Expressions (and reductions)

The usual way to calculate an expression, one step at a time, with '='s:

$$\begin{array}{rcl} 2\cdot 3+4\cdot 5 & = & 2\cdot 3+20 \\ & = & 6+20 \\ & = & 26 \end{array}$$

$$\begin{array}{rcl} 2\cdot 3+4\cdot 5 & = & 6+4\cdot 5 \\ & = & 6+20 \\ & = & 26 \end{array}$$

Each '=' corresponds to a '$\longrightarrow$' in the reduction diagram below.

A notation for calculating the value of an expression by calculating the values of all its subexpressions:

$$\underbrace{\underbrace{2\cdot 3}_{6}+\underbrace{4\cdot 5}_{20}}_{26}$$

Each '=' in the previous diagram corresponds to applying one '$\underbrace{\quad}$'.

A *reduction diagram* for $2\cdot 3+4\cdot 5$:
(See Hindley/Seldin, pages 14 and 17)

$$\begin{array}{ccccc} 2\cdot 3+4\cdot 5 & \longrightarrow & 2\cdot 3+20 & & \\ \downarrow & & \downarrow & & \\ 6+4\cdot 5 & \longrightarrow & 6+20 & \longrightarrow & 26 \end{array}$$

Note that when we can choose two subexpressions to calculate the '$\downarrow$' evaluatess the leftmost one, and the '$\rightarrow$' evaluates the rightmost one.

The *subexpressions* of $2\cdot 3+4\cdot 5$:

$$\underbrace{\underbrace{\underbrace{2}\cdot\underbrace{3}}+\underbrace{\underbrace{4}\cdot\underbrace{5}}}$$

Exercise:
Do the same as above for these expressions:
a) $2\cdot(3+4)+5\cdot 6$
b) $2+3+4$
c) $2+3+4+5$
(Improvise when needed)

Expressions with variables

If $a = 5$ and $b = 2$, then:

$$\underbrace{(\underbrace{\underbrace{a}_{5}+\underbrace{b}_{2}}_{7})\cdot(\underbrace{\underbrace{a}_{5}-\underbrace{b}_{2}}_{3})}_{21}$$

If $a = 10$ and $b = 1$, then:

$$\underbrace{(\underbrace{\underbrace{a}_{10}+\underbrace{b}_{1}}_{11})\cdot(\underbrace{\underbrace{a}_{10}-\underbrace{b}_{1}}_{9})}_{99}$$

We know – by algebra, which is not for (tiny) children –
that $(a + b) \cdot (a - b) = a \cdot a - b \cdot b$ is true for all $a, b \in \mathbb{R}$
We know – without algebra – how to test
"$(a + b) \cdot (a - b) = a \cdot a - b \cdot b$"
for specific values of a and b...

If $a = 5$ and $b = 2$, then:

$$\underbrace{\underbrace{(\underbrace{\underbrace{a}_{5}+\underbrace{b}_{2}}_{7})\cdot(\underbrace{\underbrace{a}_{5}-\underbrace{b}_{2}}_{3})}_{21}=\underbrace{\underbrace{\underbrace{a}_{5}\cdot\underbrace{a}_{5}}_{25}-\underbrace{\underbrace{b}_{2}\cdot\underbrace{b}_{2}}_{4}}_{21}}_{\text{true}}$$

If $a = 10$ and $b = 1$, then:

$$\underbrace{\underbrace{(\underbrace{\underbrace{a}_{10}+\underbrace{b}_{1}}_{11})\cdot(\underbrace{\underbrace{a}_{10}-\underbrace{b}_{1}}_{9})}_{99}=\underbrace{\underbrace{\underbrace{a}_{10}\cdot\underbrace{a}_{10}}_{100}-\underbrace{\underbrace{b}_{1}\cdot\underbrace{b}_{1}}_{1}}_{99}}_{\text{true}}$$

A notation for (simultaneous) substitution:

$$((x + y) \cdot z) \begin{bmatrix} x:=a+y \\ y:=b+z \\ z:=c+x \end{bmatrix} \quad = \quad ((a + y) + (b + z)) \cdot (c + x).$$

Note that $((a + b) \cdot (a - b)) \left[\begin{smallmatrix} a:=5 \\ b:=2 \end{smallmatrix} \right] = (5 + 2) \cdot (5 - 2)$.

