

# Lecture 7: nominal logic

# Freshness quantifier $\forall$

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 $\mathbb{N}$

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

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

# Alpha-equivalence

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

$$\frac{a \in A}{a =_{\alpha} a}$$

$$\frac{t =_{\alpha} t'}{\text{op } t =_{\alpha} \text{op } t'}$$

$$\frac{}{() =_{\alpha} ()}$$

$$\frac{t_1 =_{\alpha} t'_1 \quad t_2 =_{\alpha} t'_2}{t_1, t_2 =_{\alpha} t'_1, t'_2}$$

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

# Freshness quantifier $\mathbb{N}$

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

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

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

# Freshness quantifier $\mathbb{N}$

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.  $\{a \mid (a, x) \in S\}$  is cofinite

# Freshness quantifier $\mathbb{N}$

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

# Freshness quantifier $\mathbb{N}$

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.**  $1 \Rightarrow 2$ : 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 $\mathbb{N}$

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.**  $2 \Rightarrow 3$ : property 2 gives

$\{a \mid (a, x) \notin S\} \subseteq \{a \mid \neg(a \# x)\} = \text{supp } x$ , finite.

# Freshness quantifier $\mathbb{N}$

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

# Freshness quantifier $\mathbb{N}$

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

**Notation:** if  $S = \{(a, x) \mid \varphi(a, x)\}$ , then we write

$(\mathbb{N}a) \varphi(a, x)$  to indicate 1/2/3 hold.

E.g.  $a_1 \cdot t_1 =_\alpha a_2 \cdot t_2 \Leftrightarrow (\mathbb{N}a) (a_1 a) \cdot t_1 =_\alpha (a_2 a) \cdot t_2$

# Freshness quantifier $\mathbb{N}$

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

**Notation:** if  $S = \{(a, x) \mid \varphi(a, x)\}$ , then we write

$(\mathbb{N}a) \varphi(a, x)$  to indicate 1/2/3 hold.

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

# Freshness quantifier $\mathbb{N}$

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

**Notation:** if  $S = \{(a, x) \mid \varphi(a, x)\}$ , then we write

$(\mathbb{N}a) \varphi(a, x)$  to indicate 1/2/3 hold.

$\mathbb{N}$  has lots of nice properties, e.g.  $\neg(\mathbb{N}a) \varphi \Leftrightarrow (\mathbb{N}a) \neg\varphi$ .

(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  $(\forall a) (-)$ .
- ▶ 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](http://homepages.inf.ed.ac.uk/jcheney/programs/aprolog)

‘ $\alpha$ Prolog is a logic programming language with built-in names, name binding, and unification up to  $\alpha$ -equivalence (that is, consistent renaming of bound names.)’

# Sample $\alpha$ Prolog 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 $\lambda$ Prolog 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  $\Sigma(\text{Var} . \text{Term}, \text{Term}, \text{Term})$ , for the nominal algebraic signature for  $\lambda$ -terms from Lecture 5:

$$V : \text{Var} \rightarrow \text{Term}$$
$$A : \text{Term}, \text{Term} \rightarrow \text{Term}$$
$$L : \text{Var} . \text{Term} \rightarrow \text{Term}$$

# Sample $\alpha$ Prolog 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.
```

---

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

# Sample $\alpha$ Prolog 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.
```

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

# Sample $\alpha$ Prolog 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.
```

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

# Sample $\alpha$ Prolog 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.
```

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

# Sample $\alpha$ Prolog 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 =  $\lambda a.b[a/b]$  *)
Yes. X = lam(a'\var a) (* X is  $\lambda a'.a$ , not  $\lambda a.a$  *)
```

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  $\alpha$ -equivalence and freshness constraints.

$\Sigma(S)$  = raw terms over  $\Sigma$  of sort  $S$

$$\frac{a \in \mathbb{A}}{a \in \Sigma(N)} \quad \frac{t \in \Sigma(S) \quad \text{op} : S \rightarrow D}{\text{op } t \in \Sigma(D)} \quad \frac{}{() \in \Sigma(1)}$$
$$\frac{t_1 \in \Sigma(S_1) \quad t_2 \in \Sigma(S_2)}{t_1, t_2 \in \Sigma(S_1, S_2)} \quad \frac{a \in \mathbb{A} \quad t \in \Sigma(S)}{a.t \in \Sigma(N.S)}$$

Each  $\Sigma(S)$  is a nominal set once equipped with the obvious **Perm**  $\mathbb{A}$ -action—any finite set of atoms containing all those occurring in  $t$  supports  $t \in \Sigma(S)$ .

# Alpha-equivalence

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

$$\frac{a \in A}{a =_{\alpha} a}$$

$$\frac{t =_{\alpha} t'}{\text{op } t =_{\alpha} \text{op } t'}$$

$$\frac{}{() =_{\alpha} ()}$$

$$\frac{t_1 =_{\alpha} t'_1 \quad t_2 =_{\alpha} t'_2}{t_1, t_2 =_{\alpha} t'_1, t'_2}$$

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



# Examples of unification 'mod $\alpha$ '

over the nominal algebraic signature  $\Sigma$  for  $\lambda$ -calculus:  
name-sort **Var**, data-sort **Term**, operations

$V : \text{Var} \rightarrow \text{Term}$

$A : \text{Term}, \text{Term} \rightarrow \text{Term}$

$L : \text{Var} . \text{Term} \rightarrow \text{Term}$

**Ex. 1:** does there exist a  $t \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(t, Vb))) =_{\alpha} L(b . L(a . A(Va, t)))$$

(where  $a \neq b$ )?

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(Vb, t_1))) =_{\alpha} L(a . L(a . A(Va, t_2)))$$

(where  $a \neq b$ )?

# Examples of unification 'mod $\alpha$ '

over the nominal algebraic signature  $\Sigma$  for  $\lambda$ -calculus:  
name-sort **Var**, data-sort **Term**, operations

$V : \text{Var} \rightarrow \text{Term}$

$A : \text{Term}, \text{Term} \rightarrow \text{Term}$

$L : \text{Var} . \text{Term} \rightarrow \text{Term}$

**Ex. 1:** does there exist a  $\lambda$ -term  $e$  with

$$\lambda a. \lambda b. e b = \lambda b. \lambda a. a e$$

(where  $a \neq b$ )?

**Ex. 2:** do there exist  $\lambda$ -terms  $e_1, e_2$  with

$$\lambda a. \lambda b. b e_1 = \lambda a. \lambda a. a e_2$$

(where  $a \neq b$ )?