Nominal Reasoning Techniques in Isabelle/HOL

Christian Urban

Thank you very much for the invitation!

Existing Techniques. . . ?

- Somehow our (my) intuition from pencil-and-paper proofs is not as good supported by them as one would hope.
- Nominal Approach - in some sense no original idea: we will just formalise what happens on paper:
  - we will work with alpha-equivalence classes (just in a named fashion),
  - we will have something like the variable convention,
  - we will choose fresh names,. . .

Existing Techniques. . . ?

- Somehow our (my) intuition from pencil-and-paper proofs is not as good s
- N o h
  - we will have something like the variable convention,
  - we will choose fresh names,. . .

Just "copying" what is on paper might not be a good idea (we are not improving anything; we are not getting substitution for free, for example) — on the other hand we are in a quite powerful framework (we have no problems with SN for example).

The Nominal Datatype Package

What we can do:

- automatically prove the weakening lemma:

  $$\Gamma \vdash t : \tau \Rightarrow \text{valid}\,\Gamma' \Rightarrow \Gamma \subseteq \Gamma' \Rightarrow \Gamma' \vdash t : \tau$$

- automatically prove the substitution lemma:

  $$x \neq y \Rightarrow x \# L \Rightarrow$$
  $$M[x:=N][y:=L] = M[y:=L][x:=N[y:=L]]$$

- formalise the usual SN-proof.

The Nominal Datatype Package

What we can do:

- a:

It is **not** true with the nominal datatype package that you can prove everything as long as it is about the lambda-calculus.

- automatically prove the substitution lemma:

$$x \neq y \Rightarrow x \# L \Rightarrow$$
$$M[x:=N][y:=L] = M[y:=L][x:=N[y:=L]]$$

- formalise the usual SN-proof.

Other Calculi

Lambda-mu
Var "var"
App "trm" "trm"
Lam "«var»trm"
Pss "mvar" "trm"
Act "«mvar»trm"

System F-sub
TVar "tvar"
Top
Fun "ty" "ty"
TAll "«tvar»ty" "ty"

Var "var"
Lam "«var»trm" "ty"
TAbs "«tvar»trm" "ty"
App "trm" "trm"
TApp "trm" "ty"

Term-Calc. from my PhD
Ax "name" "coname"
Imp_R "«name»«coname»trm" "coname"
Imp_L "«coname»trm" "«name»trm" "name"
Cut "«coname»trm" "«coname»trm"

Lam + LetRec (with enc.)
π-calculus

Other Calculi

Lambda-mu

Var "

App

Lam

Pss "

Act "

Term

Ax "

Imp

Imp

Cut "

System F-sub

, "ty"

m" "ty"

"

"

nc.)

We **cannot** do some of the fancy bindings from Philadelphia and Cambridge. Also, there are still a number of gaps in the infrastructure in the nominal datatype-package...some time needed before they can be closed. One intention of this course is that you can help yourself until the gaps are closed.

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

a countable infinite set — this will be important on later on.

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example lambda-calculus

$$\lambda a.\lambda b.(a\, b\, c)$$

a and b are atoms—bound and binding

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example lambda-calculus

$$\lambda a.\lambda b.(a\, b\, c)$$

c is an atom—bindable

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example lambda-calculus

$$\lambda c.\lambda a.\lambda b.(a\,b\,c)$$

now c is bound

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example integrals

$$\int_0^1 x^2 + y\, dx$$

x is an atom—bound and binding

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example integrals

$$\int_{-\infty}^{\infty} \left(\int_{0}^{1} x^2 + y \, dx \right) dy$$

y is an atom—bindable

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example integrals

$$\int_{0}^{1} x^{2} + y\,dx$$

$\mathbf{0}$, $\mathbf{1}$ and $\mathbf{2}$ are constants

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** and **bindable** is an atom (independent from the language at hand).

example integrals

$$\int_{-\infty}^{\infty} \left(\int_{0}^{1} x^{2} + y \, dx \right) d2$$

binding 2 does not make sense

We Start with Atoms

We introduce **atoms**. Everything that is **bound**, **binding** ... ndent from t...

examp...

Why atoms? Because an operation we introduce shortly will act on atoms **only** and leaves everything else alone.

$$\int_{-\infty}^{\infty}\left(\int_{0}^{1} x^{2} + y\, dx\right) d2$$

binding $\mathbf{2}$ does not make sense

Swappings

In general, renaming substitutions do not respect α-equivalence, e.g.

$$\lambda a.b \qquad\qquad \lambda c.b$$

Swappings

In general, renaming substitutions do not respect α-equivalence, e.g.

$$\begin{aligned}[b := a]\,\lambda a.b &= \lambda a.a\end{aligned} \qquad \begin{aligned}[b := a]\,\lambda c.b &= \lambda c.a\end{aligned}$$

Swappings

In general, renaming substitutions do not respect α-equivalence, e.g.

$$[b := a]\,\lambda a.b = \lambda a.a \qquad\qquad [b := a]\,\lambda c.b = \lambda c.a$$

<u>Traditional Solution:</u> replace $[b := a]t$ by a more complicated, 'capture-avoiding' form of substitution.

Swappings

In general, renaming substitutions do not respect α-equivalence, e.g.

$$(b\ a)\bullet \lambda a.b = \lambda b.a \qquad\qquad (b\ a)\bullet \lambda c.b = \lambda c.a$$

Nice Alternative: use a less complicated operation for renaming

$$(b\ a)\bullet t \stackrel{\text{def}}{=} \text{swap } \textbf{all} \text{ occurrences of } b \text{ and } a \text{ in } t$$

Swappings

In general, renaming substitutions do not respect α-equivalence, e.g.

$$(b\ a)\bullet \lambda a.b = \lambda b.a \qquad (b\ a)\bullet \lambda c.b = \lambda c.a$$

<u>Nice Alternative:</u> use a less complicated operation for renaming

$(b\ a)\bullet t \stackrel{\text{def}}{=}$ swap **all** occurrences of b and a in t

be they bound, binding or bindable

Swappings

In general, renaming substitutions do not respect α-equivalence, e.g.

$$(b\,a)\bullet\lambda a.b = \lambda b.a \qquad\qquad (b\,a)\bullet\lambda c.b = \lambda c.a$$

<u>Nice Alternative:</u> use a less complicated operation for renaming

$(b\,a)\bullet t \stackrel{\text{def}}{=}$ swap **all** occurrences of b and a in t

Unlike for $[b:=a](-)$, for $(b\,a)\bullet(-)$ we do have if $t =_\alpha t'$ then $(b\,a)\bullet t =_\alpha (b\,a)\bullet t'$.