Operations with substitution and copying

We know that $\Sigma_{i=2}^{5} i^3 = 2^3 + 3^3 + 4^3 + 5^3$.
If we introduce some intermediate steps we get:

$\Sigma_{i=2}^{5} i^3$
$\rightsquigarrow$ $\Sigma_{i \text{ in } (2,3,4,5)} i^3$
$\rightsquigarrow$ $(i^3)[i:=2] + (i^3)[i:=3] + (i^3)[i:=4] + (i^3)[i:=5]$
$\rightsquigarrow$ $2^3 + 3^3 + 4^3 + 5^3$

$\forall a \in \{2,3,5\}.\, a < 4$
$\rightsquigarrow$ $\forall a \text{ in } (2,3,5).\, a < 4$
$\rightsquigarrow$ $(a<4)[a:=2]\ \&\ (a<4)[a:=3]\ \&\ (a<4)[a:=5]$
$\rightsquigarrow$ $(2<4)\ \&\ (3<4)\ \&\ (5<4)$
$\rightsquigarrow$ false

$\forall a \in \{2,3,3,5\}.\, a < 4$
$\rightsquigarrow$ $\forall a \text{ in } (2,3,3,5).\, a < 4$
$\rightsquigarrow$ $(a<4)[a:=2]\ \&\ (a<4)[a:=3]\ \&\ (a<4)[a:=3]\ \&\ (a<4)[a:=5]$
$\rightsquigarrow$ $(2<4)\ \&\ (3<4)\ \&\ (3<4)\ \&\ (5<4)$
$\rightsquigarrow$ false

$\{\, a^3 \mid a \in \{2,3,5\} \,\}$
$\rightsquigarrow$ $\{(a^3)[a:=2], (a^3)[a:=3], (a^3)[a:=5]\}$
$\rightsquigarrow$ $\{2^3, 3^3, 5^3\}$

$\{\, (a, a^3) \mid a \in \{2,3,5\} \,\}$
$\rightsquigarrow$ $\{(a,a^3)[a:=2], (a,a^3)[a:=3], (a,a^3)[a:=5]\}$
$\rightsquigarrow$ $\{(2,2^3), (3,3^3), (5,5^3)\}$
$\rightsquigarrow$ $\{(2,8), (3,27), (5,125)\}$

One way to understand the 'λ' operator is using the idea — from Calculus 1 and Discrete Mathematics – that a function is a set of pairs (its "graph")...

$\lambda a{:}\{2,3,5\}.\, a^3$
$\rightsquigarrow$ $\{\, (a, a^3) \mid a \in \{2,3,5\} \,\}$
$\rightsquigarrow$ $\{(a,a^3)[a:=2], (a,a^3)[a:=3], (a,a^3)[a:=5]\}$
$\rightsquigarrow$ $\{(2,2^3), (3,3^3), (5,5^3)\}$
$\rightsquigarrow$ $\{(2,8), (3,27), (5,125)\}$

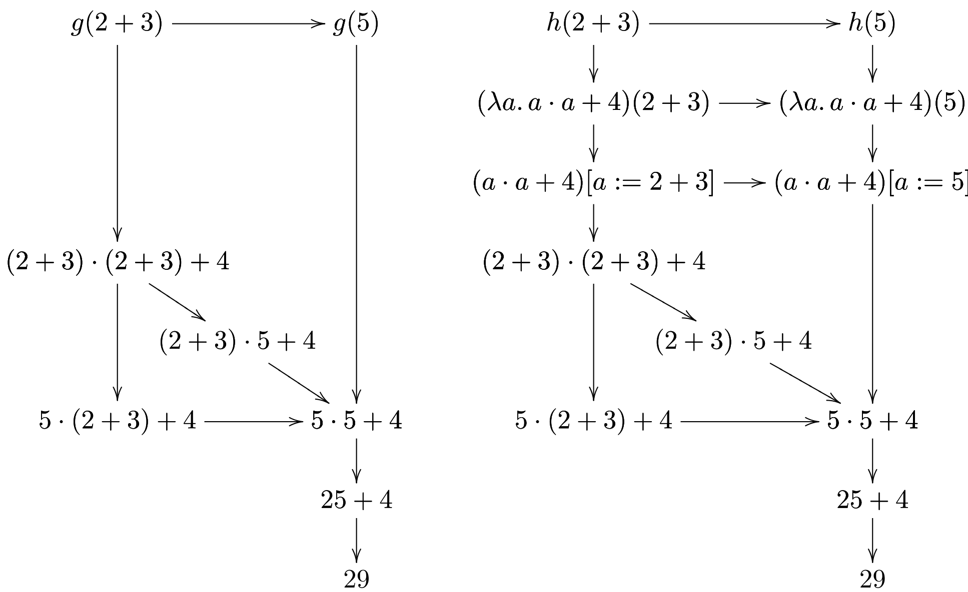
Note that

$(\lambda a{:}\{2,3,5\}.\, a^3)(5)$
$\rightsquigarrow$ $(\{(2,2^3), (3,3^3), (5,5^3)\})(5)$
$\rightsquigarrow$ 5^3
$\rightsquigarrow$ 125

