An Introduction to Nominal Sets

Andrew Pitts

UNIVERSITY OF CAMBRIDGE

Computer Science & Technology

EWSCS 2020
Lecture 3
Outline

L1 Structural recursion and induction in the presence of name-binding operations.

L2 Introducing the category of nominal sets.

L3 Nominal algebraic data types and $\alpha$-structural recursion.

L4 Dependently typed $\lambda$-calculus with locally fresh names and name-abstraction.

References:

AMP, *Nominal Sets: Names and Symmetry in Computer Science*, CUP 2013


Recall: Alpha-equivalence

Smallest binary relation \(\equiv_{\alpha}\) on \(Tr\) closed under the rules:

\[
\begin{align*}
& a \in A \quad t_1 \equiv_{\alpha} t'_1 \
& V a =_{\alpha} V a \quad t_2 \equiv_{\alpha} t'_2 \
& A(t_1, t_2) =_{\alpha} A(t'_1, t'_2) \
& (a \, b) \cdot t =_{\alpha} (a' \, b) \cdot t' \
& b \notin \{a, a'\} \cup \text{var}(t) \cup \text{var}(t') \
& L(a, t) =_{\alpha} L(a', t')
\end{align*}
\]

E.g. \(A(L(a, A(V a, V b)), V c) =_{\alpha} A(L(c, A(V c, V b)), V c)\)
\(\not=_{\alpha} A(L(b, A(V b, V b)), V c)\)

**Fact:** \(\equiv_{\alpha}\) is transitive (and reflexive & symmetric). [Ex. 1]
Freshness

For each nominal set $X$, we can define a relation $\# \subseteq A \times X$ of freshness:

$$a \# x \iff a \not\in \text{supp } x$$
Freshness

For each nominal set $X$, we can define a relation $\# \subseteq A \times X$ of freshness:

$$a \# x \equiv a \notin \text{supp } x$$

- In $\IN$, $a \# n$ always.
- In $A$, $a \# b$ iff $a \neq b$.
- In $\Lambda$, $a \# t$ iff $a \notin \text{fv } t$.
- In $X \times Y$, $a \# (x, y)$ iff $a \# x$ and $a \# y$.
- In $X \rightarrow_{fs} Y$, $a \# f$ can be subtle!
  (and hence ditto for $P_{fs}X$)
Freshness Quantifier

If $\varphi(a)$ is a property of atoms $a \in A$, we write $\forall a, \varphi(a)$ to mean: $\{a \in A \mid \neg \varphi(a)\}$ is finite, i.e. $\varphi(a)$ holds for all but finitely many $a$. 
If $\varphi(a)$ is a property of atoms $a \in A$, we write $\forall a, \varphi(a)$ to mean: $\{a \in A | \neg \varphi(a)\}$ is finite, i.e. $\varphi(a)$ holds for all but finitely many $a$.

**Theorem.** Writing $S = \{a \in A | \varphi(a)\}$, then t.f.a.e.

1. $\forall a, \varphi(a)$
2. $S \in P_{fs}A$ and $\exists a \in A, a \# S \land a \in S$
3. $S \in P_{fs}A$ and $\forall b \in A, b \# S \Rightarrow b \in S$

So can read $\forall a, \varphi(a)$ as

“for some/any fresh $a$, $\varphi(a)$ holds”

**Proof.**
Freshness Quantifier

If \( \varphi(a) \) is a property of atoms \( a \in \mathbb{A} \), we write \( \forall a, \varphi(a) \) to mean: \( \{ a \in \mathbb{A} \mid \neg \varphi(a) \} \) is finite, i.e. \( \varphi(a) \) holds for all but finitely many \( a \).

