We start with Atoms

- We distinguish a set of types $\{\alpha_1, \ldots, \alpha_n\}$, called atoms (that is nothing Isabelle-specific, but a notational convention for what follows).
- There is a type abbreviation for permutations α prm:

$$\alpha\ \mathsf{prm} := (\alpha \times \alpha)\ \mathsf{list}$$

- We introduce a polymorphic constant:

$$\_\bullet\_ : \alpha\ \mathsf{prm} \Rightarrow \iota \Rightarrow \iota$$

for a permutation of type α to act on an object of type ι.

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
- $\pi \bullet () \stackrel{\text{def}}{=} ()$ unit
- $\pi \bullet (x_1, x_2) \stackrel{\text{def}}{=} (\pi \bullet x_1, \pi \bullet x_2)$ products
- $\pi \bullet \mathsf{None} \stackrel{\text{def}}{=} \mathsf{None}$ options

  $\pi \bullet \mathsf{Some}(x) \stackrel{\text{def}}{=} \mathsf{Some}(\pi \bullet x)$
- $\pi \bullet f \stackrel{\text{def}}{=} \lambda x. \pi \bullet (f(\pi^{-1} \bullet x))$ functions
- $\pi \bullet x \stackrel{\text{def}}{=} x$ integers, strings, bools

Permutations

- 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
- **permutation equality**, two permutations π and π' (of type α prm) are equal iff

$$\pi \sim \pi' \stackrel{\text{def}}{=} \forall a{::}\alpha.\ \pi\bullet a = \pi'\bullet a$$

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

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

I will often drop the α in supp_α.

Support and Freshness

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

We can calculate what the support/freshness in "uninteresting" cases, such as products, lists, options, unit, bool... E.g. lists

$$\text{supp}\,[\,] = \{\}$$
$$\text{supp}\,(x :: xs) = \text{supp}\,x \cup \text{supp}\,xs$$

but not in "interesting" cases like sets, functions,...

I will often drop the α in supp_α.

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: assuming $pt_{\alpha,\iota}$, pt_{α,ι_1} and pt_{α,ι_2}

- $pt_{\alpha,\iota\; list}$, $pt_{\alpha,\iota_1 \times \iota_2}$
- $pt_{\alpha,nat}$, $pt_{\alpha,string}$, $pt_{\alpha,bool}$

    similar for sets, functions, options,...

The nominal datatype-package can find out that a type is a permutation type by using **axiomatic type-classes** (no action from the user needed). The implementer only has to establish the property for each type-constructor.

Finite Support

We can do almost the same with **finitely supported** permutation types, written $fs_{\alpha,\iota}$: assuming $fs_{\alpha,\iota}$, fs_{α,ι_1} and fs_{α,ι_2}

- $fs_{\alpha,\iota\, list}$, $fs_{\alpha,\iota_1 \times \iota_2}$
- $fs_{\alpha,nat}$, $fs_{\alpha,string}$, $fs_{\alpha,bool}$

  similar for options,... but **not** sets and functions

Again the nominal datatype-package can find out that a type is finitely supported for "finitary" types. For sets and functions the user has to ensure finite support himself.

Atom Properties

Atoms have to satisfy

- $[]\bullet x = x$
- $((a_1\, a_2)::\pi)\bullet a = \begin{cases} a_1 & \text{if } \pi\bullet a = a_2 \\ a_2 & \text{if } \pi\bullet a = a_1 \\ \pi\bullet a & \text{otherwise} \end{cases}$
- there have to be infinitely many of them

We refer to these properties as at_α. The properties imply that the support of an atom is the atoms itself.

General Properties

From this one can now start to prove general nominal logic lemmas, such as:

- if $pt_{\alpha,\iota}$, at_α then

$$\pi \bullet \mathsf{supp}_\alpha\,(x :: \iota) = \mathsf{supp}_\alpha\,(\pi \bullet x)$$

- Let $f :: \iota_1 \Rightarrow \iota_2$. If pt_{α,ι_1}, at_α then

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

- ...

The User Comes Into Play

- When the user issues

$$\text{atom\_decl}\ \alpha_1 \ldots \alpha_n$$

the nominal datatype package makes n "copies" of the integers; defines an appropriate permutation operation and ensures that the type α_i satisfies at_{α_i}

- The same for

$$\text{nominal\_datatype}\ \iota = \ldots$$

$pt_{\alpha,\iota}$, $fs_{\alpha,\iota}$ for all α.

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

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

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

We know

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

hence

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

Support of a Product

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

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

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

Support of a Product

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

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

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

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

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

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

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

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

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

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

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

Support of a Product

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

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

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

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

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

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

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

```
lemma supp_prod:
shows "supp (x₁, x₂) = (supp x₁) ∪ (supp x₂)"
proof -
have "supp (x₁, x₂) = {a.inf{b.[(a, b)]•(x₁, x₂) ≠ (x₁, x₂)}}"
by (simp add: supp_def)
also have "... = {a.inf{b.([(a, b)]•x₁, [(a, b)]•x₂) ≠ (x₁, x₂)}}"
by simp
also have "... = {a.inf{b.[(a, b)]•x₁ ≠ x₁ ∨ [(a, b)]•x₂ ≠ x₂}}"
by simp
also have "... = {a.inf({b.[(a, b)]•x₁ ≠ x₁} ∪ {b.[(a, b)]•x₂ ≠ x₂})}"
by (simp only: Collect_disj_eq)
also have "... = {a.inf{b.[(a, b)]•x₁ ≠ x₁} ∨ inf{b.[(a, b)]•x₂ ≠ x₂}}"
by simp
also have "... = {a.inf{b.[(a, b)]•x₁ ≠ x₁} ∪ {a.inf{b.[(a, b)]•x₂ ≠ x₂}
by force
also have "... = (supp x₁) ∪ (supp x₂)" by (simp add: supp_def)
finally show "supp (x₁, x₂) = (supp x₁) ∪ (supp x₂)" by simp
qed
```