$(\lambda a{:}\{2,3,5\}.\, a^3)(4)$
$\rightsquigarrow$ $(\{(2,2^3), (3,3^3), (5,5^3)\})(4)$
$\rightsquigarrow$ error

Lambda

A named function: $g(a) = a \cdot a + 4$
An unnamed function: $\lambda a.\, a \cdot a + 4$
Let $h = \lambda a.\, a \cdot a + 4$.
Then:



The usual notation for defining functions is like this:

$$\begin{array}{rccc} f: & \mathbb{N} & \to & \mathbb{R} \\ & n & \mapsto & 2 + \sqrt{n} \end{array}$$

$$\begin{array}{rccc} \text{(name)}: & \text{(domain)} & \to & \text{(codomain)} \\ & \text{(variable)} & \mapsto & \text{(expression)} \end{array}$$

It creates *named* functions
(with domains and codomains).

The usual notation for creating named functions
without specifying their domains and codomains
is just $f(n) = 2 + \sqrt{n}$.

Note that this is:

$$\begin{array}{cccc} f & (\, n \,) & = & 2 + \sqrt{n} \\ \text{(name)} & (\,\text{(variable)}\,) & = & \text{(expression)} \end{array}$$

Lambda notation: exercises

a) $(\lambda a.10a)(2+3)$
b) $(\lambda a.10a)((\lambda b.b+4)(3))$

Hint: use the speed you're most comfortable with. For example:

$$\underbrace{\underbrace{\underbrace{(\lambda a.10a)(\underbrace{\underbrace{\underbrace{(\lambda b.b+4)(3)}_{(b+4)[b:=3]}}_{3+4}}_{7})}_{(10a)[a:=7]}}_{10\cdot 7}}_{70}$$

$$\underbrace{\underbrace{(\lambda a.10a)(\underbrace{\underbrace{(\lambda b.b+4)(3)}_{3+4}}_{7})}_{10\cdot 7}}_{70}$$

$$\underbrace{(\lambda a.10a)(\underbrace{(\lambda b.b+4)(3)}_{7})}_{70}$$

c) $((\lambda a.(\lambda b.10a+b))(3))(4)$
d) $((\lambda f.(\lambda a.f(f(a))))(\lambda x.10x))(7)$

Functions as their graphs

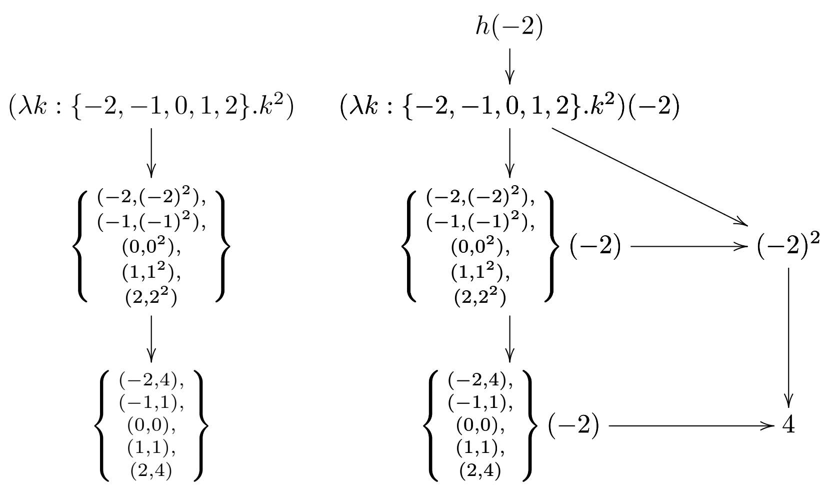
The *graph* of

$$\begin{array}{rccc} h: & \{-2,-1,0,1,2\} & \to & \{0,1,2,3,4\} \\ & k & \mapsto & k^2 \end{array}$$

is $\{(-2,4),(-1,1),(0,0),(1,1),(2,4)\}$.