Permutations

We shall extend 'swappings' to '(finite) lists of swappings'

$$(a_1\, b_1) \ldots (a_n\, b_n),$$

also called **permutations** (we shall often write π for them). Permutations are **bijective** mappings from atoms to atoms. For example

$$\pi = \begin{pmatrix} a \mapsto b \\ b \mapsto a \\ c \mapsto c \end{pmatrix} = (c\, b)(a\, b)(a\, c)$$

Permutations

We shall extend 'swappings' to '(finite) lists of swappings'

$$(a_1\, b_1) \ldots (a_n\, b_n),$$

also called **permutations** (we shall often write π for them). Permutations are **bijective** mappings from atoms to atoms. For example

$$\pi = \begin{pmatrix} a \mapsto b \\ b \mapsto a \\ c \mapsto c \end{pmatrix} \qquad (c\, b)(a\, b)(a\, c) \bullet a = b$$

Permutations

We shall extend 'swappings' to '(finite) lists of swappings'

$$(a_1\, b_1) \ldots (a_n\, b_n),$$

also called **permutations** (we shall often write π for them). Permutations are **bijective** mappings from atoms to atoms. For example

$$\pi = \begin{pmatrix} a \mapsto b \\ b \mapsto a \\ c \mapsto c \end{pmatrix} \qquad (c\, b)(a\, b)(a\, c) \bullet b = a$$

Permutations

We shall extend 'swappings' to '(finite) lists of swappings'

$$(a_1\, b_1) \ldots (a_n\, b_n),$$

also called **permutations** (we shall often write π for them). Permutations are **bijective** mappings from atoms to atoms. For example

$$\pi = \begin{pmatrix} a \mapsto b \\ b \mapsto a \\ c \mapsto c \end{pmatrix} \qquad (c\, b)(a\, b)(a\, c) \bullet c = c$$

Permutations

We shall extend 'swappings' to '(finite) lists of swappin

also cal en write
π for t **ve**
mapping xample

Our list-representation is **not** unique, because

$$(c\,b)(a\,b)(a\,c) \text{ and } (a\,b)$$

are the 'same' permutation.

$$\pi = \begin{pmatrix} a \mapsto b \\ b \mapsto a \\ c \mapsto c \end{pmatrix} \qquad (c\,b)(a\,b)(a\,c) \bullet c = c$$

Permutations on Atoms

A permutation **acts** on an atom as follows:

$$[]\bullet a \stackrel{\text{def}}{=} a$$

$$((a_1\, a_2) :: \pi)\bullet a \stackrel{\text{def}}{=} \begin{cases} a_1 & \text{if } \pi\bullet a = a_2 \\ a_2 & \text{if } \pi\bullet a = a_1 \\ \pi\bullet a & \text{otherwise} \end{cases}$$

- $[]$ stands for the empty list (the identity permutation), and
- $(a_1\, a_2) :: \pi$ stands for the permutation π followed by the swapping $(a_1\, a_2)$

Permutations on Atoms (ct.)

- the **composition** of two permutations is given by list-concatenation, written as $\pi'@\pi$,
- the **inverse** of a permutation is given by list reversal, written as π^{-1}, and
- the **disagreement set** of two permutations π and π' is the set of atoms

$$ds(\pi, \pi') \stackrel{\text{def}}{=} \{a \mid \pi \bullet a \neq \pi' \bullet a\}$$

- $\pi \sim \pi' \stackrel{\text{def}}{=} ds(\pi, \pi') = \varnothing$.

Permutations on Atoms (ct.)

- the **composition** of two permutations is given by list-concatenation, written as $\pi'@\pi$,
- the **inverse** of a permutation is given by list reversal, written as π^{-1}, and
- the **disagreement set** of two permutations π and π' is the set of atoms

$$\pi = \begin{pmatrix} a \mapsto b \\ b \mapsto c \\ c \mapsto a \end{pmatrix} \qquad \pi^{-1} = \begin{pmatrix} b \mapsto a \\ c \mapsto b \\ a \mapsto c \end{pmatrix}$$

$$= (a\,c)(a\,b) \qquad = (a\,b)(a\,c)$$

- π

Permutations on Atoms (ct.)

- the **composition** of two permutations is given by list-concatenation, written as $\pi' @ \pi$,
- the **inverse** of a permutation is given by list reversal, written as π^{-1}, and
- the **disagreement set** of two permutations π and π' is the set of atoms

$$ds(\pi, \pi') \stackrel{\text{def}}{=} \{a \mid \pi \bullet a \neq \pi' \bullet a\}$$

- $\pi \sim \pi' \stackrel{\text{def}}{=} ds(\pi, \pi') = \varnothing$.

Permutations on Atoms (ct.)

- the **composition** of two permutations is given by list-concatenation, written as $\pi' @ \pi$,
- the
  list
- the
  permutations π and π' is the set of atoms

> for (finite) permutations this set is always finite (namely a subset of the atoms occurring π and π')

$$ds(\pi, \pi') \stackrel{\text{def}}{=} \{a \mid \pi \bullet a \neq \pi' \bullet a\}$$

- $\pi \sim \pi' \stackrel{\text{def}}{=} ds(\pi, \pi') = \varnothing.$

Permutations on Atoms (ct.)

- the **composition** of two permutations is given by list concatenation, written as $\pi'@($
- the
  list
- the
  per… …ns

Example $ds((a\,c)(a\,b),(a\,b))$?

$$\begin{pmatrix} a \mapsto b \\ b \mapsto c \\ c \mapsto a \end{pmatrix} \quad \begin{pmatrix} a \mapsto b \\ b \mapsto a \\ c \mapsto c \end{pmatrix}$$

$$= \{b, c\}$$

$$ds(\pi,\pi') \stackrel{\text{def}}{=} \{a \mid \pi \bullet a \neq \pi' \bullet a\}$$

- $\pi \sim \pi' \stackrel{\text{def}}{=} ds(\pi,\pi') = \varnothing.$

Permutations on Atoms (ct.)