**Theorem.** Writing \( S = \{ a \in \mathbb{A} \mid \varphi(a) \} \), then t.f.a.e.

1. \( \forall a, \varphi(a) \)
2. \( S \in P_{fs} \mathbb{A} \) and \( \exists a \in \mathbb{A}, a \neq S \land a \in S \)
3. \( S \in P_{fs} \mathbb{A} \) and \( \forall b \in \mathbb{A}, b \neq S \Rightarrow b \in S \)

So can read \( \forall a, \varphi(a) \) as

"for some/any fresh \( a, \varphi(a) \) holds"

**Proof.** If (1), then \( A \triangleq \mathbb{A} - S \) is finite and necessarily supports \( S \) w.r.t. action of \( \mathbb{A} \) on subsets of atoms. Since \( \mathbb{A} \) is infinite and \( A \) finite, there is some \( a \in S = \mathbb{A} - A \); and \( a \neq S \) because \( a \notin A \). So (2) holds.
Freshness Quantifier

If $\varphi(a)$ is a property of atoms $a \in A\backslash$, we write $\forall a, \varphi(a)$ to mean: $\{a \in A\backslash \mid \neg \varphi(a)\}$ is finite, i.e. $\varphi(a)$ holds for all but finitely many $a$.

**Theorem.** Writing $S = \{a \in A\backslash \mid \varphi(a)\}$, then t.f.a.e.

(1) $\forall a, \varphi(a)$

(2) $S \in \mathcal{P}_{fs}A\backslash$ and $\exists a \in A\backslash, a \# S \land a \in S$

(3) $S \in \mathcal{P}_{fs}A\backslash$ and $\forall b \in A\backslash, b \# S \Rightarrow b \in S$

So can read $\forall a, \varphi(a)$ as

“for some/any fresh $a$, $\varphi(a)$ holds”

**Proof.** If (2), say $a \in S$ and $a \# S$, then for any $b$ with $b \# S$, we have $(a b) \cdot S = S$, so $b = (a b) \cdot a \in (a b) \cdot S = S$. So (3) holds.
Freshness Quantifier

If $\varphi(a)$ is a property of atoms $a \in \mathbb{A}$, we write $\forall a, \varphi(a)$ to mean: $\{a \in \mathbb{A} \mid \neg \varphi(a)\}$ is finite, i.e. $\varphi(a)$ holds for all but finitely many $a$.

**Theorem.** Writing $S = \{a \in \mathbb{A} \mid \varphi(a)\}$, then t.f.a.e.

1. $\forall a, \varphi(a)$
2. $S \in P_{fs}\mathbb{A}$ and $\exists a \in \mathbb{A}$, $a \not\in S$ \land a \in S
3. $S \in P_{fs}\mathbb{A}$ and $\forall b \in \mathbb{A}$, $b \not\in S \Rightarrow b \in S$

So can read $\forall a, \varphi(a)$ as
"for some/any fresh $a$, $\varphi(a)$ holds"

**Proof.** If (3), then there is some finite $A \subseteq \mathbb{A}$ supporting $S$ w.r.t. action of $\mathbb{A}$ on subsets of atoms. Since $A$ is finite, to prove (1) it suffices to show $\mathbb{A} \setminus S \subseteq A$, i.e. $\mathbb{A} \setminus A \subseteq S$. But if $b \not\in A$, then because $A$ supports $S$, we have $b \not\in S$ and so by (3) we do have $b \in S$. $\square$
Name abstraction

Each $X \in \textbf{Nom}$ yields a nominal set $[\mathbb{A}]X$ of name-abstractions $\langle a \rangle x$ are $\sim$-equivalence classes of pairs $(a, x) \in \mathbb{A} \times X$, where

$$(a, x) \sim (a', x') \iff \forall b, (b \ a) \cdot x = (b \ a') \cdot x'$$

The $\text{Perm \mathbb{A}}$-action on $[\mathbb{A}]X$ is well-defined by

$$\pi \cdot \langle a \rangle x = \langle \pi(a) \rangle (\pi \cdot x)$$

Fact: $\text{supp}(\langle a \rangle x) = \text{supp} x - \{a\}$, so that

$$b \not\equiv \langle a \rangle x \iff b = a \lor b \not\equiv x$$
Name abstraction

