Lecture 7: nominal logic

Freshness quantifier И

Common 'some/any' pattern for reasoning about binders:

> choose some name with a certain property, fresh with respect to the current context; later, context changes and we need to know that any choice of fresh name with the property will do.

Equivariance of the property guarantees this.

Freshness quantifier N

Common 'some/any' pattern for reasoning about binders...

For example, $=_\alpha$ for raw terms over a nominal algebraic signature (Lecture 5)...

Alpha-equivalence

$$=_\alpha \subseteq \Sigma(\mathrm{S}) \times \Sigma(\mathrm{S})$$

$$\frac{a \in \mathbb{A}}{a =_\alpha a} \qquad \frac{t =_\alpha t'}{\mathrm{op}\, t =_\alpha \mathrm{op}\, t'} \qquad \frac{}{() =_\alpha ()}$$

$$\frac{t_1 =_\alpha t_1' \qquad t_2 =_\alpha t_2'}{t_1, t_2 =_\alpha t_1', t_2'}$$

$$\frac{(a_1\ a) \cdot t_1 =_\alpha (a_2\ a) \cdot t_2 \qquad a \# (a_1, t_1, a_2, t_2)}{a_1 . t_1 =_\alpha a_2 . t_2}$$

Freshness quantifier И

Common 'some/any' pattern for reasoning about binders...

For example, $=_\alpha$ for raw terms over a nominal algebraic signature (Lecture 5)...

$$
\begin{aligned}
& (\exists a \# (a_1, t_1, a_2, t_2))\ (a_1\ a) \cdot t_1 =_\alpha (a_2\ a) \cdot t_2 \\
\Rightarrow\ & a_1 \,.\, t_1 =_\alpha a_2 \,.\, t_2 \\
\Rightarrow\ & (\forall a \# (a_1, t_1, a_2, t_2))\ (a_1\ a) \cdot t_1 =_\alpha (a_2\ a) \cdot t_2
\end{aligned}
$$

Freshness quantifier И

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a, x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a, x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a, x) \in S$
3. $\{a \mid (a, x) \in S\}$ is cofinite

Freshness quantifier И

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a, x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a, x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a, x) \in S$
3. $\{a \mid (a, x) \notin S\}$ is finite

Freshness quantifier $\mathcal{N}$

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a, x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a, x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a, x) \in S$
3. $\{a \mid (a, x) \notin S\}$ is finite

Proof. 1⇒2: if $a \# x$ and $(a, x) \in S$, then for any $b \# x$, $(a\ b) \cdot x = x$, so by equivariance of S, $(b, x) = (a\ b) \cdot (a, x) \in S$.

Freshness quantifier $И$

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a, x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a, x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a, x) \in S$
3. $\{a \mid (a, x) \notin S\}$ is finite

Proof. 2⇒3: property 2 gives $\{a \mid (a, x) \notin S\} \subseteq \{a \mid \neg(a \# x)\} = \mathit{supp}\, x$, finite.

Freshness quantifier $\mathcal{N}$

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a, x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a, x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a, x) \in S$
3. $\{a \mid (a, x) \notin S\}$ is finite

Proof. 3⇒1: if $\{a \mid (a, x) \notin S\}$ is finite, then so is $\{a \mid (a, x) \notin S\} \cup \mathit{supp}\, x$, so we can find some $a \in \mathbb{A}$ not in this finite set ($\mathbb{A}$ is infinite!), so 1 holds.

Freshness quantifier И

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a,x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a,x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a,x) \in S$
3. $\{a \mid (a,x) \notin S\}$ is finite

Notation: if $S = \{(a,x) \mid \varphi(a,x)\}$, then we write $(\text{И}a)\ \varphi(a,x)$ to indicate 1/2/3 hold.

E.g. $a_1 \,.\, t_1 =_\alpha a_2 \,.\, t_2 \Leftrightarrow (\text{И}a)\ (a_1\ a) \cdot t_1 =_\alpha (a_2\ a) \cdot t_2$

Freshness quantifier И

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a, x) \in S \Rightarrow (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a, x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a, x) \in S$
3. $\{a \mid (a, x) \notin S\}$ is finite

Notation: if $S = \{(a, x) \mid \varphi(a, x)\}$, then we write $(\text{И}a)\ \varphi(a, x)$ to indicate 1/2/3 hold.