- the **composition** of two permutations is given by list-concatenation, written as $\pi'@\pi$,
- the **inverse** of a permutation is given by list reversal, written as π^{-1}, and
- the **disagreement set** of two permutations π and π' is the set of atoms

$$ds(\pi,\pi') \stackrel{\text{def}}{=} \{a \mid \pi \bullet a \neq \pi' \bullet a\}$$

- $\pi \sim \pi' \stackrel{\text{def}}{=} ds(\pi,\pi') = \varnothing$.

Properties of Permutations

Here a, b and c are arbitrary atoms:

- $(b\,b)\bullet a = a$, $(b\,c)\bullet a = (c\,b)\bullet a$
- $\pi^{-1}\bullet(\pi\bullet a) = a$
- $\pi\bullet a = b$ if and only if $a = \pi^{-1}\bullet b$
- $\pi_1 @ \pi_2\bullet a = \pi_1\bullet(\pi_2\bullet a)$
- $\pi\bullet((b\,c)\bullet a) = (\pi\bullet b\ \ \pi\bullet c)\bullet(\pi\bullet a)$

the first, second and last fact can be generalised to

- if $\pi \sim \pi'$ then $\pi\bullet a = \pi'\bullet a$

Properties of Permutations

He

Preview: in the future, permutations will be completely characterised by the properties:

- $[\,]\bullet x = x$
- $\pi_1 @ \pi_2 \bullet x = \pi_1 \bullet (\pi_2 \bullet x)$
- if $\pi \sim \pi'$ then $\pi \bullet x = \pi' \bullet x$

where x stands also for other 'things', not just atoms. Don't worry this will become clearer later on.

the

ger

- if $\pi \sim \pi'$ then $\pi \bullet a = \pi' \bullet a$

Properties of Permutations

Here a, b and c are arbitrary atoms:

- $(b\,b)\bullet a = a$, $(b\,c)\bullet a = (c\,b)\bullet a$
- $\pi^{-1}\bullet(\pi\bullet a) = a$
- $\pi\bullet a = b$ if and only if $a = \pi^{-1}\bullet b$
- $\pi_1 @ \pi_2\bullet a = \pi_1\bullet(\pi_2\bullet a)$
- $\pi\bullet((b\,c)\bullet a) = (\pi\bullet b\;\;\pi\bullet c)\bullet(\pi\bullet a)$

the first, second and last fact can be generalised to

- if $\pi \sim \pi'$ then $\pi\bullet a = \pi'\bullet a$

Permutations on λ-Terms

$$\pi \bullet (a) \quad \text{given by the action on atoms}$$

$$\pi \bullet (t_1\, t_2) \stackrel{\text{def}}{=} (\pi \bullet t_1)(\pi \bullet t_2)$$

$$\pi \bullet (\lambda a.t) \stackrel{\text{def}}{=} \lambda(\pi \bullet a).(\pi \bullet t)$$

We have:

- $\pi^{-1} \bullet (\pi \bullet t) = t$
- $t_1 = t_2$ if and only if $\pi \bullet t_1 = \pi \bullet t_2$
- $\pi \bullet t_1 = t_2$ if and only if $t_1 = \pi^{-1} \bullet t_2$

(The attentive listener might like to prove these properties. You never know what you are being told.)

Permutations on λ-Terms

$$\begin{array}{rcl} \pi\bullet(a) & & \text{given by the action on atoms} \\ \pi\bullet(t_1\,t_2) & \stackrel{\text{def}}{=} & (\pi\bullet t_1)(\pi\bullet t_2) \\ \pi\bullet(\lambda a.t) & \stackrel{\text{def}}{=} & \lambda(\pi\bullet a).(\pi\bullet t) \end{array}$$

'we treat lambdas as if there were no binders'

We have:

- $\pi^{-1}\bullet(\pi\bullet$
- $t_1 = t_2$ if and only if $\pi\bullet t_1 = \pi\bullet t_2$
- $\pi\bullet t_1 = t_2$ if and only if $t_1 = \pi^{-1}\bullet t_2$

(The attentive listener might like to prove these properties. You never know what you are being told.)

Permutations on λ-Terms

$$\pi\bullet(a) \qquad \text{given by the action on atoms}$$

$$\pi\bullet(t_1\, t_2) \stackrel{\text{def}}{=} (\pi\bullet t_1)(\pi\bullet t_2)$$

$$\pi\bullet(\lambda a.t) \stackrel{\text{def}}{=} \lambda(\pi\bullet a).(\pi\bullet t)$$

We have:

- $\pi^{-1}\bullet(\pi\bullet t) = t$
- $t_1 = t_2$ if and only if $\pi\bullet t_1 = \pi\bullet t_2$
- $\pi\bullet t_1 = t_2$ if and only if $t_1 = \pi^{-1}\bullet t_2$

(The attentive listener might like to prove these properties. You never know what you are being told.)

Permutations on λ-Terms

What is it about permutations? Well...

- they have much nicer properties than renaming-substitutions (stemming from the fact that they are bijections on atoms),
- they give rise to a very simple definition of α-equivalence (shown next)
- and more later on

W

(The attentive listener might like to prove these properties. You never know what you are being told.)

α-Equivalence

Consider the following four rules:

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

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

assuming $a \neq b$

α-Equivalence

Consider the following four rules:

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

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

assuming $a \neq b$

$\lambda a.t \approx \lambda b.s$ iff t is α-equivalent with s in which all occurrences of b have been renamed to a...oops **permuted** to a.

α-Equivalence

But this alone leads to an 'unsound' rule! Consider

$$\lambda a.b \quad \text{and} \quad \lambda b.a$$

which are **not** α-equivalent. However, if we apply the permutation $(a\,b)$ to a we get

$$b \approx b$$

which leads to non-sense.

We need to ensure that there are **no** 'free' occurrences of a in s. This is achieved by freshness, written $a \# s$.

α-Equivalence

Consider the following four rules:

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

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

assuming $a \neq b$

$\lambda a.t \approx \lambda b.s$ iff t is α-equivalent with s in which all occurrences of b have been renamed to a... oops **permuted** to a.

Freshness

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

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

assuming $a \neq b$

Be careful, we have defined two relations over lambda-terms/syntax-trees. We have **not** defined what 'bound' or 'free' means. That is a feature, not a bug.™

$\approx$ is an Equivalence

You might be an agnostic and notice that

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

is defined rather asymmetrically. Still we have:

Theorem: $\approx$ is an equivalence relation.

