Lecture 2

To be explained:

- Nominal sets, support and the freshness relation, $(-) \# (-)$.
- How is α-structural recursion proved?
- How to generalise α-structural recursion from the example language Λ to general languages with binders?
- What's involved with applying α-structural recursion in any particular case?
- Example: normalisation by evaluation.
- Machine-assisted support?

Nominal sets

<u>Definition</u>. A finite subset $A \subseteq \mathbb{A}$ **supports** an element $s \in S$ of a $Perm$-set S if

$$(a\,a') \cdot s = s$$

holds for all $a, a' \in \mathbb{A}$ (of same sort) <u>not in A</u>

Nominal sets

- A nominal set is a set equipped with an action of the group $Perm$, all of whose elements have a finite support.
- A morphism of nominal sets $f : X \to X'$ is an equivariant function, i.e. a function that preserves the $Perm$-set action:

$$(\forall \pi \in Perm)(\forall x \in X)\ f(\pi \cdot x) = \pi \cdot (f\, x)$$

The category of nominal sets is equivalent to a well-known boolean topos (= model of classical higher-order logic). We just need to see some of that structure...

Discrete nominal sets

The trivial action of $Perm$ on any set S is given by: $\pi \cdot s = s$

Note that with respect to this action each $s \in S$ is supported by $\emptyset$.

We call S + trivial action the discrete nominal set on S.

Discrete nominal sets

The trivial action of $Perm$ on any set S is given by:
$\pi \cdot s = s$
Note that with respect to this action each $s \in S$ is supported by $\emptyset$.

We call S + trivial action the discrete nominal set on S.

$$\text{booleans} \quad \boxed{\mathbb{B} \triangleq \{true, false\}}$$

$$\text{natural numbers} \quad \boxed{\mathbb{N} \triangleq \{0, 1, 2, \ldots\}}$$

will be regarded as nominal sets in this way.

Nominal sets of atoms

We make $\mathbb{A}$ a $\boldsymbol{Perm}$-set via action $\boldsymbol{\pi \cdot a = \pi(a)}$.

By definition of $\boldsymbol{Perm}$, this action restricts to atoms of any particular sort, e.g. to $\mathbb{V}$.

It is not hard to see that the support of each atom $\boldsymbol{a}$ is just $\boldsymbol{\{a\}}$.

Products of nominal sets

If X_1 and X_2 are nominal sets, we get a $Perm$-action on

$$X_1 \times X_2 \triangleq \{(x_1, x_2) \mid x_1 \in X_1 \;\&\; x_2 \in X_2\}$$

by defining

$$\pi \cdot (x_1, x_2) \triangleq (\pi \cdot x_1, \pi \cdot x_2)$$

and then every pair in $X_1 \times X_2$ is finitely supported. In fact (exercise)

$$supp((x_1, x_2)) = supp(x_1) \cup supp(x_2)$$

Nominal function sets

The exponential of X and X' in the category of $Perm$-sets is the set of all functions $f : X \to X'$, equipped with the $Perm$-action:

$$\boxed{\begin{array}{rcl} \pi \cdot f : X & \to & X' \\ x & \mapsto & \pi \cdot (f(\pi^{-1} \cdot x)) \end{array}}$$

With this definition, $\pi \cdot (-)$ preserves function application:

$$\begin{aligned} (\pi \cdot f)(\pi \cdot x) &= \pi \cdot (f(\pi^{-1} \cdot (\pi \cdot x))) \\ &= \pi \cdot (f(\iota \cdot x)) \\ &= \pi \cdot (f\, x) \end{aligned}$$

Nominal function sets

The exponential of X and X' in the category of Perm-sets is the set of all functions $f : X \to X'$, equipped with the Perm-action:

$$\boxed{\begin{array}{rcl} \pi \cdot f : X & \to & X' \\ x & \mapsto & \pi \cdot (f(\pi^{-1} \cdot x)) \end{array}}$$

Even if X and X' are nominal, not every function from X to X' is necessarily finitely supported w.r.t. this action.

Exercise: any surjection $\mathbb{N} \to \mathbb{V}$ cannot have finite support.

Nominal function sets

The exponential of X and X' in the category of $Perm$-sets is the set of all functions $f : X \to X'$, equipped with the $Perm$-action:

$$\boxed{\begin{array}{rcl} \pi \cdot f : X & \to & X' \\ x & \mapsto & \pi \cdot (f(\pi^{-1} \cdot x)) \end{array}}$$

The set $\boxed{X \to_{\mathrm{fs}} X'}$ of finitely supported functions from a nominal set X to a nominal set X' is, by construction, a nominal set.

Nominal function sets

The exponential of X and X' in the category of $Perm$-sets is the set of all functions $f : X \to X'$, equipped with the $Perm$-action:

$$\boxed{\begin{array}{rcl} \pi \cdot f : X & \to & X' \\ x & \mapsto & \pi \cdot (f(\pi^{-1} \cdot x)) \end{array}}$$

The set $\boxed{X \to_{\mathrm{fs}} X'}$ of finitely supported functions from a nominal set X to a nominal set X' is, by construction, a nominal set.

Exercise: show that $f \in (X \to_{\mathrm{fs}} X')$ satisfies $supp(f) = \emptyset$ iff f is an equivariant function.

Finitely supported subsets of a nominal set

If X is a nominal set, we get a $Perm$-action on the set of all subsets $S \subseteq X$ by defining:

$$\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$$

As for functions, not every $S \subseteq X$ is finitely supported w.r.t. this action.

Finitely supported subsets of a nominal set

If X is a nominal set, we get a $Perm$-action on the set of all subsets $S \subseteq X$ by defining:

$$\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$$

As for functions, not every $S \subseteq X$ is finitely supported w.r.t. this action.

(e.g. take $X = \mathbb{A}$, enumerate it and let S consist of the even-numbered atoms.)

Finitely supported subsets of a nominal set

If X is a nominal set, we get a $Perm$-action on the set of all subsets $S \subseteq X$ by defining:

$$\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$$

As for functions, not every $S \subseteq X$ is finitely supported w.r.t. this action.

