\ &\Leftrightarrow (\exists a : A)\ a \# a \end{aligned}$$

oops

Loose Ends

The Some/Any-property depended on the equivariance of formulae:

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a')\bullet\vec{x})$$

Proof: By Induction.*

- Quantifiers (be careful)

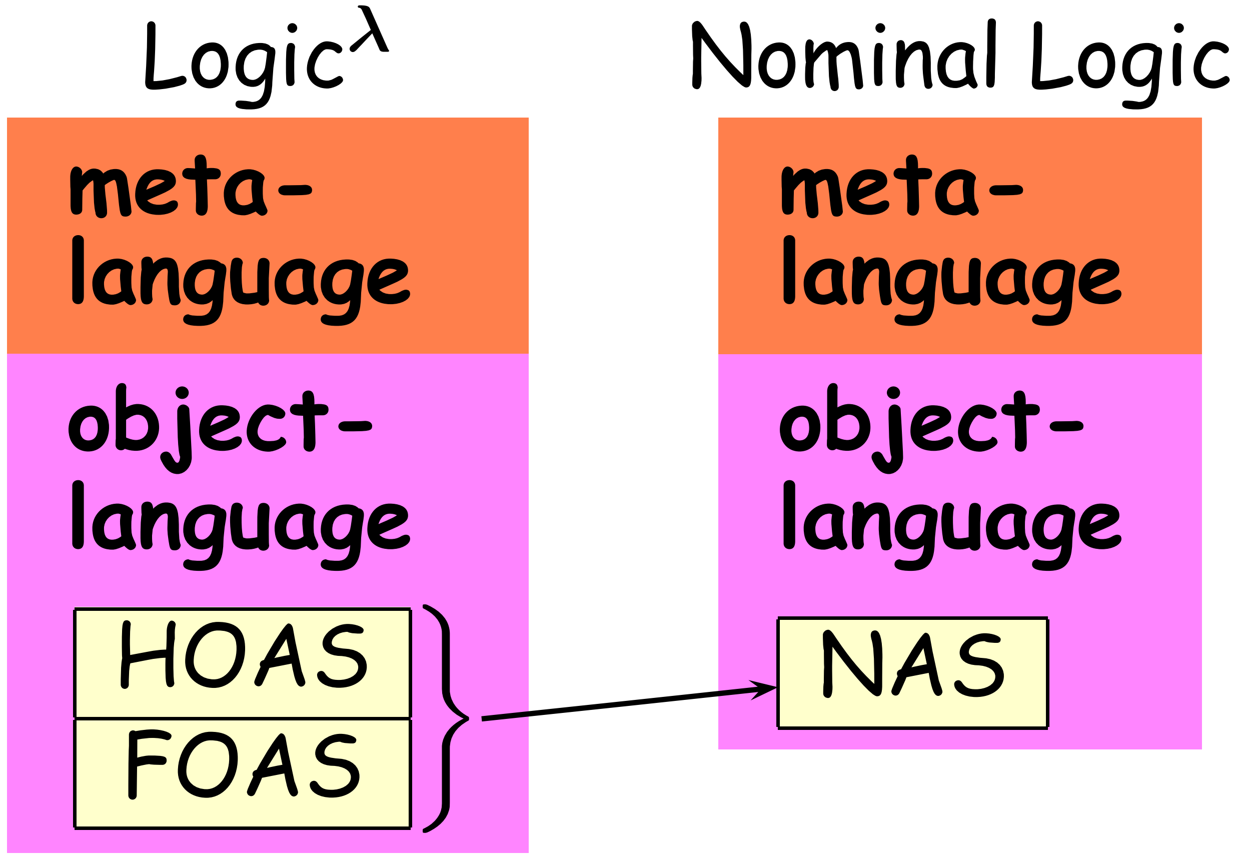
$$(\exists a : A)\ \varphi(a, \vec{x}) \Leftrightarrow (\exists a : A)\ \varphi(a, (a\,a')\bullet\vec{x})$$

$$(\exists a : A)\ a \# a' \Leftrightarrow (\exists a : A)\ a \# (a\,a')\bullet a'$$

$$\Leftrightarrow (\exists a : A)\ a \# a \qquad \text{oops}$$

We need to do renamings on the meta-level.
*I am even prepared to use the sledge-hammer method **once**.