Each $X \in \text{Nom}$ yields a nominal set $[\mathbb{A}]X$ of name-abstractions $\langle a \rangle x$ are $\sim$-equivalence classes of pairs $(a, x) \in A \times X$, where

$$(a, x) \sim (a', x') \iff \forall b, (b a) \cdot x = (b a') \cdot x'$$

We get a functor $[\mathbb{A}](\_): \text{Nom} \to \text{Nom}$ sending $f \in \text{Nom}(X, Y)$ to $[\mathbb{A}]f \in \text{Nom}([\mathbb{A}]X, [\mathbb{A}]Y)$ where

$$[\mathbb{A}]f (\langle a \rangle x) = \langle a \rangle (f x)$$
Name abstraction

$[\Lambda](\cdot) : \text{Nom} \to \text{Nom}$ is a kind of (affine) function space—it is right adjoint to the functor $\Lambda \otimes (\cdot) : \text{Nom} \to \text{Nom}$ sending $X$ to $\Lambda \otimes X = \{(a, x) \mid a \neq x\}$.

Co-unit of the adjunction is ‘concretion’ of an abstraction

$\_ @ \_ : ([\Lambda]X) \otimes \Lambda \to X$

defined by computation rule:

$\forall a, x, \forall b, (\langle a \rangle x) @ b = (b a) \cdot x$

[Ex. 6]
Name abstraction

Generalising concretion, we have the following characterization of morphisms out of $[\{A\} X]$

**Theorem.** $f \in (\{A\} \times X) \rightarrow_{fs} Y$ factors through the subquotient $\{A\} \times X \supseteq \{(a, x) \mid a \neq f\} \rightarrow [\{A\} X]$ to give a unique element of $\overline{f} \in ([\{A\} X] \rightarrow_{fs} Y$ satisfying

$$\forall a, \forall x, \overline{f}(\langle a \rangle x) = f(a, x)$$

iff $f$ satisfies: $\forall a, \forall x, a \neq f(a, x)$.
Initial algebras

\[ A \] (−) has excellent exactness properties. It can be combined with \( \times \), + and \( X \to_{fs} (−) \) to give functors \( T : \text{Nom} \to \text{Nom} \) that have initial algebras

\[ I : T D \to D \]

\[
\begin{array}{ccc}
TD & \to & TX \\
\downarrow I & & \downarrow F \\
D & & X \\
\end{array}
\]
Initial algebras

\[ \mathcal{A}(-) \] has excellent exactness properties. It can be combined with \( \times, + \) and \( X \to_{fs} (-) \) to give functors \( T : \text{Nom} \to \text{Nom} \) that have initial algebras

\[ I : TD \to D \]

\[
\begin{array}{ccc}
TD & \xrightarrow{T\hat{F}} & TX \\
\downarrow I & & \downarrow F \\
D & \xrightarrow{\hat{F}} & X
\end{array}
\]

exists unique \( \hat{F} \)
Initial algebras

- $[\mathcal{A}](\cdot)$ has excellent exactness properties. It can be combined with $\times$, $+$ and $X \rightarrow_{fs} (\cdot)$ to give functors $T : \Nom \rightarrow \Nom$ that have initial algebras $I : TD \rightarrow D$

- For a wide class of such functors (nominal algebraic functors) the initial algebra $D$ coincides with ASTs/$\alpha$-equivalence. E.g. $\Lambda$ is the initial algebra for

\[
T(\cdot) \triangleq \mathcal{A} \cdot \cdot \cdot \times \cdot \cdot \cdot + [\mathcal{A}](\cdot)
\]
Nominal algebraic signatures

- **Sorts** $\mathcal{S} ::= \quad \text{N } \quad \text{name-sort (here just one, for simplicity)}$
  - $\quad \text{D } \quad \text{data-sorts}$
  - $\quad \text{1 } \quad \text{unit}$
  - $\quad \text{S, S } \quad \text{pairs}$
  - $\quad \text{N . S } \quad \text{name-binding}$

- **Typed operations** $\text{op : } \mathcal{S} \rightarrow \mathcal{D}$

Signature $\Sigma$ is specified by the stuff in red.
Nominal algebraic signatures

Example: $\lambda$-calculus

name-sort $\text{Var}$ for variables, data-sort $\text{Term}$ for terms, and operations

\[
\begin{align*}
V & : \text{Var} \rightarrow \text{Term} \\
A & : \text{Term}, \text{Term} \rightarrow \text{Term} \\
L & : \text{Var}. \text{Term} \rightarrow \text{Term}
\end{align*}
\]
Nominal algebraic signatures