We write $\boxed{P_{\mathrm{fs}}(X)}$ for the nominal set of finitely supported subsets of X.

Finitely supported subsets of a nominal set

If X is a nominal set, we get a $Perm$-action on the set of all subsets $S \subseteq X$ by defining:

$$\pi \cdot S \triangleq \{\pi \cdot x \mid x \in S\}$$

As for functions, not every $S \subseteq X$ is finitely supported w.r.t. this action.

We write $\boxed{P_{\mathrm{fs}}(X)}$ for the nominal set of finitely supported subsets of X.

($P_{\mathrm{fs}}(X)$ is isomorphic to $X \to_{\mathrm{fs}} \mathbb{B}$.)

(Choice functions)

Theorem No function $ch : P_{\mathrm{fs}}\mathbb{V} \longrightarrow \mathbb{V}$ satisfying

$$(\forall S \in P_{\mathrm{fs}}\mathbb{V})\ S \neq \emptyset \ \Rightarrow\ ch\, S \in S$$

can have finite support.

Proof Suppose such a ch is supported by a finite subset $A \subseteq \mathbb{V}$ and derive a contradiction.

(Choice functions)

<u>Theorem</u> No function $ch : P_{\mathrm{fs}}\mathbb{V} \longrightarrow \mathbb{V}$ satisfying

$$(\forall S \in P_{\mathrm{fs}}\mathbb{V})\ S \neq \emptyset \ \Rightarrow\ ch\, S \in S$$

can have finite support.

<u>Proof</u> Suppose such a ch is supported by a finite subset $A \subseteq \mathbb{V}$. Let $S \triangleq \mathbb{V} - A$.

(Choice functions)

Theorem No function $ch : P_{\mathrm{fs}}\mathbb{V} \longrightarrow \mathbb{V}$ satisfying

$$(\forall S \in P_{\mathrm{fs}}\mathbb{V})\ S \neq \emptyset \ \Rightarrow\ ch\, S \in S$$

can have finite support.

Proof Suppose such a ch is supported by a finite subset $A \subseteq \mathbb{V}$. Let $S \triangleq \mathbb{V} - A$.

So $S \in P_{\mathrm{fs}}(\mathbb{V})$, $supp(S) = A$ & $S \neq \emptyset$.

So $a_0 \triangleq ch\, S \in S \triangleq \mathbb{V} - A$ & hence $a_0 \mathrel{\#} (ch, S)$.

(Choice functions)

<u>Theorem</u> No function $ch : P_{\mathrm{fs}}\mathbb{V} \to \mathbb{V}$ satisfying

$$(\forall S \in P_{\mathrm{fs}}\mathbb{V})\ S \neq \emptyset \ \Rightarrow \ ch\, S \in S$$

can have finite support.

<u>Proof</u> Suppose such a ch is supported by a finite subset $A \subseteq \mathbb{V}$. Let $S \triangleq \mathbb{V} - A$.

So $S \in P_{\mathrm{fs}}(\mathbb{V})$, $supp(S) = A$ & $S \neq \emptyset$.

So $a_0 \triangleq ch\, S \in S \triangleq \mathbb{V} - A$ & hence $a_0 \,\#\, (ch, S)$.

Pick any $a_1 \,\#\, (ch, S, a_0)$.

Then $a_1 = (a_0\, a_1) \cdot a_0 \triangleq (a_0\, a_1) \cdot (ch\, S) = ((a_0\, a_1) \cdot ch)((a_0\, a_1) \cdot S) = ch\, S \triangleq a_0$, contradicting $a_0 \neq a_1$. $\square$

Running example (reminder)

Concrete syntax:

$$t ::= a \mid t\,t \mid \lambda a.t \mid \texttt{letrec}\, a\, a = t\ \texttt{in}\ t$$

ASTs:

$$\Lambda \triangleq \mu S.(\mathbb{V} + (S \times S) + (\mathbb{V} \times S) + (\mathbb{V} \times \mathbb{V} \times S \times S))$$

where $\mathbb{V}$ is some fixed, countably infinite set (of names a of variables).

α-Structural recursion for Λ/α

Given a nominal set X

and functions $\begin{cases} f_{\mathrm{V}} : \mathbb{V} \to X \\ f_{\mathrm{A}} : X \times X \to X \\ f_{\mathrm{L}} : \mathbb{V} \times X \to X \\ f_{\mathrm{F}} : \mathbb{V} \times \mathbb{V} \times X \times X \to X, \end{cases}$

all supported by a finite subset $A \subseteq \mathbb{V}$,

there is a unique function $\hat{f} : \Lambda/\alpha \to X$ (supported by A as well) such that...

α-Structural recursion for Λ/α

... $\exists!$ function $\hat{f} : \Lambda/\alpha \to X$ such that:

$$
\boxed{
\begin{array}{rcl}
\hat{f}\, a_1 & = & f_{\mathrm{V}}\, a_1 \\
\hat{f}(e_1\, e_2) & = & f_{\mathrm{A}}(\hat{f}\, e_1, \hat{f}\, e_2) \\
a_1 \notin A \;\Rightarrow\; \hat{f}(\lambda a_1.e_1) & = & f_{\mathrm{L}}(a_1, \hat{f}\, e_1) \\
\multicolumn{3}{l}{a_1, a_2 \notin A \;\&\; a_1 \neq a_2 \;\&\; a_2 \notin fv(e_2) \;\Rightarrow} \\
\hat{f}(\mathtt{letrec}\, a_1\, a_2 = e_1\; \mathtt{in}\; e_2) & = & f_{\mathrm{F}}(a_1, a_2, \hat{f}\, e_1, \hat{f}\, e_2)
\end{array}
}
$$

provided freshness condition for binders (FCB) holds

for f_L: $(\exists a_1 \notin A)(\forall x \in X)\; a_1 \# f_L(a_1, x)$

for f_F: $(\exists a_1, a_2 \notin A)\; a_1 \neq a_2 \;\&$

$(\forall x_1, x_2 \in X)\; a_2 \# x_1 \;\Rightarrow$

$a_1, a_2 \# f_F(a_1, a_2, x_1, x_2)$

Example: capture-avoiding substitution

$\boxed{(a := e)(-) : \Lambda/{=_\alpha} \to \Lambda/{=_\alpha}}$ is $\hat{f}$ for:

- $f_V(a_1) \triangleq$ (if $a_1 = a$ then e else a_1)
- $f_A(e_1, e_2) \triangleq e_1\, e_2$
- $f_L(a_1, e_1) \triangleq \lambda a_1.\, e_1$
- $f_F(a_1, a_2, e_1, e_2) \triangleq \mathit{letrec}\, a_1\, a_2 = e_1\, \mathit{in}\, e_2$

These functions are all supported by

$A \triangleq \{a\} \cup \mathit{supp}(e)$

and the (FCB) holds because

$a_1 \mathrel{\#} \lambda a_1.\, e_1 = f_L(a_1, e_1)$,

$a_1, a_2 \mathrel{\#} \mathit{letrec}\, a_1\, a_2 = e_1\, \mathit{in}\, e_2 = f_F(a_1, a_2, e_1, e_2)$.

Pause

To be explained:

- Nominal sets, support and the freshness relation, $(-) \# (-)$.
- How is α-structural recursion proved?
- How to generalise α-structural recursion from the example language Λ to general languages with binders?
- What's involved with applying α-structural recursion in any particular case?
- Example: normalisation by evaluation.
- Machine-assisted support?

α-Structural recursion for Λ/α

Given a nominal set X

and functions $\begin{cases} f_{\mathrm{V}} : \mathbb{V} \to X \\ f_{\mathrm{A}} : X \times X \to X \\ f_{\mathrm{L}} : \mathbb{V} \times X \to X \\ f_{\mathrm{F}} : \mathbb{V} \times \mathbb{V} \times X \times X \to X, \end{cases}$

all supported by a finite subset $A \subseteq \mathbb{V}$,

there is a unique function $\hat{f} : \Lambda/\alpha \to X$ such that...

α-Structural recursion for Λ/α

... $\exists!$ function $\hat{f} : \Lambda/\alpha \to X$ such that:

$$
\boxed{
\begin{array}{rcl}
\hat{f}\,a_1 & = & f_{\mathrm{V}}\,a_1 \\
\hat{f}(e_1\,e_2) & = & f_{\mathrm{A}}(\hat{f}\,e_1, \hat{f}\,e_2) \\
a_1 \notin A \;\Rightarrow\; \hat{f}(\lambda a_1.e_1) & = & f_{\mathrm{L}}(a_1, \hat{f}\,e_1) \\
\multicolumn{3}{l}{a_1, a_2 \notin A \;\&\; a_1 \neq a_2 \;\&\; a_2 \notin fv(e_2) \;\Rightarrow} \\
\hat{f}(\mathtt{letrec}\,a_1\,a_2 = e_1\;\mathtt{in}\;e_2) & = & f_{\mathrm{F}}(a_1, a_2, \hat{f}\,e_1, \hat{f}\,e_2)
\end{array}
}
$$

provided freshness condition for binders (FCB) holds

for f_L: $(\exists a_1 \notin A)(\forall x \in X)\ a_1 \quad f_L(a_1, x)$

for f_F: $(\exists a_1, a_2 \notin A)\ a_1 \neq a_2\ \&$

$$(\forall x_1, x_2 \in X)\ a_2 \mathbin{\#} x_1 \;\Rightarrow$$

$$a_1, a_2 \mathbin{\#} f_F(a_1, a_2, x_1, x_2)$$

Proof—overview

α-Structural recursion reduces to ordinary structural recursion for ASTs within higher-order logic: roughly speaking, one makes a definition for all permutations simultaneously, i.e. uses $Perm \to X$ where you might expect to use a set X.

Proof—overview

α-Structural recursion reduces to ordinary structural recursion for ASTs within higher-order logic: roughly speaking, one makes a definition for all permutations simultaneously, i.e. uses $\mathit{Perm} \to X$ where you might expect to use a set X.

A key ingredient of the proof is:

<u>Freshness theorem</u> [LN p 19]

Given a nominal set X and $h \in \mathbb{V} \to_{\mathrm{fs}} X$ satisfying

$$(\exists a \in \mathbb{V})\ a \# h \ \ \&\ \ a \# h(a)$$

then $\exists!$ element $\mathit{fresh}(h) \in X$ satisfying

$$(\forall a \in \mathbb{V})\ a \# h \ \Rightarrow\ h(a) = \mathit{fresh}(h)$$

Proof—overview

α-Structural recursion reduces to ordinary structural recursion for ASTs within higher-order logic: roughly speaking, one makes a definition for all permutations simultaneously, i.e. uses $Perm \rightarrow X$ where you might expect to use a set X.

A key ingredient of the proof is:

<u>Freshness theorem</u> [LN p 19]

which in turn follows from the

"Some/Any" property of fresh atoms [LN p 18]

Proof—sketch

Define $\hat{g} : \Lambda \to (Perm \to X)$ by ordinary structural recursion:

- $\hat{g}\, a_1 \triangleq \lambda\pi \in Perm.\, f_V(\pi(a_1))$
- $\hat{g}(t_1\, t_2) \triangleq \lambda\pi \in Perm.\, f_A(\hat{g}\, t_1\, \pi, \hat{g}\, t_2\, \pi)$
- $\hat{g}(\lambda a_1.\, t_1) \triangleq fresh(\lambda a_1' \in \mathbb{V}.$
  $\lambda\pi \in Perm.\, f_L(a_1', \hat{g}\, t_1(\pi \circ (a_1\, a_1'))))$

The (FCB) for f_F ensures that the conditions of the Freshness Theorem are met.