We can think that a function *is* its graph,
and that a lambda-expression (with domain) reduces to a graph.
Then $h = \{(-2,4),(-1,1),(0,0),(1,1),(2,4)\}$
and $h(-2) = \{(-2,4),(-1,1),(0,0),(1,1),(2,4)\}(-2) = 4$.

Let $h := (\lambda k : \{-2,-1,0,1,2\}.k^2)$.
We have:



Note:
the graph of $(\lambda n : \mathbb{N}.n^2)$ has infinite points,
the graph of $(\lambda n : \mathbb{N}.n^2)$ is an infinite set,
the graph of $(\lambda n : \mathbb{N}.n^2)$ can't be written down *explicitly* without '...'s...

Mathematicians love infinite sets.
Computers hate infinite sets.
For mathematicians a function *is* its graph
(↑ remember Discrete Mathematics!)
For computer scientists a function *is* is a finite program.
Computer scientists love 'λ's!

I love things like this: $\left\{ \begin{smallmatrix} (3,30), \\ (4,40) \end{smallmatrix} \right\}(3) = 30$

Types (introduction)

Let:
$A = \{1, 2\}$
$B = \{30, 40\}$.

If $f : A \to B$, then f is one of these four functions:

$$\begin{smallmatrix}1\mapsto 30\\ 2\mapsto 30\end{smallmatrix},\ \begin{smallmatrix}1\mapsto 30\\ 2\mapsto 40\end{smallmatrix},\ \begin{smallmatrix}1\mapsto 40\\ 2\mapsto 30\end{smallmatrix},\ \begin{smallmatrix}1\mapsto 40\\ 2\mapsto 40\end{smallmatrix}$$

or, in other notation,

$$\left\{\begin{smallmatrix}(1,30)\\ (2,30)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,30)\\ (2,40)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,40)\\ (2,30)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,40)\\ (2,40)\end{smallmatrix}\right\}$$

which means that:

$$f \in \left\{ \left\{\begin{smallmatrix}(1,30)\\ (2,30)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,30)\\ (2,40)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,40)\\ (2,30)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,40)\\ (2,40)\end{smallmatrix}\right\} \right\}$$

Let's use the notation "$A \to B$" for "the set of all functions from A to B".

Then $(A \to B) = \left\{ \left\{\begin{smallmatrix}(1,30)\\ (2,30)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,30)\\ (2,40)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,40)\\ (2,30)\end{smallmatrix}\right\}, \left\{\begin{smallmatrix}(1,40)\\ (2,40)\end{smallmatrix}\right\} \right\}$
and $f : A \to B$
means $f \in (A \to B)$.

In Type Theory and λ-calculus "$a : A$"
is pronounced "a is of type A", and the meaning
of this is *roughly* "$a \in B$".
(We'll see the differences between '$\in$' and ':' (much) later).

Note that:
1. if $f : A \to B$ and $a : A$ then $f(a) : B$
2. if $a : A$ and $b : B$ then $(a, b) : A \times B$
3. if $p : A \times B$ then $\pi p : A$ and $\pi' p : B$, where
'π' means 'first projection' and
'π'' means 'second projection';
if $p = (2, 30)$ then $\pi p = 2$, $\pi' p = 30$.

If $p : A \times B$ and $g : B \to C$, then:

$$\underbrace{(\underbrace{\pi \underbrace{p}_{:A\times B}}_{:A}, \underbrace{\underbrace{g}_{:B\to C} \underbrace{(\pi' \underbrace{p}_{:A\times B}))}_{:B}}_{:C})}_{:A\times C}$$

Typed λ-calculus: trees

$A = \{1, 2\}$
$B = \{3, 4\}$
$C = \{30, 40\}$
$D = \{10, 20\}$
$A \times B = \left\{ \begin{matrix} (1,3), (1,4), \\ (2,3), (2,4) \end{matrix} \right\}$
$B \to C = \left\{ \left\{ \begin{smallmatrix} (3,30), \\ (4,30) \end{smallmatrix} \right\}, \left\{ \begin{smallmatrix} (3,30), \\ (4,40) \end{smallmatrix} \right\}, \left\{ \begin{smallmatrix} (3,40), \\ (4,30) \end{smallmatrix} \right\}, \left\{ \begin{smallmatrix} (3,40), \\ (4,40) \end{smallmatrix} \right\} \right\}$