Example: $\pi$-calculus

name-sort $\text{Chan}$ for channel names, data sorts $\text{Proc}$, $\text{Pre}$ and $\text{Sum}$ for processes, prefixed processes and summations, and operations

\[
\begin{align*}
S &: \text{Sum} \rightarrow \text{Proc} \\
\text{Comp} &: \text{Proc}, \text{Proc} \rightarrow \text{Proc} \\
\text{Nu} &: \text{Chan}, \text{Proc} \rightarrow \text{Proc} \\
! &: \text{Proc} \rightarrow \text{Proc} \\
P &: \text{Pre} \rightarrow \text{Sum} \\
0 &: \text{1} \rightarrow \text{Sum} \\
\text{Plus} &: \text{Sum}, \text{Sum} \rightarrow \text{Sum} \\
\text{Out} &: \text{Chan}, \text{Chan}, \text{Proc} \rightarrow \text{Pre} \\
\text{In} &: \text{Chan}, (\text{Chan}, \text{Proc}) \rightarrow \text{Pre} \\
\text{Tau} &: \text{Proc} \rightarrow \text{Pre} \\
\text{Match} &: \text{Chan}, \text{Chan}, \text{Pre} \rightarrow \text{Pre}
\end{align*}
\]
Nominal algebraic signatures

Closely related notions:

- **binding signatures** of Fiore, Plotkin & Turi (LICS 1999)
- **nominal algebras** of Honsell, Miculan & Scagnetto (ICALP 2001)

N.B. all these notions of signature restrict attention to iterated, but *unary* name-binding—there are other kinds of lexically scoped binder (e.g. see Pottier’s *Caml* language, or Blanchette *et al* POPL 2019.)
\[ \Sigma(S) = \text{raw terms over } \Sigma \text{ of sort } S \]

Each \( \Sigma(S) \) is a nominal set once equipped with the obvious \( \text{Perm } \mathcal{A} \)-action—any finite set of atoms containing all those occurring in \( t \) supports \( t \in \Sigma(S) \).
Alpha-equivalence $=_{\alpha} \subseteq \Sigma(S) \times \Sigma(S)$

\[
\begin{align*}
  a & \in A \\
  \iff a =_{\alpha} a \\
  t =_{\alpha} t' & \implies \op t =_{\alpha} \op t' \\
  () =_{\alpha} ()
\end{align*}
\]

\[
\begin{align*}
  t_1 =_{\alpha} t'_1 & \quad t_2 =_{\alpha} t'_2 \\
  t_1, t_2 =_{\alpha} t'_1, t'_2
\end{align*}
\]

\[
\begin{align*}
  (a_1 a) \cdot t_1 =_{\alpha} (a_2 a) \cdot t_2 & \quad a \# (a_1, t_1, a_2, t_2) \\
  a_1 \cdot t_1 =_{\alpha} a_2 \cdot t_2
\end{align*}
\]
Alpha-equivalence $=_{\alpha} \subseteq \Sigma(S) \times \Sigma(S)$

**Fact:** $=_{\alpha}$ is equivariant ($t_1 =_{\alpha} t_2 \Rightarrow \pi \cdot t_1 =_{\alpha} \pi \cdot t_2$) and each quotient

$$\Sigma_{\alpha}(S) \triangleq \{ [t]_{\alpha} \mid t \in \Sigma(S) \}$$

is a nominal set with

- $\pi \cdot [t]_{\alpha} = [\pi \cdot t]_{\alpha}$
- $\text{supp} \ [t]_{\alpha} = fn(t)$

where

- $fn(a \cdot t) = fn(t) - \{a\}$
- $fn(t_1, t_2) = fn(t_1) \cup fn(t_2)$

etc.
Theorem. Given a nominal algebraic signature $\Sigma$
(for simplicity, assume $\Sigma$ has a single data-sort $D$ as well as a single
name-sort $N$)
$\Sigma_\alpha(D)$ is an initial algebra for the
associated functor $T_\Sigma : \text{Nom} \to \text{Nom}$.
**Theorem.** Given a nominal algebraic signature $\Sigma$ (for simplicity, assume $\Sigma$ has a single data-sort $D$ as well as a single name-sort $N$)