(Reflexivity) $t \approx t$

(Symmetry) if $t_1 \approx t_2$ then $t_2 \approx t_1$

(Transitivity) if $t_1 \approx t_2$ and $t_2 \approx t_3$ then $t_1 \approx t_3$

$\Rightarrow$ is rather tricky to prove

Comparison with $=_\alpha$

Traditionally $=_\alpha$ is defined as

> least congruence which identifies $\lambda a.t$ with $\lambda b.[a := b]t$ provided b is not free in t

where $[a := b]t$ replaces all free occurrences of a by b in t.

- with $\approx$ and $\#$ we never need to choose a 'fresh' atom (good for implementations)
- permutation respects both relations, whilst renaming-substitution does not

General Permutations

So far we have only considered permutations acting on atoms and lambda-terms. We are now going to overload $\_\bullet\_ : \alpha\ prm \Rightarrow \iota \Rightarrow \iota$ to act on other types as well.

- $\pi \bullet a$ $\quad$ a being an atom (of type α)

$$[] \bullet a \stackrel{\text{def}}{=} a$$

$$((a_1\, a_2) :: \pi) \bullet a \stackrel{\text{def}}{=} \begin{cases} a_1 & \text{if } \pi \bullet a = a_2 \\ a_2 & \text{if } \pi \bullet a = a_1 \\ \pi \bullet a & \text{otherwise} \end{cases}$$

General Permutations

So far we have only considered permutations acting on atoms and lambda-terms. We are now going to overload $\_\bullet\_ : \alpha\ prm \Rightarrow \iota \Rightarrow \iota$ to act on other types as well.

For sake of simplicity, let us assume we only have one type of atoms.

- π …)

$$((a_1\,a_2) :: \pi)\bullet a \stackrel{\text{def}}{=} \begin{cases} a_1 & \text{if } \pi\bullet a = a_2 \\ a_2 & \text{if } \pi\bullet a = a_1 \\ \pi\bullet a & \text{otherwise} \end{cases}$$

General Permutations

So far we have only considered permutations acting on atoms and lambda-terms. We are now going to overload $\_\bullet\_ : \alpha\ prm \Rightarrow \iota \Rightarrow \iota$ to act on other types as well.

- $\pi \bullet a$ $\quad a$ being an atom (of type α)

$$
\begin{array}{rcl}
[]\bullet a & \stackrel{\text{def}}{=} & a \\
((a_1\, a_2) :: \pi)\bullet a & \stackrel{\text{def}}{=} & \begin{cases} a_1 & \text{if } \pi\bullet a = a_2 \\ a_2 & \text{if } \pi\bullet a = a_1 \\ \pi\bullet a & \text{otherwise} \end{cases}
\end{array}
$$

Overloading of $\_ \bullet \_$

- $\pi \bullet [] \stackrel{\text{def}}{=} []$ lists

  $\pi \bullet (x :: xs) \stackrel{\text{def}}{=} (\pi \bullet x) :: (\pi \bullet xs)$

- $\pi \bullet X \stackrel{\text{def}}{=} \{\pi \bullet x \mid x \in X\}$ sets

- $\pi \bullet (x_1, x_2) \stackrel{\text{def}}{=} (\pi \bullet x_1, \pi \bullet x_2)$ products

- $\pi \bullet \mathsf{None} \stackrel{\text{def}}{=} \mathsf{None}$ options

  $\pi \bullet \mathsf{Some}(x) \stackrel{\text{def}}{=} \mathsf{Some}(\pi \bullet x)$

- $\pi \bullet x \stackrel{\text{def}}{=} x$ integers, strings, bools

Permutation Properties

It is a bit inconvenient to ensure for every definition that it makes sense as a permutation operation. Therefore we require that every permutation definition satisfies:

- $[]\bullet x = x$
- $(\pi_1 @ \pi_2)\bullet x = \pi_1\bullet(\pi_2\bullet x)$
- $\pi_1 \sim \pi_2$ implies $\pi_1\bullet x = \pi_2\bullet x$

We refer to these properties as $pt_{\alpha,\iota}$ and refer to the type ι as permutation type if they are satisfied.

Permutation Types

The property of being a permutation type is in some sense hereditary:

- $pt_{\alpha,\alpha}$
- $pt_{\alpha,\iota\ list}$ provided $pt_{\alpha,\iota}$

  similar for sets, products and options
- $pt_{\alpha,nat}$, $pt_{\alpha,string}$, $pt_{\alpha,bool}$

The nominal datatype-package "knows" about a type being a permutation type by using axiomatic type-classes (I am happy to explain more about them, if wanted).

Permutations on Functions

Given $f : \iota_1 \Rightarrow \iota_2$ and

- $\pi \bullet f \stackrel{\text{def}}{=} \lambda x.\pi \bullet (f(\pi^{-1} \bullet x))$

then pt_{α,ι_1} and pt_{α,ι_2} imply $pt_{\alpha,\iota_1 \Rightarrow \iota_2}$.

The definition on functions implies that

- $\pi \bullet (f\ x) = (\pi \bullet f)(\pi \bullet x)$

holds for permutation types.

Support and Freshness

The support of an object $x : \iota$ is a set of atoms α:

$$\mathsf{supp}_\alpha\ x \stackrel{\text{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\ b)\bullet x \neq x\}\}$$

An atom is fresh for an x, if it is not in the support of x:

$$a \# x \stackrel{\text{def}}{=} a \notin \mathsf{supp}_\alpha(x)$$

I will often drop the α in supp_α.

Support and Freshness

The support of an object $x : \iota$ is a set of atoms α:

$$\mathsf{supp}_\alpha\ x \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\ b) \bullet x \neq x\}\}$$

An a OK, this definition is a tiny bit com- he
supp plicated, so let's go slowly...

$$a \# x \stackrel{\mathsf{def}}{=} a \notin \mathsf{supp}_\alpha(x)$$

I will often drop the α in supp_α.