If we know [the values of] a, b, f
then we know [the value of] $(a, f(b))$.
If $(a, b) = (2, 3)$ and $f = \left\{ \begin{smallmatrix} (3,30), \\ (4,40) \end{smallmatrix} \right\}$
then $(a, f(b)) = (2, 30)$.

$$\dfrac{\dfrac{(a,b)}{a}\,\pi \qquad \dfrac{\dfrac{(a,b)}{b}\,\pi' \qquad f}{f(b)}\,\mathsf{app}}{(a, f(b))}\,\mathsf{pair} \qquad\qquad \dfrac{\dfrac{(2,3)}{2}\,\pi \qquad \dfrac{\dfrac{(2,3)}{3}\,\pi' \qquad \{(3,30),(4,40)\}}{30}\,\mathsf{app}}{(2,30)}\,\mathsf{pair}$$

If we know the *types* of a, b, f
we know the type of $(a, f(b))$.
If we know the types of p, f
we know the type of $(\pi p, f(\pi' p))$.
If we know the types of p, f
we know the type of $(\lambda p : A \times B.(\pi p, f(\pi' p)))$.

$$\dfrac{\dfrac{(a,b) : A \times B}{a : A}\,\pi \qquad \dfrac{\dfrac{(a,b) : A \times B}{b : B}\,\pi' \qquad f : B \to C}{f(b) : C}\,\mathsf{app}}{(a, f(b)) : A \times C}\,\mathsf{pair}$$

$$\dfrac{\dfrac{\dfrac{p : A \times B}{\pi p : A}\,\pi \qquad \dfrac{\dfrac{p : A \times B}{\pi' p : B}\,\pi' \qquad f : B \to C}{f(\pi' p) : C}\,\mathsf{app}}{(\pi p, f(\pi' p)) : A \times C}\,\mathsf{pair}}{(\lambda p : A \times B.(\pi p, f(\pi' p))) : A \times B \to A \times C}\,\lambda$$

Types: exercises

Let:
$A = \{1, 2\}$
$B = \{3, 4\}$
$C = \{30, 40\}$
$D = \{10, 20\}$
$f = \left\{ \begin{smallmatrix} (3,30), \\ (4,40) \end{smallmatrix} \right\}$
$g = \left\{ \begin{smallmatrix} (1,10), \\ (2,20) \end{smallmatrix} \right\}$
Note that $f : B \to C$ and $g : A \to D$.

a) Evaluate $A \times B$.
b) Evaluate $A \to D$.
c) Evaluate $(\pi p, f(\pi' p))$ for each of the four possible values of $p : A \times B$.
d) Evaluate $\lambda p{:}A{\times}B.(\pi p, f(\pi' p))$.
e) Is this true?

$$(\lambda p{:}A{\times}B.(\pi p, f(\pi' p))) = \left\{ \begin{smallmatrix} ((1,3),(1,30)), \\ ((1,4),(1,40)), \\ ((2,3),(2,30)), \\ ((2,4),(2,40)) \end{smallmatrix} \right\}$$

f) Let $p = (2, 3)$. Evaluate $(g(\pi p), f(\pi' p))$.
g) Check that if $p : A \times B$ then $(g(\pi p), f(\pi' p)) : D \times C$.
h) Check that

$$(\lambda p{:}A{\times}B.(g(\pi p), f(\pi' p))) : A \times B \to D \times C.$$

i) Evaluate $(\lambda p{:}A{\times}B.(g(\pi p), f(\pi' p)))$.

Type inference

Here is another notation for checking types:

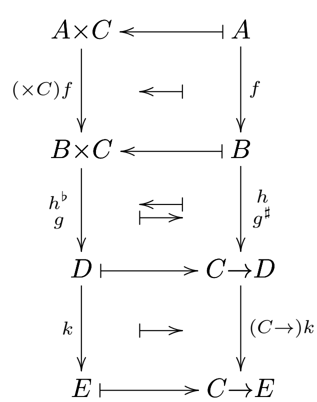
$$\underbrace{(\lambda \underbrace{p}_{:A\times B} : A\times B.\ \underbrace{(\underbrace{\pi \underbrace{p}_{:A\times B}}_{:A},\ \underbrace{\underbrace{f}_{:B\to C}\underbrace{(\pi' \underbrace{p}_{:A\times B})}_{:B}}_{:C})}_{:A\times C})}_{:A\times B\to A\times C}$$

Compare it with:

$$\dfrac{\dfrac{\dfrac{p : A\times B}{\pi p : A}\ \pi \qquad \dfrac{\dfrac{p : A\times B}{\pi' p : B}\ \pi' \qquad f : B\to C}{f(\pi' p) : C}\ \mathsf{app}}{(\pi p, f(\pi' p)) : A\times C}\ \mathsf{pair}}{(\lambda p : A\times B.(\pi p, f(\pi' p))) : A\times B\to A\times C}\ \lambda$$

Exercise:
Infer the type of each of the terms below (at the right of the ':=').
Use the two notations above.
The types of f, g, h, k are shown in the diagram below.