$\Sigma_{\alpha}(D)$ is an initial algebra for the associated functor $T_{\Sigma} : \text{Nom} \to \text{Nom}$.

$$T_{\Sigma}(-) = [S_1](-) + \cdots + [S_n](-)$$

where $\Sigma$ has operations $\text{op}_i : S_i \to D$ ($i = 1..n$)

and $[S](-) : \text{Nom} \to \text{Nom}$ is defined by:

$$\begin{align*}
[N](-) &= A \\
[D](-) &= (-) \\
[1](-) &= 1 \\
[S_1, S_2](-) &= [S_1](-) \times [S_2](-) \\
[N \cdot S](-) &= [A\backslash][S](-)
\end{align*}$$
Theorem. Given a nominal algebraic signature $\Sigma$ (for simplicity, assume $\Sigma$ has a single data-sort $D$ as well as a single name-sort $N$) $\Sigma_\alpha(D)$ is an initial algebra for the associated functor $T_\Sigma : \text{Nom} \to \text{Nom}$.

E.g. for the $\lambda$-calculus signature with operations

$V : \text{Var} \to \text{Term}$
$A : \text{Term}, \text{Term} \to \text{Term}$
$L : \text{Var} \cdot \text{Term} \to \text{Term}$

we have

$T_\Sigma(-) = A\!\!\!\!\!: + (- \times -) + [A\!\!\!\!](-)$
**Theorem.** Given a nominal algebraic signature $\Sigma$ (for simplicity, assume $\Sigma$ has a single data-sort $D$ as well as a single name-sort $N$)

$\Sigma_\alpha(D)$ is an initial algebra for the associated **enriched functor** $T_\Sigma : \text{Nom} \to \text{Nom}$.

$T_\Sigma$ not only acts on equivariant (=emptily supported) functions, but also on finitely supported functions:

$$(X \to_{fs} Y) \quad \mapsto \quad (T_\Sigma X \to_{fs} T_\Sigma Y)$$

$$F \quad \mapsto \quad T_\Sigma F$$
For $\lambda$-terms:

**Theorem.** Given any $X \in \text{Nom}$ and

$$
\begin{align*}
&f_1 \in A \to_{fs} X \\
&f_2 \in X \times X \to_{fs} X \\
&f_3 \in [A]X \to_{fs} X
\end{align*}
$$

$$
\exists! \hat{f} \in \Lambda \to_{fs} X \quad \text{s.t. } \begin{cases} 
\hat{f} a = f_1 a \\
\hat{f} (e_1 e_2) = f_2 (\hat{f} e_1, \hat{f} e_2) \\
\hat{f} (\lambda a.e) = f_3 (\langle a \rangle (\hat{f} e)) \quad \text{if } a \# (f_1, f_2, f_3)
\end{cases}
$$

The enriched functor $[A](-) : \text{Nom} \to \text{Nom}$ sends $f \in X \to_{fs} Y$ to $[A]f \in [A]X \to_{fs} [A]Y$ where

$$
[A]f \langle a \rangle x = \langle a \rangle (f x) \quad \text{if } a \# f
$$
Recall: Name abstraction

**Theorem.** \( f \in (\mathbb{A} \times X) \rightarrow_{fs} Y \) factors through the subquotient \( \mathbb{A} \times X \supseteq \{(a, x) \mid a \# f\} \rightarrow [\mathbb{A}]X \) to give a unique element of \( \bar{f} \in ([\mathbb{A}]X) \rightarrow_{fs} Y \) satisfying

\[
\forall a, \forall x, \bar{f}(\langle a \rangle x) = f(a, x)
\]

iff \( f \) satisfies: \( \forall a, \forall x, a \# f(a, x) \).
$\alpha$-Structural recursion

For $\lambda$-terms:

**Theorem.**
Given any $X \in \text{Nom}$ and

\[
\begin{align*}
    f_1 & \in A \to_{fs} X \\
    f_2 & \in X \times X \to_{fs} X \quad \text{s.t.} \\
    f_3 & \in A \times X \to_{fs} X \\
\end{align*}
\]

\[
\begin{align*}
\forall a, \forall x, \; a \neq f_3(a, x) \quad \text{(FCB)}
\end{align*}
\]

\[
\exists! \; \hat{f} \in \Lambda \to_{fs} X
\]

s.t.

\[
\begin{align*}
    \hat{f} a &= f_1 a \\
    \hat{f}(e_1 e_2) &= f_2(\hat{f} e_1, \hat{f} e_2) \\
    \hat{f}(\lambda a. e) &= f_3(a, \hat{f} e) \quad \text{if} \; a \neq (f_1, f_2, f_3)
\end{align*}
\]
α-Structural recursion

For λ-terms:

**Theorem.**

Given any $X \in \text{Nom}$ and

\[
\begin{align*}
  f_1 & \in A \to_{fs} X \\
  f_2 & \in X \times X \to_{fs} X \\
  f_3 & \in A \times X \to_{fs} X
\end{align*}
\]

\[
\forall a, \forall x, \ a \not\equiv f_3(a, x) \quad \text{(FCB)}
\]

\[
\exists! \hat{f} \in \Lambda \to_{fs} X \\
\text{s.t. } \begin{cases}
  \hat{f} (e_1 e_2) = f_2(\hat{f} e_1, \hat{f} e_2) \\
  \hat{f}(\lambda a. e) = f_3(a, \hat{f} e) & \text{if } a \not\equiv (f_1, f_2, f_3)
\end{cases}
\]

E.g. capture-avoiding substitution $(-)[e'/a'] : \Lambda \to \Lambda$ is the $\hat{f}$ for

\[
\begin{align*}
  f_1 a & \triangleq \text{if } a = a' \text{ then } e' \text{ else } a \\
  f_2(e_1, e_2) & \triangleq e_1 e_2 \\
  f_3(a, e) & \triangleq \lambda a.e
\end{align*}
\]

for which (FCB) holds, since $a \not\equiv \lambda a.e$
α-Structural recursion

For λ-terms:

**Theorem.** Given any \( X \in \text{Nom} \) and

\[
\begin{align*}
    f_1 & \in \Lambda \rightarrow_{fs} X \\
    f_2 & \in X \times X \rightarrow_{fs} X \quad \text{s.t.} \\
    f_3 & \in \Lambda \times X \rightarrow_{fs} X
\end{align*}
\]

\( \forall a, \forall x, \quad a \not\# f_3(a, x) \quad \text{(FCB)} \)

\( \exists! \hat{f} \in \Lambda \rightarrow_{fs} X \)

\( \hat{f} a = f_1 a \)

s.t.

\( \hat{f} (e_1 e_2) = f_2(\hat{f} e_1, \hat{f} e_2) \)

\( \hat{f}(\lambda a. e) = f_3(a, \hat{f} e) \quad \text{if} \quad a \not\# (f_1, f_2, f_3) \)

E.g. size function \( \Lambda \rightarrow \mathbb{N} \) is the \( \hat{f} \) for

\[
\begin{align*}
    f_1 a & \triangleq 0 \\
    f_2(n_1, n_2) & \triangleq n_1 + n_2 \\
    f_3(a, n) & \triangleq n + 1
\end{align*}
\]

for which (FCB) holds, since \( a \not\# (n + 1) \)
\(\alpha\)-Structural recursion

For \(\lambda\)-terms:

**Theorem.**

Given any \(X \in \text{Nom}\) and

\[
\begin{align*}
  f_1 & \in \Lambda \rightarrow_{fs} X \\
  f_2 & \in X \times X \rightarrow_{fs} X \quad \text{s.t.} \\
  f_3 & \in \Lambda \times X \rightarrow_{fs} X \\
\end{align*}
\]

\[
\forall a, \forall x, \ a \not\# f_3(a, x) \quad \text{(FCB)}
\]

\[
\exists! \hat{f} \in \Lambda \rightarrow_{fs} X \quad \text{s.t.}
\quad \begin{align*}
  \hat{f} a &= f_1 a \\
  \hat{f}(e_1 e_2) &= f_2(\hat{f} e_1, \hat{f} e_2) \\
  \hat{f}(\lambda a.e) &= f_3(a, \hat{f} e) \quad \text{if} \ a \not\# (f_1, f_2, f_3)
\end{align*}
\]