Proof—sketch

Define $\hat{g} : \boldsymbol{\Lambda} \to (\boldsymbol{Perm} \to \boldsymbol{X})$ by ordinary structural recursion:

- $\hat{g}\, a_1 \triangleq \boldsymbol{\lambda}\pi \in \boldsymbol{Perm}.\, f_V(\pi(a_1))$
- $\hat{g}(t_1\, t_2) \triangleq \boldsymbol{\lambda}\pi \in \boldsymbol{Perm}.\, f_A(\hat{g}\, t_1\, \pi, \hat{g}\, t_2\, \pi)$
- $\hat{g}(\boldsymbol{\lambda} a_1.\, t_1) \triangleq \mathit{fresh}(\boldsymbol{\lambda} a_1' \in \mathbb{V}.$
  $\boldsymbol{\lambda}\pi \in \boldsymbol{Perm}.\, f_L(a_1', \hat{g}\, t_1(\pi \circ (a_1\, a_1'))))$
- $\hat{g}(\texttt{letrec}\, a_1\, a_2 = t_1 \;\texttt{in}\; t_2) \triangleq \cdots$(exercise)$\cdots$

Proof—sketch

Define $\hat{g} : \Lambda \rightarrow (Perm \rightarrow X)$ by ordinary structural recursion:

- $\hat{g}\, a_1 \triangleq \lambda\pi \in Perm.\, f_V(\pi(a_1))$
- $\hat{g}(t_1\, t_2) \triangleq \lambda\pi \in Perm.\, f_A(\hat{g}\, t_1\, \pi, \hat{g}\, t_2\, \pi)$
- $\hat{g}(\lambda a_1.\, t_1) \triangleq fresh(\lambda a_1' \in \mathbb{V}.$
  $\lambda\pi \in Perm.\, f_L(a_1', \hat{g}\, t_1(\pi \circ (a_1\, a_1'))))$
- $\hat{g}(\mathtt{letrec}\, a_1\, a_2 = t_1 \;\mathtt{in}\; t_2) \triangleq \cdots$(exercise)$\cdots$

Can prove (by rule induction for $=_\alpha$) that $t_1 =_\alpha t_2 \;\Rightarrow\; \hat{g}\, t_1 = \hat{g}\, t_2$.

Then $\hat{f}[t]_\alpha \triangleq \hat{g}\, t\, \iota$ well-defines the function $\hat{f} : \Lambda/\alpha \rightarrow X$ we seek.

Proof—sketch

Define $\hat{g} : \Lambda \to (Perm \to X)$ by ordinary structural recursion:

- $\hat{g}\, a_1 \triangleq \lambda\pi \in Perm.\, f_V(\pi(a_1))$
- $\hat{g}$
- $\hat{g}$ $)))$
- $\hat{g}$

Ca

t_1 =

E.g. if $a_1 \notin A$, then

$$\begin{aligned} \hat{f}[\lambda a_1.\, t_1]_\alpha &= \hat{g}(\lambda a_1.\, t_1)\,\iota \\ &= f_L(a_1, \hat{g}\, t_1(\iota \circ (a_1\, a_1))) \\ &= f_L(a_1, \hat{g}\, t_1\, \iota) \\ &= f_L(a_1, \hat{f}[t_1]_\alpha) \end{aligned}$$

Then $\hat{f}[t]_\alpha \triangleq \hat{g}\, t\, \iota$ well-defines the function $\hat{f} : \Lambda/\alpha \to X$ we seek.

Proof—overview

α-Structural recursion reduces to ordinary structural recursion for ASTs within higher-order logic: roughly speaking, one makes a definition for all permutations simultaneously, i.e. uses $Perm \longrightarrow X$ where you might expect to use a set X.

A key ingredient of the proof is:

Freshness theorem [LN p 19]

which in turn follows from the

"Some/Any" property of fresh atoms [LN p 18]

“Some/any” proof pattern

$$\boxed{\dfrac{t\{a''/a\} =_\alpha t'\{a''/a'\} \qquad a'' \# (a, t, a', t')}{\lambda a.\, t =_\alpha \lambda a'.\, t'}}$$

top-down proof	bottom-up proof
$(\exists a'' \in \mathbb{V})$	$(\forall a'' \in \mathbb{V})$
$\left(\begin{array}{c} a'' \# (a, t, a', t')\ \& \\ t\{a''/a\} =_\alpha t'\{a''/a'\} \end{array} \right)$	$\left(\begin{array}{c} a'' \# (a, t, a', t') \Rightarrow \\ t\{a''/a\} =_\alpha t'\{a''/a'\} \end{array} \right)$
$\Downarrow$	$\Uparrow$
$\lambda a.\, t =_\alpha \lambda a'.\, t'$	$\lambda a.\, t =_\alpha \lambda .a'\, t'$

“Some/Any” theorem [LN p 18]

If $S \in P_{\mathrm{fs}}(\mathbb{V})$,
then

$$(\forall a \in \mathbb{V})\ a \mathbin{\#} S \ \Rightarrow\ a \in S$$

iff

$$(\exists a \in \mathbb{V})\ a \mathbin{\#} S \ \&\ a \in S$$

“Some/Any” theorem [LN p 18]

If $S \in P_{\mathrm{fs}}(\mathbb{V})$ is supported by the finite subset $A \subseteq \mathbb{V}$, then

$$(\forall a \in \mathbb{V})\ a \notin A \ \Rightarrow \ a \in S$$

iff

$$(\exists a \in \mathbb{V})\ a \notin A \ \& \ a \in S$$

"Some/Any" theorem [LN p 18]

If $S \in P_{\mathrm{fs}}(\mathbb{V})$,
then

$$(\forall a \in \mathbb{V})\ a \# S \;\Rightarrow\; a \in S$$

iff

$$(\exists a \in \mathbb{V})\ a \# S \;\&\; a \in S$$

Proof If $a \# S$ and $a \in S$, then for any other a' with $a' \# S$ we have:

$$a' = (a\,a') \cdot a \in (a\,a') \cdot S = S$$

because $a, a' \# S$

$\square$

Freshness theorem [LN p 19]

Given a nominal set X and $h \in \mathbb{V} \to_{\mathrm{fs}} X$ satisfying

$$(\exists a \in \mathbb{V})\ a \# h \ \ \&\ \ a \# h(a) \qquad (*)$$

then $\exists!$ element $fresh(h) \in X$ satisfying

$$(\forall a \in \mathbb{V})\ a \# h \ \Rightarrow\ h(a) = fresh(h)$$

Freshness theorem [LN p 19]

Given a nominal set X and $h \in \mathbb{V} \to_{\mathrm{fs}} X$ satisfying

$$(\exists a \in \mathbb{V})\ a \mathrel{\#} h \ \ \&\ \ a \mathrel{\#} h(a) \qquad (*)$$

then $\exists!$ element $fresh(h) \in X$ satisfying

$$(\forall a \in \mathbb{V})\ a \mathrel{\#} h \ \Rightarrow\ h(a) = fresh(h)$$

Proof Suffices to show that h is constant on the non-empty set $\mathbb{V} - supp(h)$.