Conclusion for Today



Reasoning about binders is in Nominal Logic as easy (or as hard) as reasoning about first-order abstract syntax using traditional techniques. (Well, marginally harder than FOAS. ;o)

Conclusion for Today

In the meta-language renamings are still necessary, **but**

- they never get in the way of what is being proved (they are also necessary in FOAS), and
- they are the concern of the theorem-tool implementor, **not** the user.

Nominal Logic

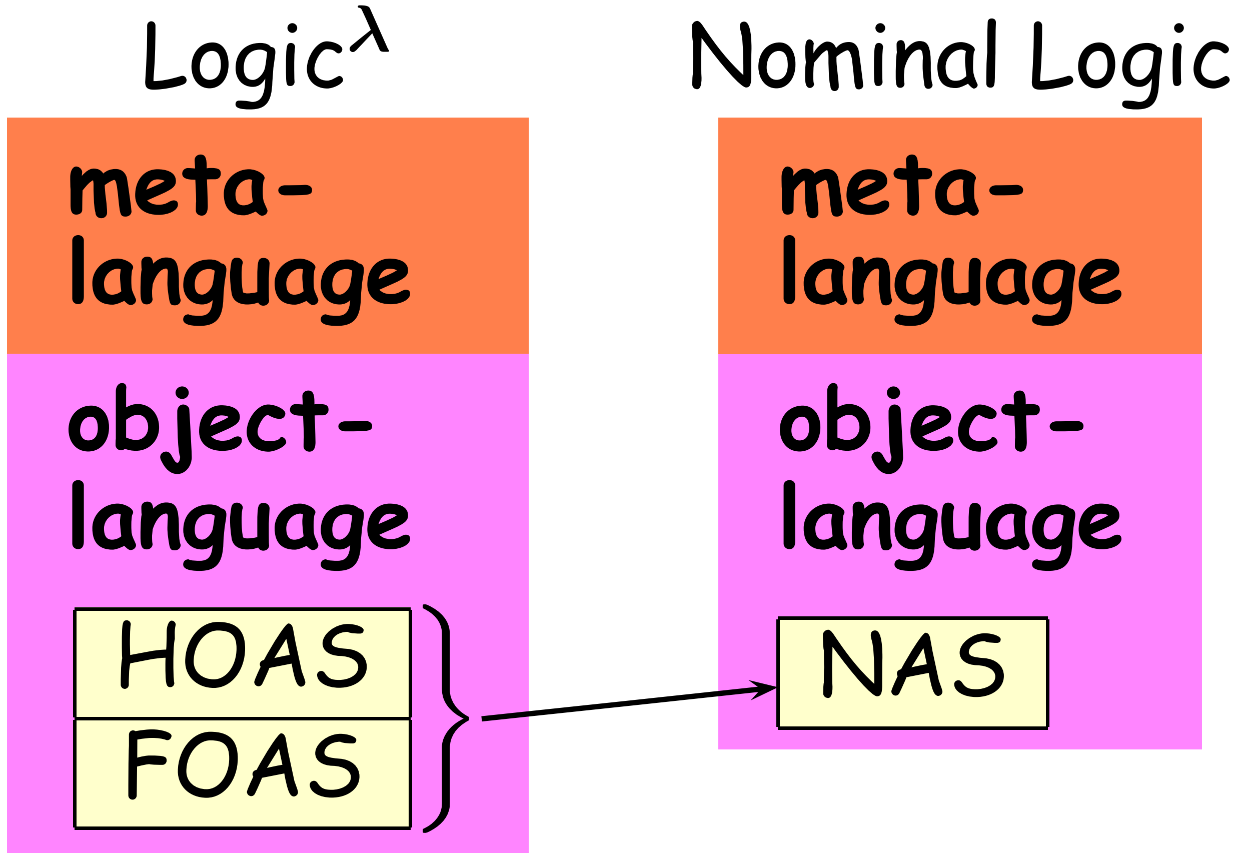
meta-language

object-language

NAS

Nominal Logic as easy (or as hard) as reasoning about first-order abstract syntax using traditional techniques. (Well, marginally harder than FOAS. ;o)

Conclusion for Today



Reasoning about binders is in Nominal Logic as easy (or as hard) as reasoning about first-order abstract syntax using traditional techniques. (Well, marginally harder than FOAS. ;o)