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

$$a\text{:} \quad (a\,?)\bullet c \neq c$$

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

a: $(a\,?)\bullet c \neq c$ no

b: $(b\,?)\bullet c \neq c$

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

$$\begin{array}{lll} a: & (a\,?)\bullet c \neq c & \text{no} \\ b: & (b\,?)\bullet c \neq c & \text{no} \\ c: & (c\,?)\bullet c \neq c & \end{array}$$

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \overset{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

a:	$(a\,?)\bullet c \neq c$	no
b:	$(b\,?)\bullet c \neq c$	no
c:	$(c\,?)\bullet c \neq c$	yes
d:	$(d\,?)\bullet c \neq c$	

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

a:	$(a\,?)\bullet c \neq c$	no
b:	$(b\,?)\bullet c \neq c$	no
c:	$(c\,?)\bullet c \neq c$	yes
d:	$(d\,?)\bullet c \neq c$	no
	$\vdots$	no

Support of an Atom

What is the support of the atom c?

$$\mathsf{supp}(c) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{infinite}\{b \mid (a\,b)\bullet c \neq c\}\}$$

Let's check the (infinitely many) atoms one by one:

So $\mathsf{supp}(c) = \{c\}$

a:	$(a\,?)\bullet c \neq c$	no
b:	$(b\,?)\bullet c \neq c$	no
c:	$(c\,?)\bullet c \neq c$	yes
d:	$(d\,?)\bullet c \neq c$	no
	$\vdots$	no

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\,\{b \mid (a\,b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\ \{b \mid (a\ b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid ((a\ b)\bullet x_1, (a\ b)\bullet x_2) \neq (x_1, x_2)\}\}$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \,|\, \mathsf{inf}\, \{b \,|\, (a\,b) \bullet (x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \,|\, \mathsf{inf}\{b \,|\, ((a\,b) \bullet x_1, (a\,b) \bullet x_2) \neq (x_1, x_2)\}\}$$

We know

$(x_1, x_2) = (y_1, y_2)$ iff $x_1 = y_1 \wedge x_2 = y_2$

hence

$(x_1, x_2) \neq (y_1, y_2)$ iff $x_1 \neq y_1 \vee x_2 \neq y_2$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\ \{b \mid (a\,b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid ((a\,b)\bullet x_1, (a\,b)\bullet x_2) \neq (x_1, x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1 \vee (a\,b)\bullet x_2 \neq x_2\}\}$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\ \{b \mid (a\,b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid ((a\,b)\bullet x_1, (a\,b)\bullet x_2) \neq (x_1, x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1 \vee (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\{a \mid \mathsf{inf}(\{b \mid (a\,b)\bullet x_1 \neq x_1\} \cup \{b \mid (a\,b)\bullet x_2 \neq x_2\})\}$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\,\{b \mid (a\,b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid ((a\,b)\bullet x_1, (a\,b)\bullet x_2) \neq (x_1, x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1 \vee (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\{a \mid \mathsf{inf}(\{b \mid (a\,b)\bullet x_1 \neq x_1\} \cup \{b \mid (a\,b)\bullet x_2 \neq x_2\})\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1\} \vee \mathsf{inf}\{b \mid (a\,b)\bullet x_2 \neq x_2\}\}$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\ \{b \mid (a\,b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid ((a\,b)\bullet x_1, (a\,b)\bullet x_2) \neq (x_1, x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1 \vee (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\{a \mid \mathsf{inf}(\{b \mid (a\,b)\bullet x_1 \neq x_1\} \cup \{b \mid (a\,b)\bullet x_2 \neq x_2\})\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1\} \vee \mathsf{inf}\{b \mid (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1\}\} \cup \{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_2 \neq x_2\}\}$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \,|\, \mathsf{inf}\, \{b \,|\, (a\, b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \,|\, \mathsf{inf}\{b \,|\, ((a\, b)\bullet x_1, (a\, b)\bullet x_2) \neq (x_1, x_2)\}\}$$

$$\{a \,|\, \mathsf{inf}\{b \,|\, (a\, b)\bullet x_1 \neq x_1 \vee (a\, b)\bullet x_2 \neq x_2\}\}$$

$$\{a \,|\, \mathsf{inf}(\{b \,|\, (a\, b)\bullet x_1 \neq x_1\} \cup \{b \,|\, (a\, b)\bullet x_2 \neq x_2\})\}$$

$$\{a \,|\, \mathsf{inf}\{b \,|\, (a\, b)\bullet x_1 \neq x_1\} \vee \mathsf{inf}\{b \,|\, (a\, b)\bullet x_2 \neq x_2\}\}$$

$$\{a \,|\, \mathsf{inf}\{b \,|\, (a\, b)\bullet x_1 \neq x_1\}\} \cup \{a \,|\, \mathsf{inf}\{b \,|\, (a\, b)\bullet x_2 \neq x_2\}\}$$

$$\mathsf{supp}(x_1) \qquad \cup \qquad \mathsf{supp}(x_2)$$

Support of a Product

$$\mathsf{supp}(x_1,x_2) \stackrel{\mathsf{def}}{=} \{a \mid \mathsf{inf}\,\{b \mid (a\,b)\bullet(x_1,x_2) \neq (x_1,x_2)\}\}$$

$$\{a \mid \mathsf{inf}\{b$$

$$\text{So}\quad \mathsf{supp}(x_1,x_2) = \mathsf{supp}(x_1) \cup \mathsf{supp}(x_2)$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1 \vee (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\{a \mid \mathsf{inf}(\{b \mid (a\,b)\bullet x_1 \neq x_1\} \cup \{b \mid (a\,b)\bullet x_2 \neq x_2\})\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1\} \vee \mathsf{inf}\{b \mid (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_1 \neq x_1\}\} \cup \{a \mid \mathsf{inf}\{b \mid (a\,b)\bullet x_2 \neq x_2\}\}$$

$$\mathsf{supp}(x_1) \qquad\qquad \cup \qquad\qquad \mathsf{supp}(x_2)$$

Some Simple Properties

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

  $$a \mathrel{\#} (x_1, x_2) \text{ iff } a \mathrel{\#} x_1 \wedge a \mathrel{\#} x_2$$

- $\text{supp}_\alpha\ (a : \alpha) = \{a\}$
- supp $[] = \varnothing$,
  $\text{supp}(x :: xs) = \text{supp}(x) \cup \text{supp}(xs)$
- $\text{supp}(\text{None}) = \varnothing$,
  $\text{supp}(\text{Some}(x)) = \text{supp}(x)$
- $\text{supp}(1) = \text{supp}(\text{"}s\text{"}) = \text{supp}(\mathit{True}) = \varnothing$