Freshness theorem [LN p 19]

Given a nominal set X and $h \in \mathbb{V} \to_{\mathrm{fs}} X$ satisfying

$$(\exists a \in \mathbb{V})\ a \mathbin{\#} h \ \ \&\ \ a \mathbin{\#} h(a) \qquad (*)$$

then $\exists!$ element $fresh(h) \in X$ satisfying

$$(\forall a \in \mathbb{V})\ a \mathbin{\#} h \ \Rightarrow\ h(a) = fresh(h)$$

Proof Suffices to show that h is constant on the non-empty set $\mathbb{V} - supp(h)$.
By Some/Any theorem can replace $(*)$ with $(\dagger)$.

Freshness theorem [LN p 19]

Given a nominal set X and $h \in \mathbb{V} \to_{\mathrm{fs}} X$ satisfying

$$(\forall a \in \mathbb{V})\ a \# h \ \Rightarrow\ a \# h(a) \qquad (\dagger)$$

then $\exists!$ element $fresh(h) \in X$ satisfying

$$(\forall a \in \mathbb{V})\ a \# h \ \Rightarrow\ h(a) = fresh(h)$$

Proof Suffices to show that h is constant on the non-empty set $\mathbb{V} - supp(h)$.
By Some/Any theorem can replace $(*)$ with $(\dagger)$.

Freshness theorem [LN p 19]

Given a nominal set X and $h \in \mathbb{V} \rightarrow_{\mathrm{fs}} X$ satisfying

$$(\forall a \in \mathbb{V})\ a \# h \ \Rightarrow\ a \# h(a) \qquad (\dagger)$$

then $\exists!$ element $\mathit{fresh}(h) \in X$ satisfying

$$(\forall a \in \mathbb{V})\ a \# h \ \Rightarrow\ h(a) = \mathit{fresh}(h)$$

Proof Suffices to show that h is constant on the non-empty set $\mathbb{V} - \mathit{supp}(h)$.

So for any $a \neq a'$ with $a, a' \# h$, we have $a' \# h(a')$ by $(\dagger)$ and $a \# h(a')$ because $a \# (h, a')$.

Hence

$$\begin{aligned} h(a') = (a\, a') \cdot h(a') &= ((a\, a') \cdot h)((a\, a') \cdot a') \\ &= h(a). \end{aligned}$$

$\square$

Pause

To be explained:

- Nominal sets, support and the freshness relation, $(-) \# (-)$.
- How is α-structural recursion proved?
- How to generalise α-structural recursion from the example language Λ to general languages with binders?
- What's involved with applying α-structural recursion in any particular case?
- Example: normalisation by evaluation.
- Machine-assisted support?

To be explained:

- Nominal sets, support and the freshness relation, $(-) \# (-)$.
- How is α-structural recursion proved?
- How to generalise α-structural recursion from the example language Λ to general languages with binders? Nominal signatures
- What's involved with applying α-structural recursion in any particular case?
- Example: normalisation by evaluation.
- Machine-assisted support?

α-Structural recursion for Λ/α

... $\exists!$ function $\hat{f} : \Lambda/\alpha \to S$ such that:

$$
\boxed{
\begin{array}{rcl}
\hat{f}\, x_1 & = & f_{\mathrm{V}}\, x_1 \\
\hat{f}(e_1\, e_2) & = & f_{\mathrm{A}}(\hat{f}\, e_1, \hat{f}\, e_2) \\
x_1 \notin A \;\Rightarrow\; \hat{f}(\lambda x_1 . e_1) & = & f_{\mathrm{L}}(x_1, \hat{f}\, e_1) \\
\multicolumn{3}{l}{x_1, x_2 \notin A \;\&\; x_1 \neq x_2 \;\&\; x_2 \notin fv(e_2) \;\Rightarrow} \\
\hat{f}(\mathtt{letrec}\, x_1\, x_2 = e_1\; \mathtt{in}\; e_2) & = & f_{\mathrm{F}}(x_1, x_2, \hat{f}\, e_1, \hat{f}\, e_2)
\end{array}
}
$$

provided freshness condition for binders (FCB) holds

for f_L: $(\exists x_1 \notin A)(\forall s \in S)\; x_1 \# f_L(x_1, s)$

for f_F: $(\exists x_1, x_2 \notin A)\; x_1 \neq x_2 \;\&$

$$(\forall s_1, s_2 \in S)\; x_2 \# s_1 \;\Rightarrow$$

$$x_1, x_2 \# f_F(x_1, x_2, s_1, s_2)$$

α-Structural recursion for Λ/α

...∃! function $\hat{f} : \Lambda/$

Using nominal signatures, these conditions can be determined automatically from the pattern of bindings in a constructor's arity...