$a)$	$(\times C)f$	$:=$	$\lambda p{:}A\times C.(f(\pi p), \pi' p)$
$b)$	$h^\flat$	$:=$	$\lambda q{:}B\times C.(h(\pi q))(\pi' q)$
$c)$	$g^\sharp$	$:=$	$\lambda b{:}B.\lambda c{:}C.g(b,c)$
$d)$	$(C\to)k$	$:=$	$\lambda\varphi{:}C\to D.\lambda c{:}C.k(\varphi c)$



Term inference

Exercises:

$$\dfrac{\dfrac{\dfrac{\dfrac{p : A \times C}{: A}\,\pi \qquad f : A \to B}{: B}\,\mathsf{app} \qquad \dfrac{p : A \times C}{: C}\,\pi'}{: B \times C}\,\mathsf{pair}}{: A \times C \to B \times C}\,\lambda$$

$$\dfrac{\dfrac{\dfrac{q : B \times C}{: C}\,\pi' \qquad \dfrac{\dfrac{q : B \times C}{: B}\,\pi \qquad h : B \to (C \to D)}{: C \to D}\,\mathsf{app}}{: D}\,\mathsf{app}}{: B \times C \to D}\,\lambda$$

$$\dfrac{\dfrac{\dfrac{\dfrac{b : B \quad c : C}{: B \times C}\,\mathsf{pair} \qquad g : B \times C \to D}{: D}\,\mathsf{app}}{: C \to D}\,\lambda}{: B \to (C \to D)}\,\lambda$$

$$\dfrac{\dfrac{\dfrac{\dfrac{c : C \quad \varphi : C \to D}{: D}\,\mathsf{app} \qquad k : D \to E}{: E}\,\mathsf{app}}{: (C \to E)}\,\lambda}{: (C \to D) \to (C \to E)}\,\lambda$$

Term inference: answers

$$\dfrac{\dfrac{\dfrac{\dfrac{p : A \times C}{\pi p : A}\,\pi \qquad f : A \to B}{f(\pi p) : B}\,\mathsf{app} \qquad \dfrac{p : A \times C}{\pi' p : C}\,\pi'}{(f(\pi p), \pi' p) : B \times C}\,\mathsf{pair}}{\lambda p{:}A{\times}C.(f(\pi p), \pi' p) : A \times C \to B \times C}\,\lambda$$

$$\dfrac{\dfrac{\dfrac{q : B \times C}{\pi' q : C}\,\pi' \qquad \dfrac{\dfrac{q : B \times C}{\pi q : B}\,\pi \qquad h : B \to (C \to D)}{h(\pi q) : C \to D}\,\mathsf{app}}{h(\pi q)(\pi' q) : D}\,\mathsf{app}}{\lambda q{:}B{\times}C.h(\pi q)(\pi' q) : B \times C \to D}\,\lambda$$

$$\dfrac{\dfrac{\dfrac{\dfrac{b : B \qquad c : C}{(b, c) : B \times C}\,\mathsf{pair} \qquad g : B \times C \to D}{g(b, c) : D}\,\mathsf{app}}{\lambda c{:}C.g(b, c) : C \to D}\,\lambda}{\lambda b{:}B.\lambda c{:}C.g(b, c) : B \to (C \to D)}\,\lambda$$

$$\dfrac{\dfrac{\dfrac{\dfrac{c : C \qquad \varphi : C \to D}{\varphi c : D}\,\mathsf{app} \qquad k : D \to E}{k(\varphi c) : E}\,\mathsf{app}}{\lambda c{:}C.k(\varphi c) : (C \to E)}\,\lambda}{\varphi{:}C{\to}D.\lambda c{:}C.k(\varphi c) : (C \to D) \to (C \to E)}\,\lambda$$

Contexts and '$\vdash$'

Suppose that A, B, C are known, and are sets.
(Jargon: "fix sets A, B, C".)
Then this

$$\underbrace{p:A\times B, f:B\to C}_{\substack{\text{"context": a series of}\\ \text{declarations like}\\ var{:}type}} \vdash \underbrace{f(\pi'p):C}_{term{:}type}$$

Means:
"In this context the expression *expr* makes sense, is not $\textsf{error}$, and its result is of type *type*."