Some Simple Properties

- su
- su
- su
  su
- su
  su

The support of "finitary" structures is usually quite simple: for example the support of a lambda-term t is the set of atoms occuring in t.

$$\begin{array}{lcl} \pi \bullet a & \stackrel{\text{def}}{=} & \ldots \\ \pi \bullet (t_1\, t_2) & \stackrel{\text{def}}{=} & (\pi \bullet t_1)\, (\pi \bullet t_2) \\ \pi \bullet \lambda a.t & \stackrel{\text{def}}{=} & \lambda(\pi \bullet a)\, (\pi \bullet t) \end{array}$$

- supp(1) = supp$(\text{"}s\text{"})$ = supp(True) = $\varnothing$

FYI: Infinitary Structures

- supp $A = \varnothing$ $\quad$ A set of all atoms in α

  since $\forall a, b.\ (a\ b) \bullet A = A$
- supp $F = \{a_1, \ldots, a_n\}$ assuming F is a finite set of atoms $a_1, \ldots, a_n$
- not every set of atoms has finite support: e.g. "$A/2$"
- the support of functions is even more interesting (later)

Existence of a Fresh Atom

Q: Why do we assume that there are infinitely many atoms?

A: For any finitely supported x:

$$\boxed{\exists c.\ \ c \# x}$$

If something is finitely supported, then we can always choose a fresh atom (also for finitely supported functions).

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$\pi \bullet \mathsf{supp}(x) = \mathsf{supp}(\pi \bullet x)$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{aligned}
& \pi \bullet \mathsf{supp}(x) \\
\stackrel{\mathsf{def}}{=}\ & \pi \bullet \{a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\}
\end{aligned}
$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{aligned}
&\pi \bullet \mathsf{supp}(x) \\
\stackrel{\text{def}}{=}\ &\pi \bullet \{a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\} \\
\stackrel{\text{def}}{=}\ &\{\pi \bullet a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\}
\end{aligned}
$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{aligned}
& \pi \bullet \mathsf{supp}(x) \\
\overset{\mathsf{def}}{=}\; & \pi \bullet \{a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\} \\
\overset{\mathsf{def}}{=}\; & \{\pi \bullet a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\} \\
=\; & \{\pi \bullet a \mid \mathsf{inf}\{\pi \bullet b \mid (a\,b) \bullet x \neq x\}\}
\end{aligned}
$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{aligned}
& \pi \bullet \mathsf{supp}(x) \\
\stackrel{\text{def}}{=}\; & \pi \bullet \{a \mid \mathsf{inf}\{b \mid (a\, b) \bullet x \neq x\}\} \\
\stackrel{\text{def}}{=}\; & \{\pi \bullet a \mid \mathsf{inf}\{b \mid (a\, b) \bullet x \neq x\}\} \\
=\; & \{\pi \bullet a \mid \mathsf{inf}\{\pi \bullet b \mid (a\, b) \bullet x \neq x\}\} \\
=\; & \{a \mid \mathsf{inf}\{b \mid (\pi^{-1} \bullet a \;\; \pi^{-1} \bullet b) \bullet x \neq x\}\}
\end{aligned}
$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{array}{ll}
& \pi \bullet \mathsf{supp}(x) \\
\stackrel{\text{def}}{=} & \pi \bullet \{a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\} \\
\stackrel{\text{def}}{=} & \{\pi \bullet a \mid \mathsf{inf}\{b \mid (a\,b) \bullet x \neq x\}\} \\
= & \{\pi \bullet a \mid \mathsf{inf}\{\pi \bullet b \mid (a\,b) \bullet x \neq x\}\} \\
= & \{a \mid \mathsf{inf}\{b \mid (\pi^{-1} \bullet a \;\; \pi^{-1} \bullet b) \bullet x \neq x\}\} \\
= & \{a \mid \mathsf{inf}\{b \mid \pi \bullet (\pi^{-1} \bullet a \;\; \pi^{-1} \bullet b) \bullet x \neq \pi \bullet x\}\}
\end{array}
$$

$$\boxed{x = y \quad \text{iff} \quad \pi \bullet x = \pi \bullet y}$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{aligned}
& \pi \bullet \mathsf{supp}(x) \\
\overset{\text{def}}{=}\ & \pi \bullet \{a \mid \mathsf{inf}\{b \mid (a\, b) \bullet x \neq x\}\} \\
\overset{\text{def}}{=}\ & \{\pi \bullet a \mid \mathsf{inf}\{b \mid (a\, b) \bullet x \neq x\}\} \\
=\ & \{\pi \bullet a \mid \mathsf{inf}\{\pi \bullet b \mid (a\, b) \bullet x \neq x\}\} \\
=\ & \{a \mid \mathsf{inf}\{b \mid (\pi^{-1} \bullet a \ \ \pi^{-1} \bullet b) \bullet x \neq x\}\} \\
=\ & \{a \mid \mathsf{inf}\{b \mid \pi \bullet (\pi^{-1} \bullet a \ \ \pi^{-1} \bullet b) \bullet x \neq \pi \bullet x\}\} \\
=\ & \{a \mid \mathsf{inf}\{b \mid (a\, b) \bullet \pi \bullet x \neq \pi \bullet x\}\}
\end{aligned}
$$

$$\boxed{\pi_1 \bullet (\pi_2 \bullet x) = (\pi_1 \bullet \pi_2) \bullet (\pi_1 \bullet x)}$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$
\begin{aligned}
& \pi \bullet \mathsf{supp}(x) \\
\stackrel{\text{def}}{=}\ & \pi \bullet \{a \,|\, \mathsf{inf}\{b \,|\, (a\, b) \bullet x \neq x\}\} \\
\stackrel{\text{def}}{=}\ & \{\pi \bullet a \,|\, \mathsf{inf}\{b \,|\, (a\, b) \bullet x \neq x\}\} \\
=\ & \{\pi \bullet a \,|\, \mathsf{inf}\{\pi \bullet b \,|\, (a\, b) \bullet x \neq x\}\} \\
=\ & \{a \,|\, \mathsf{inf}\{b \,|\, (\pi^{-1} \bullet a \;\; \pi^{-1} \bullet b) \bullet x \neq x\}\} \\
=\ & \{a \,|\, \mathsf{inf}\{b \,|\, \pi \bullet (\pi^{-1} \bullet a \;\; \pi^{-1} \bullet b) \bullet x \neq \pi \bullet x\}\} \\
=\ & \{a \,|\, \mathsf{inf}\{b \,|\, (a\, b) \bullet \pi \bullet x \neq \pi \bullet x\}\} \\
\stackrel{\text{def}}{=}\ & \mathsf{supp}(\pi \bullet x)
\end{aligned}
$$