E.g. $(\forall y \in [\mathbb{A}]X)(\text{И}a)(\exists!\, x \in X)\ y = \langle a \rangle x$

Freshness quantifier $И$

Common 'some/any' pattern for reasoning about binders...

Lemma 4. Suppose $X \in \mathbf{Nom}$ and that $S \subseteq \mathbb{A} \times X$ is equivariant: $(\forall \pi)\ (a,x) \in S \ \Rightarrow\ (\pi\, a, \pi \cdot x) \in S$. Then for all $x \in X$, the following are equivalent:

1. $(\exists a)\ a \# x \wedge (a,x) \in S$
2. $(\forall a)\ a \# x \Rightarrow (a,x) \in S$
3. $\{a \mid (a,x) \notin S\}$ is finite

Notation: if $S = \{(a,x) \mid \varphi(a,x)\}$, then we write $\boxed{(Иa)\ \varphi(a,x)}$ to indicate 1/2/3 hold.

$И$ has lots of nice properties, e.g. $\neg(Иa)\,\varphi \ \Leftrightarrow\ (Иa)\,\neg\varphi$. (Exercise)

Nominal first-order logic

- AMP [I& C 186(2003) 165-193]: first-order (hence incomplete) axiomatization of properties of swapping $(a\ b) \cdot (-)$, freshness $a \# (-)$, and the freshness quantifier $(\mathcal{N} a)\ (-)$.
- Gabbay & Cheney [LICS '04], Cheney [JSL 71(2006)299–320]: improved proof theory — sequent calculus with cut elimination.

Nominal logic programming

Cheney & Urban [ACM ToPLaS 30(2008)1–47]:

$$\frac{\text{first-order logic}}{\text{Prolog}} \sim \frac{\text{nominal logic}}{\alpha\text{Prolog}}$$

homepages.inf.ed.ac.uk/jcheney/programs/aprolog

'αProlog is a logic programming language with built-in names, name binding, and unification up to α-equivalence (that is, consistent renaming of bound names.)'

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

Declaration of `subst` corresponds to an inductively defined subset of $\mathbf{\Sigma}(\mathtt{Var.Term,Term,Term})$, for the nominal algebraic signature for λ-terms from Lecture 5:

$$\mathtt{V} : \mathtt{Var} \longrightarrow \mathtt{Term}$$

$$\mathtt{A} : \mathtt{Term,Term} \longrightarrow \mathtt{Term}$$

$$\mathtt{L} : \mathtt{Var.Term} \longrightarrow \mathtt{Term}$$

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

$$\frac{}{\texttt{subst}(a\,.\,\texttt{V}\,a\,,Y\,,Y)}$$

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

$$\frac{a \# X}{\texttt{subst}(a\,.\,X\,,Y\,,X)}$$

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

$$\frac{\mathtt{subst}(a.X, Y, Z) \qquad \mathtt{subst}(a.X', Y, Z')}{\mathtt{subst}(a.\mathtt{A}(X, X'), Y, \mathtt{A}(Z, Z'))}$$

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

$$\frac{\mathtt{subst}(a\,.\,X\,,Y\,,Z) \qquad b\,\#\,Y}{\mathtt{subst}(a\,.\,\mathtt{L}(b\,.\,X)\,,Y\,,\mathtt{L}(b\,.\,Z))}$$

Sample αProlog code

```
id : name_type.  (* variables *)
tm : type.       (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.
```

```
?- subst (b\lam(a\var b)) (var a) X. (* search for X satisfying X = λa.b[a/b] *)
Yes. X = lam(a'\var a) (* X is λa'.a, not λa.a *)
```

As for Prolog, search for solutions to queries involves resolution (try to unify query with head of each clause), but using nominal unification, which solves α-equivalence and freshness constraints.