Note that calculating $f(\pi'p)$ yields $\textsf{error}$ if we do not know the values of f or p.

What happens if we add contexts to each $term : type$ in a tree?
The two bottom nodes in

$$\dfrac{\dfrac{\dfrac{p:A\times B}{\pi p:A}\,\pi \qquad \dfrac{\dfrac{p:A\times B}{\pi'p:B}\,\pi' \qquad f:B\to C}{f(\pi'p):C}\,\textsf{app}}{(\pi p, f(\pi'p)):A\times C}\,\textsf{pair}}{(\lambda p:A\times B.(\pi p,f(\pi'p))):A\times B\to A\times C}\,\lambda$$

would become:

$$f:B\to C, p:A\times B \vdash (\pi p, f(\pi'p)):A\times C$$
$$f:B\to C \vdash (\lambda p:A\times B.(\pi p, f(\pi'p))):A\times B\to A\times C$$

After the rule 'λ' the 'p' is no longer needed!

If we add the contexts and omit the types, the tree becomes:

$$\dfrac{\dfrac{\dfrac{p\vdash p}{p\vdash \pi p}\,\pi \qquad \dfrac{\dfrac{p\vdash p}{p\vdash \pi'p}\,\pi' \qquad f\vdash f}{f,p\vdash f(\pi'p)}\,\textsf{app}}{f,p\vdash(\pi p, f(\pi'p))}\,\textsf{pair}}{f\vdash(\lambda p{:}A{\times}B.(\pi p,f(\pi'p)))}\,\lambda \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{[p]^1}{\pi p}\,\pi \qquad \dfrac{\dfrac{[p]^1}{\pi'p}\,\pi' \qquad f}{f(\pi'p)}\,\textsf{app}}{(\pi p, f(\pi'p))}\,\textsf{pair}}{(\lambda p{:}A{\times}B.(\pi p,f(\pi'p)))}\,\lambda;1$$

Notational trick:
below the bar '$\lambda;1$' the value of p is no longer needed;
we say that the p is "discharged" (from the list of hypotheses)
and we mark the 'p' on the leaves of the tree with '$[\cdot]^1$';
a '$[\cdot]^1$' on a hypothesis means: "below the bar '$\lambda;1$' I am no longer a hypothesis".

Curry-Howard: introduction

We are learning a system called
"the simply-typed λ-calculus (with binary products)" —
system $\lambda 1$, for short.

In $\lambda 1$ in its fullest form,
its objects are trees of '$\ldots \vdash$ *term* : *type*'s,
but we saw (evidence) that we can:
• reconstruct the full tree from just the '*term* : *type*'s,
• write just ': *type*'s (except on the leaves, to get the var names),
• reconstruct the full tree from just the bottom '*term* : *type*'...

For example, we can reconstruct the whole tree,
with contexts, from:

$$\dfrac{\dfrac{\dfrac{[p:A{\times}B]^1}{:A}\,\pi \qquad \dfrac{\dfrac{[p:A{\times}B]^1}{:B}\,\pi' \qquad f:B\to C}{:C}\,\mathsf{app}}{:A{\times}C}\,\mathsf{pair}}{:A{\times}B\to A{\times}C}\,\lambda$$

If we erase the terms and the ':'s and leave only the types,
we get something that is strikingly similar to a tree in
Natural Deduction,

$$\dfrac{\dfrac{\dfrac{[A{\times}B]^1}{A}\,\pi \qquad \dfrac{\dfrac{[A{\times}B]^1}{B}\,\pi' \qquad B\to C}{C}\,\mathsf{app}}{A{\times}C}\,\mathsf{pair}}{A{\times}B\to A{\times}C}\,\lambda$$

$$\rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{[P\&Q]^1}{P}\,\&E_1 \qquad \dfrac{\dfrac{[P\&Q]^1}{Q}\,\&E_2 \qquad Q{\to}R}{R}\,{\to}E}{P\&Q}\,\&I}{P\&R\to P\&Q}\,{\to}I;1$$

which talks about *logic*.