$\hat{f}($

$x_1 \notin A \Rightarrow \hat{f}(\lambda$

$x_1, x_2 \notin A \;\&\; x_1 \neq x_2 \;\&\; x_2 \notin fv(e_2) \;\Rightarrow$

$\hat{f}(\mathtt{letrec}\, x_1\, x_2 = e_1 \;\mathtt{in}\; e_2) \;=\; f_{\mathrm{F}}(x_1, x_2, \hat{f}\, e_1, \hat{f}\, e_2)$

provided freshness condition for binders (FCB) holds

for f_L: $(\exists x_1 \notin A)(\forall s \in S)\; x_1 \,\#\, f_L(x_1, s)$

for f_F: $(\exists x_1, x_2 \notin A)\; x_1 \neq x_2 \;\&$

$(\forall s_1, s_2 \in S)\; x_2 \,\#\, s_1 \;\Rightarrow$

$x_1, x_2 \,\#\, f_F(x_1, x_2, s_1, s_2)$

Nominal signatures

Generalisation of many-sorted, algebraic signatures that includes info about how constructors bind names.

Not as general as some schemes for expressing binding patterns (cf. Pottier's Cαml), but a good compromise between expressiveness and simplicity.

Nominal signatures

are specified by:

- a set of atom-sorts **as** and a set of data-sorts **ds**.
- a set of constructors $K : \sigma \longrightarrow \mathbf{ds}$ whose arities σ are given by

σ	$::=$	**as**	atom-sort
	$\mid$	**ds**	data-sort
	$\mid$	$\mathbf{1}$	unit arity
	$\mid$	$\sigma * \sigma$	pair arity
	$\mid$	$\langle\!\langle \mathbf{as} \rangle\!\rangle \sigma$	atom-binding arity

Nominal signatures

are specified by:

- a set of **atom-sorts** as and a set of **data-sorts** ds.
- a set of **constructors** $K : \sigma \longrightarrow \mathsf{ds}$

E.g. nominal signature for
$\Lambda = \{t ::= x \mid t\,t \mid \lambda x.t \mid \mathtt{letrec}\, x\, x = t\ \mathtt{in}\ t\}$ has
atom-sort var, data-sort term and constructors:

$$V : \mathsf{var} \longrightarrow \mathsf{term}$$

$$A : \mathsf{term} * \mathsf{term} \longrightarrow \mathsf{term}$$

$$L : \langle\!\langle \mathsf{var} \rangle\!\rangle \mathsf{term} \longrightarrow \mathsf{term}$$

$$F : \langle\!\langle \mathsf{var} \rangle\!\rangle ((\langle\!\langle \mathsf{var} \rangle\!\rangle \mathsf{term}) * \mathsf{term}) \longrightarrow \mathsf{term}$$

A nominal signature for the π-calculus [LN p 9]

$$
\begin{aligned}
P &::= P|P \mid \nu(c)P \mid !P \mid S \\
S &::= 0 \mid S + S \mid G \\
G &::= c\,c.\,P \mid c(c).\,P \mid \tau.\,P \mid [c = c]G
\end{aligned}
$$

A nominal signature for the π-calculus [LN p 9]

atom-sorts	data-sorts	constructors
chan	**proc**	$Gsum$: **gsum** $\longrightarrow$ **proc**
	gsum	Par : **proc** $*$ **proc** $\longrightarrow$ **proc**
	pre	Res : «**chan**»**proc** $\longrightarrow$ **proc**
		Rep : **proc** $\longrightarrow$ **proc**
		$Zero$: **1** $\longrightarrow$ **gsum**
		Pre : **pre** $\longrightarrow$ **gsum**
		$Plus$: **gsum** $*$ **gsum** $\longrightarrow$ **gsum**
		Out : (**chan** $*$ **chan**) $*$ **proc** $\longrightarrow$ **pre**
		In : **chan** $*$ «**chan**»**proc** $\longrightarrow$ **pre**
		Tau : **proc** $\longrightarrow$ **pre**
		$Match$: (**chan** $*$ **chan**) $*$ **pre** $\longrightarrow$ **pre**

A nominal signature for polymorphic λ-calculus

$$\begin{aligned} \tau &::= \alpha \mid \tau \to \tau \mid \forall \alpha .\, \tau \\ t &::= x \mid t\, t \mid \lambda x : \tau .\, t \mid \Lambda \alpha .\, t \mid t\, \tau \end{aligned}$$

A nominal signature for polymorphic λ-calculus

atom-sorts	data-sorts	constructors
tyvar	**type**	$\mathit{Tyvar} : \mathbf{tyvar} \longrightarrow \mathbf{type}$
var	**term**	$\mathit{Fun} : \mathbf{type} * \mathbf{type} \longrightarrow \mathbf{type}$
		$\mathit{All} : \langle\!\langle \mathbf{tyvar} \rangle\!\rangle \mathbf{type} \longrightarrow \mathbf{type}$
		$\mathit{Var} : \mathbf{var} \longrightarrow \mathbf{term}$
		$\mathit{App} : \mathbf{term} * \mathbf{term} \longrightarrow \mathbf{term}$
		$\mathit{Lam} : \mathbf{type} * \langle\!\langle \mathbf{var} \rangle\!\rangle \mathbf{term} \longrightarrow \mathbf{term}$
		$\mathit{Gen} : \langle\!\langle \mathbf{tyvar} \rangle\!\rangle \mathbf{term} \longrightarrow \mathbf{term}$
		$\mathit{Spec} : \mathbf{term} * \mathbf{type} \longrightarrow \mathbf{term}$

Nominal terms

Nominal terms (t) and their arities ($t : \sigma$) over a nominal signature Σ:

- $\boxed{a : \mathsf{as}}$ if $a \in \mathbb{A}$ and $sort(a) = \mathsf{as}$
- $\boxed{K\,t : \mathsf{ds}}$ if $K : \sigma \longrightarrow \mathsf{ds}$ and $t : \sigma$
- $\boxed{\langle\rangle : \mathbf{1}}$
- $\boxed{\langle t_1, t_2\rangle : \sigma_1 * \sigma_2}$ if $t_1 : \sigma_1$ & $t_2 : \sigma_2$
- $\boxed{\text{«}a\text{»}t : \text{«}\mathsf{as}\text{»}\sigma}$ if $a : \mathsf{as}$ & $t : \sigma$

Nominal terms

$Perm$-action on nominal terms over Σ:

- $\boxed{a : \mathsf{as}}$ $\pi \cdot a = \pi(a)$
- $\boxed{K\,t : \mathsf{ds}}$ $\pi \cdot (K\,t) = K(\pi \cdot t)$
- $\boxed{\langle\rangle : \mathbf{1}}$ $\pi \cdot \langle\rangle = \langle\rangle$
- $\boxed{\langle t_1, t_2\rangle : \sigma_1 * \sigma_2}$ $\pi \cdot \langle t_1, t_2\rangle = \langle \pi \cdot t_1, \pi \cdot t_2\rangle$
- $\boxed{«a»t : «\mathsf{as}»\sigma}$ $\pi \cdot «a»t = «\pi(a)»(\pi \cdot t)$

For this $Perm$-action we get
$supp(t)$ = finite set of <u>all</u> atoms occurring in t

Nominal terms

$Perm$-action on nominal terms over Σ:

- $\boxed{a : \mathsf{as}}$ $\pi \cdot a = \pi(a)$
- $\boxed{K\, t : \mathsf{ds}}$ $\pi \cdot (K\, t) = K(\pi \cdot t)$
- $\boxed{\langle\rangle : 1}$ $\pi \cdot \langle\rangle = \langle\rangle$
- $\boxed{\langle t_1, t_2\rangle : \sigma_1 * \sigma_2}$ $\pi \cdot \langle t_1, t_2\rangle = \langle \pi \cdot t_1, \pi \cdot t_2\rangle$
- $\boxed{«a»t : «\mathsf{as}»\sigma}$ $\pi \cdot «a»t = «\pi(a)»(\pi \cdot t)$

$\mathsf{T}(\Sigma)_\sigma \triangleq$ nominal set of terms of arity σ over nominal signature Σ

α-Equivalence of nominal terms

$$\frac{a : \mathsf{as}}{a =_\alpha a : \mathsf{as}} \qquad \frac{K : \sigma \longrightarrow \mathsf{ds} \qquad t =_\alpha t' : \sigma}{K\,t =_\alpha K\,t' : \mathsf{ds}}$$

$$\frac{}{\langle\rangle =_\alpha \langle\rangle : \mathbf{1}} \qquad \frac{t_1 =_\alpha t_1' : \sigma_1 \qquad t_2 =_\alpha t_2' : \sigma_2}{\langle t_1, t_2\rangle =_\alpha \langle t_1', t_2'\rangle : \sigma_1 * \sigma_2}$$

$$\frac{\begin{array}{c} a, a', a'' : \mathsf{as} \qquad a'' \mathrel{\#} (a, t, a', t') \\ (a\,a'') \cdot t =_\alpha (a'\,a'') \cdot t' : \sigma \end{array}}{«a»t =_\alpha «a'»t' : «\mathsf{as}»\sigma}$$

α-Equivalence of nominal terms

$$\frac{a : \mathsf{as}}{a =_\alpha a : \mathsf{as}} \qquad \frac{K : \sigma \longrightarrow \mathsf{ds} \qquad t =_\alpha t' : \sigma}{K\, t =_\alpha K\, t' : \mathsf{ds}}$$

$$\frac{}{\langle\rangle =_\alpha \langle\rangle : \mathbf{1}} \qquad \frac{t_1 =_\alpha t_1' : \sigma_1 \qquad t_2 =_\alpha t_2' : \sigma_2}{\langle t_1, t_2\rangle =_\alpha \langle t_1', t_2'\rangle : \sigma_1 * \sigma_2}$$

$$\frac{\begin{array}{c} a, a', a'' : \mathsf{as} \qquad a'' \# (a, t, a', t') \\ (a\, a'') \cdot t =_\alpha (a'\, a'') \cdot t' : \sigma \end{array}}{«a»t =_\alpha «a'»t' : «\mathsf{as}»\sigma}$$

Action on α-equivalence classes: $\boxed{\pi \cdot [t]_\alpha \triangleq [\pi \cdot t]_\alpha}$

For this $supp([t]_\alpha)$ is finite set of all <u>free</u> atoms of t.

α-Equivalence of nominal terms

$$\frac{a : \mathsf{as}}{a =_\alpha a : \mathsf{as}} \qquad \frac{K : \sigma \longrightarrow \mathsf{ds} \qquad t =_\alpha t' : \sigma}{K\,t =_\alpha K\,t' : \mathsf{ds}}$$

$$\frac{}{\langle\rangle =_\alpha \langle\rangle : \mathbf{1}} \qquad \frac{t_1 =_\alpha t_1' : \sigma_1 \qquad t_2 =_\alpha t_2' : \sigma_2}{\langle t_1, t_2\rangle =_\alpha \langle t_1', t_2'\rangle : \sigma_1 * \sigma_2}$$

$$\frac{\begin{array}{c} a, a', a'' : \mathsf{as} \qquad a'' \mathbin{\#} (a, t, a', t') \\ (a\,a'') \cdot t =_\alpha (a'\,a'') \cdot t' : \sigma \end{array}}{«a»t =_\alpha «a'»t' : «\mathsf{as}»\sigma}$$

$\boxed{\mathsf{T}_\alpha(\Sigma)_\sigma \triangleq}$ nominal set of α-equivalence classes of terms of arity σ over Σ

α-Structural recursion for a general nominal signature Σ

Two forms given in the paper:

- first, “arity-directed” version [Theorem 17, p 21]
- second, “sort-directed” version [Theorem 22, p 26]
  - harder to state & prove, but more useful
  - recursion for running example $\Lambda/{=_\alpha}$ is an instance.

Second α-structural recursion theorem

Input:

1. Nominal signature Σ.
2. Family of nominal sets X_{ds} indexed by the data-sorts ds of Σ.
3. Family of functions $f_K \in (X^{(\sigma)} \to_{\mathrm{fs}} X_{\mathsf{ds}})$ indexed by the constructors $K : \sigma \to \mathsf{ds}$ of Σ.