More Interesting Prop's

Assuming $pt_{\alpha,\iota}$:

$$\pi \bullet \mathrm{supp}(x)$$

$$\stackrel{\mathrm{def}}{=} \pi \bullet \{a \mid \mathrm{inf}\{b \mid (a\,b) \bullet x \neq x\}\}$$

$$\stackrel{\mathrm{def}}{=} \{\pi \bullet a \mid \mathrm{inf}\{b \mid (a\,b) \bullet x \neq x\}\}$$

$$= \{\pi \bullet a \mid \mathrm{inf}\{\pi \bullet b \mid (a\,b) \bullet x \neq x\}\}$$

As a corollary: $a \# x$ iff $\pi \bullet a \# \pi \bullet x$.

$$= \{a \mid \mathrm{inf}\{b \mid \pi \bullet (\pi^{-1} \bullet a \;\; \pi^{-1} \bullet b) \bullet x \neq \pi \bullet x\}\}$$

$$= \{a \mid \mathrm{inf}\{b \mid (a\,b) \bullet \pi \bullet x \neq \pi \bullet x\}\}$$

$$\stackrel{\mathrm{def}}{=} \mathrm{supp}(\pi \bullet x)$$

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\, b) \bullet x = x$$

Proof: case $a = b$ clear

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\, b)\bullet x = x$$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\, c)\bullet x \neq x\}$ from Ass. +Def. of $\#$

$\mathsf{fin}\{c \mid (b\, c)\bullet x \neq x\}$

$$a \# x \stackrel{\text{def}}{=} a \notin \mathsf{supp}(x)$$

$$\mathsf{supp}(x) \stackrel{\text{def}}{=} \{a \mid \mathsf{inf}\{c \mid (a\, c)\bullet x \neq x\}\}$$

It is as Simple as That...

$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$

$\mathsf{fin}\{c \mid (b\,c)\bullet x \neq x\}$

(2) $\mathsf{fin}(\{c \mid (a\,c)\bullet x \neq x\} \cup \{c \mid (b\,c)\bullet x \neq x\})$ f. (1)

It is as Simple as That...

$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b){\bullet}x = x$

Proof: case $a \neq b$:

(1) fin$\{c \mid (a\,c){\bullet}x \neq x\}$ from Ass. +Def. of $\#$
fin$\{c \mid (b\,c){\bullet}x \neq x\}$

(2') fin$\{c \mid (a\,c){\bullet}x \neq x \vee (b\,c){\bullet}x \neq x\}$ f. (1)

It is as Simple as That...

$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\, b) \bullet x = x$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\, c) \bullet x \neq x\}$ from Ass. +Def. of $\#$

$\mathsf{fin}\{c \mid (b\, c) \bullet x \neq x\}$

(2') $\mathsf{fin}\{c \mid (a\, c) \bullet x \neq x \vee (b\, c) \bullet x \neq x\}$ f. (1)

(3) $\mathsf{inf}\{c \mid \neg((a\, c) \bullet x \neq x \vee (b\, c) \bullet x \neq x)\}$ f. (2')

Given a finite set of atoms,
its 'co-set' must be infinite.

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$

$\mathsf{fin}\{c \mid (b\,c)\bullet x \neq x\}$

(2') $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3') $\mathsf{inf}\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2')

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

<u>Proof:</u> case $a \neq b$:

(1) fin$\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$

fin$\{c \mid (b\,c)\bullet x \neq x\}$

(2') fin$\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3') inf$\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2')

(4) (i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$ for a $c \in$ (3')

If a set is infinite, it must contain a few elements. Let's pick c so that $c \neq a, b$.

It is as Simple as That...

$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$
$\mathsf{fin}\{c \mid (b\,c)\bullet x \neq x\}$

(2') $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3') $\mathsf{inf}\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2')

(4) (i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$ for a $c \in$ (3')

(5) $(a\,c)\bullet x = x$ by (4i)

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$
$\mathsf{fin}\{c \mid (b\,c)\bullet x \neq x\}$

(2') $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3') $\mathsf{inf}\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2')

(4) (i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$ for a $c \in$ (3')

(5) $(a\,c)\bullet x = x$ by (4i)

(6) $(b\,c)\bullet(a\,c)\bullet x = (b\,c)\bullet x$ by bij.

bij.: $x = y$ iff $\pi\bullet x = \pi\bullet y$

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\, b) \bullet x = x$$

Proof: case $a \neq b$:

(1) fin$\{c \mid (a\, c) \bullet x \neq x\}$ from Ass. +Def. of $\#$
fin$\{c \mid (b\, c) \bullet x \neq x\}$

(2′) fin$\{c \mid (a\, c) \bullet x \neq x \vee (b\, c) \bullet x \neq x\}$ f. (1)

(3′) inf$\{c \mid (a\, c) \bullet x = x \wedge (b\, c) \bullet x = x)\}$ f. (2′)

(4) (i) $(a\, c) \bullet x = x$ (ii) $(b\, c) \bullet x = x$ for a $c \in$ (3′)

(5) $(a\, c) \bullet x = x$ by (4i)

(6′) $(b\, c) \bullet (a\, c) \bullet x = x$ by bij.,(4ii)

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

Proof: case $a \neq b$:

(1)	$\mathrm{fin}\{c \mid (a\,c)\bullet x \neq x\}$	from Ass. +Def. of $\#$
	$\mathrm{fin}\{c \mid (b\,c)\bullet x \neq x\}$	
(2')	$\mathrm{fin}\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$	f. (1)
(3')	$\mathrm{inf}\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$	f. (2')
(4)	(i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$	for a $c \in$ (3')
(5)	$(a\,c)\bullet x = x$	by (4i)
(6')	$(b\,c)\bullet(a\,c)\bullet x = x$	by bij.,(4ii)
(7)	$(a\,c)\bullet(b\,c)\bullet(a\,c)\bullet x = (a\,c)\bullet x$	by bij.

