Sets vs. Types

What is a set?

A collection of elements.

$x \in A$ – x is a member of A

$x \notin A$

$A \cup B$ union

$A \cap B$ intersection

$A \times B$ cartesian product

$A + B$ disjoint union

Any thing can potentially be an element of any set.

Any x can be an element of many different sets.

What is a type?

A collection of elements.

$x : A$ – x is a member of A

No union

No intersection

$A \times B$ cartesian product

$A + B$ disjoint union

Every "thing" is a member of exactly one type.

No x can be an element of more than one type.

Define disjoint union

$$A + B = \{(0, x) \mid x \in A\} \cup \{(1, y) \mid y \in B\}$$

$$\frac{x : A}{\mathrm{left}_{A,B}\, x : A + B} \qquad \frac{y : B}{\mathrm{right}_{A,B}\, y : A + B}$$

Simply-typed lambda calculus

In set theory:
$A \to B = \{ f \mid f : A \to B \}$

Types: A, B, C

$A \times B$ cartesian product

$A \to B$ function space

What is an element of $A \times B$?

If $x : A$ and $y : B$, then $(x,y) : A \times B$

We also write $\mathrm{fst}(x,y) = x$ $\quad \mathrm{fst} : A \times B \to A$

$\mathrm{snd}(x,y) = y$ $\quad \mathrm{snd} : A \times B \to B$

What is an element of $A \to B$?

In lambda calculus, we write

$\lambda x . x^2$ "lambda abstraction"

as a notation for the function that maps x to x^2

In math, to define a function $\mathbb{N} \to \mathbb{N}$, we use a notation such as:
$f : x \mapsto x^2$
or
$f(x) = x^2$

$x+y$ "sum"
(x,y) "pair"
xy "product"

Example $(\lambda x . x^2)(5) = 5^2 = 25$

$(x, (y,z))$

$x + (x+y)$

$((x \cdot y) + z) \cdot w$

$(x, \lambda z . z^2)$

$(x \cdot y + z) \cdot w$

$\lambda x . (x, x^2)$

Notational conventions:

$f = \lambda x . (\lambda y . (x,y))$ $= \lambda x . \lambda y . (x,y)$
$= \lambda x y . (x,y)$

$f(5) = \lambda y . (5,y)$
$f(5)(7) = (5,7)$

1930's

Haskell Curry
Alonzo Church

$f : A \times B \to C$
uncurried

$g : A \to (B \to C)$
curried

$g\,x\,y$

"currying"

If $f : A \to B$ and $x : A$
we write $f x : B$ for the application of f to x.

More formally, the simply-typed lambda calculus is defined as follows:

<u>Types</u>   <u>Given</u> an infinite set of <u>type variables</u> $TV = \{\alpha, \beta, \gamma, \dots\}$

The set of <u>types</u> is defined by the following grammar

BNF Baccus-Naur Form

$$\boxed{\text{Type} \quad A, B \ ::= \ \alpha \mid A \times B \mid A \to B}$$

In more traditional mathematical terms, this means the following:

Let Σ be an <u>alphabet</u> consisting of the following symbols: TV and the symbols $($, $)$, $\times$, $\to$. In other words

$$\Sigma = TV \cup \{ (,), \times, \to \}$$

Let Σ^* be the set of <u>strings</u>, i.e. finite sequences, of symbols from the alphabet Σ.

<u>Examples:</u> $(\alpha \to \beta) \in \Sigma^*$

$) \to \alpha\alpha)) \in \Sigma^*$

We write $\varepsilon \in \Sigma^*$ for the empty string.

Formally, the set of <u>types</u> is the smallest subset $T \subseteq \Sigma^*$ with the following properties:

(1) Whenever $\alpha \in TV$ is a type variable, then $\alpha \in T$

(2) Whenever $A, B \in T$, then

$$(A \times B) \in T$$

(3) Whenever $A, B \in T$, then

$$(A \to B) \in T.$$

<u>Examples</u> The following are elements of T:

$$\alpha \qquad (\alpha \to \beta)$$

$$(\alpha \times (\beta \to ((\alpha \times \gamma) \to \alpha)))$$

The following are not elements of T:

$$\varepsilon \qquad (((\qquad \alpha \to \beta$$

$$\alpha\beta$$

<u>Notational convention</u> We will often drop excessive parentheses.

- The outermost parentheses can be dropped, e.g. we write $A \to (B \to C)$ instead of $(A \to (B \to C))$.
- $\times$ binds more strongly than $\to$, so we write $A \times B \to C \times D$ for $(A \times B) \to (C \times D)$
- $\times$ associates to the left, so we write $A \times B \times C$ for $(A \times B) \times C$
- $\to$ associates to the right, so we write $A \to B \to C$ for $A \to (B \to C)$

Terms ("lambda terms").

Given an infinite set $V = \{x, y, z, \ldots\}$ of variables

Terms are defined by the following BNF:

$$\text{Terms} \quad M, N ::= x \mid (M, N) \mid \mathrm{fst}\, M \mid \mathrm{snd}\, M \mid \lambda x{:}A.M \mid MN$$

As usual, this BNF is an abbreviation for an inductive definition of a set of strings. Parentheses must be added to make the terms unambiguous.

$$(\lambda x^A.(\lambda y^B.(yx)))$$

Conventions

- Outermost brackets can be dropped
- Application associates to the left, so

  MNP means $(MN)P$
- The part "after the dot" extends as far to the right as possible, i.e.

  $\lambda x^A.MN$ means $\lambda x^A.(MN)$

  and not $(\lambda x^A.M)N$

Types $A, B ::= \alpha \mid A \times B \mid A \to B$

Terms $M, N ::= x \mid (M, N) \mid \mathrm{fst}\ M \mid \mathrm{snd}\ M \mid \lambda x{:}A.M \mid MN$

Terms have types

$$\lambda x{:}A.\,x \quad : \quad A \to A$$

$$x:A,\ y:B \ \vdash\ (x,y) : A \times B$$

$$x:A,\ y:B \ \vdash\ (x,x) : A \times A$$

$$x:A,\ y:B \ \vdash\ (\lambda z{:}C.\,z,\ \lambda w{:}D.\,x) : (C \to C) \times (D \to A)$$

Typing Judgement "entails"

$$x_1 : A_1, \dots, x_n : A_n \ \vdash\ M : B$$

$$x:A,\ y:B,\ z:C \ \vdash\ (x,z) : A \times C$$

$$x:A,\ y:B,\ z:C \ \vdash ((x, \lambda w{:}B.\,z), x) : (A \times (B \to C)) \times A$$

$$\vdash \lambda x{:}A.\,\lambda y{:}B.(x,y) : A \to B \to (A \times B)$$

$$\vdash \lambda f{:}A \to B.\,(\,?\,,\,?\,) : (A \to B) \to (A \times B)$$

Not possible

$$x:A,\ y:B \ \vdash \lambda f{:}A \to B.\,(x,y) : (A \to B) \to (A \times B)$$

$$\vdash \lambda f{:}A\to B.\ \lambda g{:}B\to C.\ \lambda x{:}A.\ g(f x) \ : (A\to B)\to(B\to C)\to(A\to C)$$