Second α-structural recursion theorem

Input:

1. Nominal signature Σ.
2. Family of nominal sets X_{ds} indexed by the data-sorts ds of Σ.
3. Family of functions $f_K \in (X^{(\sigma)} \rightarrow_{\mathrm{fs}} X_{\mathsf{ds}})$ indexed by the constructors $K : \sigma \rightarrow \mathsf{ds}$ of Σ.

$$
\begin{aligned}
X^{(\mathsf{as})} &\triangleq \mathbb{A}_{\mathsf{as}} \\
X^{(\mathsf{ds})} &\triangleq X_{\mathsf{ds}} \\
X^{(\langle\rangle)} &\triangleq 1 \\
X^{(\sigma_1 * \sigma_2)} &\triangleq X^{(\sigma_1)} \times X^{(\sigma_1)} \\
X^{(\langle\!\langle \mathsf{as} \rangle\!\rangle \sigma)} &\triangleq \mathbb{A}_{\mathsf{as}} \times X^{(\sigma)}
\end{aligned}
$$

Second α-structural recursion theorem

Input:

1. Nominal signature Σ.
2. Family of nominal sets $X_{\mathbf{ds}}$ indexed by the data-sorts $\mathbf{ds}$ of Σ.
3. Family of functions $f_K \in (X^{(\sigma)} \longrightarrow_{\mathrm{fs}} X_{\mathbf{ds}})$ indexed by the constructors $K : \sigma \rightarrow \mathbf{ds}$ of Σ.
4. A single finite set A of atoms that supports all the functions f_K
5. Proof that each f_K satisfies a FCB whose statement is determined by the arity σ of $K : \sigma \rightarrow \mathbf{ds}$.

Second α-structural recursion theorem

Input:

E.g. for $F : \langle\!\langle \mathbf{var} \rangle\!\rangle ((\langle\!\langle \mathbf{var} \rangle\!\rangle \mathbf{term}) * \mathbf{term}) \longrightarrow \mathbf{term}$,
(FCB) for f_F is: $(\exists a_1, a_2 \notin A)\ a_1 \neq a_2\ \&$
$(\forall x_1, x_2 \in X)\ a_2 \# x_1 \ \Rightarrow$
$a_1, a_2 \# f_F(a_1, ((a_2, x_1), x_2))$

4. A single finite set A of atoms that supports all the functions f_K
5. Proof that each f_K satisfies a FCB whose statement is determined by the arity σ of $K : \sigma \longrightarrow \mathbf{ds}$.

Second α-structural recursion theorem

Output:

family of functions $\hat{f}_{\mathbf{ds}} \in (\mathsf{T}_\alpha(\Sigma)_{\mathbf{ds}} \longrightarrow_{\mathrm{fs}} X_{\mathbf{ds}})$ indexed by the data-sorts **ds** of Σ

- uniquely determined by mutually recursive, conditional equations

$$\text{condition} \;\Rightarrow\; \hat{f}_{\mathbf{ds}}(K\,e) = f_K(\cdots \hat{f}_{(-)} \cdots)$$

one for each constructor $K : \sigma \to \mathbf{ds}$ of Σ

Second α-structural recursion theorem

Output:

family of functions $\hat{f}_{\mathbf{ds}} \in (\mathsf{T}_\alpha(\Sigma)_{\mathbf{ds}} \longrightarrow_{\mathrm{fs}} X_{\mathbf{ds}})$ indexed by the data-sorts **ds** of Σ

- uniquely determined by mutually recursive, conditional equations

  condition $\Rightarrow$ $\hat{f}_{\mathbf{ds}}(K\,e) = f_K(\cdots \hat{f}_{(-)} \cdots)$

  one for each constructor $K : \sigma \to \mathbf{ds}$ of Σ

determined by the arity
σ of $K : \sigma \to \mathbf{ds}$

Second α-structural recursion theorem

Output:

family of functions $\hat{f}_{\mathsf{ds}} \in (\mathsf{T}_\alpha(\Sigma)_{\mathsf{ds}} \rightarrow_{\mathrm{fs}} X_{\mathsf{ds}})$ indexed by the data-sorts **ds** of Σ

- uniquely determined by mutually recursive, conditional equations

$$\text{condition} \;\Rightarrow\; \hat{f}_{\mathsf{ds}}(K\, e) = f_K(\cdots \hat{f}_{(-)} \cdots)$$

  one for each constructor $K : \sigma \rightarrow \mathsf{ds}$ of Σ

- all supported by the given finite set of atoms A

To be explained:

- Nominal sets, support and the freshness relation, $(-) \# (-)$.
- How is α-structural recursion proved?
- How to generalise α-structural recursion from the example language Λ to general languages with binders?
- What's involved with applying α-structural recursion in any particular case?
- Example: normalisation by evaluation.
- Machine-assisted support?

Given an informal recursive definition on ASTs/α for a nominal signature Σ, to show that it is an instance of (second) α-structural recursion theorem:

1. identify which sets (X_{ds}) and functions (f_K) are involved;
2. give each X_{ds} a nominal-set structure and prove the f_K are all supported by a single finite set;
3. for each constructor K in Σ, verify the (FCB) for f_K.

Given an informal recursive definition on ASTs/α for a nominal signature Σ, to show that it is an instance of (second) α-structural recursion theorem:

1. identify which sets (X_{ds}) and functions (f_K) are involved;
2. give each X_{ds} a nominal-set structure and prove the f_K are all supported by a single finite set;
3. for each constructor K in Σ, verify the (FCB) for f_K.

For step 2 we can use:

<u>Fact</u> The standard set-theoretic model of HOL (without choice) restricts to finitely supported elements; e.g. if we apply a construction of HOL-ε to finitely supported functions we get another such.

Given an informal recursive definition on ASTs/α for a nominal signature Σ, to show that it is an instance of (second) α-structural recursion theorem:

1. identify which sets (X_{ds}) and functions (f_K) are involved;
2. give each X_{ds} a nominal-set structure and prove the f_K are all supported by a single finite set;
3. for each constructor K in Σ, verify the (FCB) for f_K.

Step 3 is sometimes trivial (e.g. capture-avoiding substitution), sometimes not (see next lecture).

End of lecture 2