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

Proof: case $a \neq b$:

(1) fin$\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$
fin$\{c \mid (b\,c)\bullet x \neq x\}$

(2') fin$\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3') inf$\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2')

(4) (i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$ for a $c \in$ (3')

(5) $(a\,c)\bullet x = x$ by (4i)

(6') $(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4ii)

(7') $(a\,c)\bullet(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4i)

It is as Simple as That...

$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$

<u>Proof:</u> case $a \neq b$:

(1) fin$\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$

fin$\{c \mid (b\,c)\bullet x \neq x\}$

(2′) fin$\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3′) inf$\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2′)

(4) (i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$ for a $c \in$ (3′)

(5) $(a\,c)\bullet x = x$ by (4i)

(6′) $(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4ii)

(7′) $(a\,c)\bullet(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4i)

$$(a\,c)(b\,c)(a\,c) \sim (a\,b)$$

It is as Simple as That...

$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$

Proof: case $a \neq b$:

(1) $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$

$\mathsf{fin}\{c \mid (b\,c)\bullet x \neq x\}$

(2′) $\mathsf{fin}\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3′) $\mathsf{inf}\{c \mid (a$ f. (2′)

(4) (i) $(a\,c)\bullet$ $a\ c \in$ (3′)

3rd property of $pt_{\alpha,\iota}$:

$\pi_1 \sim \pi_2 \Rightarrow \pi_1\bullet x = \pi_2\bullet x$

(5) $(a\,c)\bullet x = x$ by (4i)

(6′) $(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4ii)

(7′) $(a\,c)\bullet(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4i)

(8) $(a\,b)\bullet x = x$ by 3rd. prop.

It is as Simple as That...

$$pt_{\alpha,\iota} : a \# x \wedge b \# x \Rightarrow (a\,b)\bullet x = x$$

Proof: case $a \neq b$:

(1) $\mathrm{fin}\{c \mid (a\,c)\bullet x \neq x\}$ from Ass. +Def. of $\#$
$\quad\;\mathrm{fin}\{c \mid (b\,c)\bullet x \neq x\}$

(2′) $\mathrm{fin}\{c \mid (a\,c)\bullet x \neq x \vee (b\,c)\bullet x \neq x\}$ f. (1)

(3′) $\mathrm{inf}\{c \mid (a\,c)\bullet x = x \wedge (b\,c)\bullet x = x)\}$ f. (2′)

(4) (i) $(a\,c)\bullet x = x$ (ii) $(b\,c)\bullet x = x$ for a $c \in$ (3′)

(5) $(a\,c)\bullet x = x$ by (4i)

(6′) $(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4ii)

(7′) $(a\,c)\bullet(b\,c)\bullet(a\,c)\bullet x = x$ by bij.,(4i)

(8) $(a\,b)\bullet x = x$ by 3rd. prop.

Done.

Nominal Abstractions

We are now going to specify what abstraction 'abstractly' means: it is an operation $[\_].(\_) : \alpha \Rightarrow \iota \Rightarrow \iota$ which has to satisfy:

- $\pi \bullet ([a].x) = [\pi \bullet a].(\pi \bullet x)$
- $[a].x = [b].y$ iff

  $(a = b \wedge x = y) \vee$
  $(a \neq b \wedge x = (a\,b) \bullet y \wedge a \# y)$

At the moment we assume that only one atom at a time is abstracted; more interesting abstractions are certainly possible (research).

Nominal Abstractions

Remember the definition of α-equivalence from the beginning:

$$\frac{t_1 \approx t_2}{\lambda a.t_1 \approx \lambda a.t_2} \qquad \frac{a \neq b \quad t_1 \approx (a\,b)\bullet t_2 \quad a \# t_2}{\lambda a.t_1 \approx \lambda b.t_2}$$

- $[a].x = [b].y$ iff

  $(a = b \wedge x = y) \vee$
  $(a \neq b \wedge x = (a\,b)\bullet y \wedge a \# y)$

At the moment we assume that only one atom at a time is abstracted; more interesting abstractions are certainly possible (research).

Nominal Abstractions

We are now going to specify what abstraction 'abstractly' means: it is an operation $[\_].(\_) : \alpha \Rightarrow \iota \Rightarrow \iota$ which has to satisfy:

- $\pi \bullet ([a].x) = [\pi \bullet a].(\pi \bullet x)$
- $[a].x = [b].y$ iff

  $(a = b \wedge x = y) \vee$
  $(a \neq b \wedge x = (a\,b) \bullet y \wedge a \# y)$

At the moment we assume that only one atom at a time is abstracted; more interesting abstractions are certainly possible (research).

Freshness and Abstractions

Given $pt_{\alpha,\iota}$, $\text{finite}(\text{supp } x)$ and $a \neq b$ then

$$a \# x \text{ iff } a \# [b].x$$

Proof. There exists a c with $c \# (a, b, x, [b].x)$.

($\Leftarrow$) From $a \# [b].x$ and $c \# [b].x$

$$[b].x = (a\,c)\bullet([b].x) = [b].(a\,c)\bullet x$$

Hence $x = (a\,c)\bullet x$. Now from $c \# x$:

$$c \# x \Leftrightarrow (a\,c)\bullet c \# (a\,c)\bullet x \Leftrightarrow a \# x$$

Freshness and Abstractions

Given $pt_{\alpha,\iota}$, finite(supp x) and $a \neq b$ then

$$a \;\#\; x \text{ iff } a \;\#\; [b].x$$

Proof. There exists a c with $c \;\#\; (a, b, x, [b].x)$.

($\Rightarrow$) From $c \;\#\; [b].x$ we also have

$$(a\,c){\bullet}c \;\#\; (a\,c){\bullet}[b].x$$

and

$$a \;\#\; [b].(a\,c){\bullet}x$$

Because $a \;\#\; x$ and $c \;\#\; x$, $(a\,c){\bullet}x = x$.

Freshness and Abstractions

We also have

$$a \# [a].x$$

Again from $c \# (a, x, [a].x)$ we can infer

$$\begin{aligned} c \# [a].x \quad &\Leftrightarrow \quad (a\,c)\bullet c \# (a\,c)\bullet[a].x \\ &\Leftrightarrow \quad a \# [c].(a\,c)\bullet x. \end{aligned}$$

However:

$$[c].(a\,c)\bullet x = [a].x$$

(since $c \neq a$, $[c].(a\,c)\bullet x = [a].x$
iff $(a\,c)\bullet x = (a\,c)\bullet x \wedge c \# x$)