Non-example: trying to list the bound variables of a \(\lambda\)-term

\[
\begin{align*}
  f_1 a & \triangleq \text{nil} \\
  f_2(\ell_1, \ell_2) & \triangleq \ell_1 \circ \ell_2 \\
  f_3(a, \ell) & \triangleq a :: \ell
\end{align*}
\]

for which (FCB) does not hold, since \(a \in \text{supp}(a :: \ell)\).
\(\alpha\)-Structural recursion

For \(\lambda\)-terms:

**Theorem.**

Given any \(X \in \text{Nom}\) and \(\begin{cases} f_1 & \in \Lambda \rightarrow_{fs} X \\ f_2 & \in X \times X \rightarrow_{fs} X \\ f_3 & \in \Lambda \times X \rightarrow_{fs} X \end{cases}\)

\[
\forall a, \forall x, \ a \neq f_3(a, x) \quad (\text{FCB})
\]

\[\exists! \hat{f} \in \Lambda \rightarrow_{fs} X \begin{cases} \hat{f} a = f_1 a \\ \hat{f}(e_1 e_2) = f_2(\hat{f} e_1, \hat{f} e_2) \\ \hat{f}(\lambda a.e) = f_3(a, \hat{f} e) \quad \text{if} \ a \neq (f_1, f_2, f_3) \end{cases}\]


Implemented in Urban & Berghofer’s Nominal package for Isabelle/HOL (classical higher-order logic).

Seems to capture informal usage well, but (FCB) can be tricky…
Counting occurrences of bound variables

For each $e \in \Lambda$, \[ \text{cbv } e \triangleq f \ e \ \rho_0 \in \text{IN} \]
where we want $f \in \Lambda \rightarrow_{fs} X$ with $X = (\Lambda \rightarrow_{fs} \text{IN}) \rightarrow_{fs} \text{IN}$ to satisfy

\[
\begin{align*}
    f \ a \ \rho &= \rho \ a \\
    f \ (e_1 \ e_2) \ \rho &= (f \ e_1 \rho) + (f \ e_2 \rho) \\
    f(\lambda a.e) \ \rho &= f \ e \ (\rho[a \mapsto 1])
\end{align*}
\]

and where $\rho_0 \in \Lambda \rightarrow_{fs} \text{IN}$ is $\lambda(a \in \Lambda) \rightarrow 0$.

E.g. when $e = (\lambda a. \lambda b. a) \ b$ (with $a \neq b$), then $e$ has a single occurrence of a bound variable (called $a$) and $\text{cbv } e = 1$. 
Counting occurrences of bound variables

For each \( e \in \Lambda \), \( \text{cbv} e \triangleq f e \rho_0 \in \mathbb{IN} \)

where we want \( f \in \Lambda \rightarrow_{fs} X \) with \( X = (\Lambda \rightarrow_{fs} \mathbb{IN}) \rightarrow_{fs} \mathbb{IN} \) to satisfy

\[
\begin{align*}
    f a \rho &= \rho a \\
    f (e_1 e_2) \rho &= (f e_1 \rho) + (f e_2 \rho) \\
    f(\lambda a.e) \rho &= f e (\rho[a \mapsto 1])
\end{align*}
\]

and where \( \rho_0 \in \Lambda \rightarrow_{fs} \mathbb{IN} \) is \( \lambda(a \in \Lambda) \rightarrow 0 \).

Looks like we should take \( f_3(a, x) = \lambda(\rho \in \Lambda \rightarrow_{fs} \mathbb{IN}) \rightarrow x(\rho[a \mapsto 1]) \), but this does not satisfy (FCB). Solution: take \( X \) to be a certain nominal subset of \( (\Lambda \rightarrow_{fs} \mathbb{IN}) \rightarrow_{fs} \mathbb{IN} \). [See Nominal Sets book, Example 8.20]