Curry-Howard: Natural Deduction

The tree

$$\cfrac{\cfrac{\cfrac{[P\&Q]^1}{P}\;\&E_1 \qquad \cfrac{\cfrac{[P\&Q]^1}{Q}\;\&E_2 \qquad Q{\to}R}{R}\;\to E}{P\&Q}\;\&I}{P\&R \to P\&Q}\;\to I;1$$

is in $\mathrm{ND}_{\&\to}$ (or in $\mathrm{IPL}_{\&\to}$), the fragment of
Natural Deduction (or intuitionistic predicate logic)
that only has the connectives & and $\to$.

Its rules are:

$$\frac{P \quad Q}{P\&Q}\;\&I \qquad \frac{P\&Q}{P}\;\&E_1 \qquad \frac{P\&Q}{Q}\;\&E_2$$

$$\frac{\begin{array}{cc} P & [Q]^1 \\ & \vdots \\ & R \end{array}}{Q \to R}\;\to I \qquad \frac{P \quad P \to Q}{Q}\;\to E$$

New rules (for $\top$, $\bot$, $\vee$):
(not yet — see the whiteboard for 20170418)

Planar Heyting Algebras

We read sections 1–7 of:
http://angg.twu.net/LATEX/2017planar-has.pdf

Let $B = \begin{array}{ccccc} & 32 & & & \\ & & 22 & & \\ & 21 & & 12 & \\ 20 & & 11 & & 02 \\ & 10 & & 01 & \\ & & 00 & & \end{array}$.

Exercises:
Calculate and represent in positional notation when possible:
a) $\lambda lr{:}B.l$
b) $\lambda lr{:}B.r$
c) $\lambda lr{:}B.(l \leq 1)$
d) $\lambda lr{:}B.(r \geq 1)$
e) $\lambda lr{:}B.lr \leq 11$
f) $\lambda lr{:}B.lr\&12$
g) $\lambda lr{:}B.\ \mathsf{valid}\ (\langle l+1, r\rangle)$
h) $\lambda lr{:}B.lr\ \mathsf{leftof}\ 11$
i) $\lambda lr{:}B.lr\ \mathsf{leftof}\ 12$
j) $\lambda lr{:}B.lr\ \mathsf{above}\ 11$
k) $\lambda lr{:}B.\ \mathsf{ne}\ (lr)$
l) $\lambda lr{:}B.\ \mathsf{nw}\ (lr)$
m) $20 \to 11$
n) $02 \to 11$
o) $22 \to 11$
p) $00 \to 11$
q) $\lambda lr{:}B.\neg lr$
r) $\lambda lr{:}B.\neg\neg lr$
s) $\lambda lr{:}B.lr = \neg\neg lr$

Algebraic structures

A *ring* is a 6-uple

$$(R, 0_R, 1_R, +_R, -_R, \cdot_R)$$

where $R, 0_R, \ldots, \cdot_R$ have the following types,

R is a set,
$0_R \in R$,
$1_R \in R$,
$+_R : R \times R \to R$,
$-_R : R \to R$ (unary minus),
$\cdot_R : R \to R$,

and where the components obey these equations ($\forall a, b, c \in R$):

$a + 0_R = 0_R + a = a, \quad a + b = b + a, \quad a + (b + c) = (a + b) + c, \quad a + (-a) = 0,$
$a \cdot 1_R = 1_R \cdot a = a, \quad a \cdot b = b \cdot a, \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c,$
$a \cdot (b + c) = a \cdot b + a \cdot c.$

A *proto-ring* is a 6-uple $(R, 0_R, 1_R, +_R, -_R, \cdot_R)$
that obeys the typing conditions of a ring.
A *ring* is a proto-ring plus the assurance that it obeys the ring equations.

A *proto-Heyting Algebra* is a 7-uple

$$H = (\Omega, \leq_H, \top_H, \bot_H, \&_H, \vee_H, \to_H)$$

in which:

Ω is a set (the "set of truth values"),
$\leq_H \subset \Omega \times \Omega$ (partial order),
$\top_H \in \Omega$,
$\bot_H \in \Omega$,
$\&_H : \Omega \times \Omega \to \Omega$
$\vee_H : \Omega \times \Omega \to \Omega$
$\to_H : \Omega \times \Omega \to \Omega$

Sometimes we add operations '$\neg$' and $\leftrightarrow$ to a (proto-)HA H,

$$H = (\Omega, \leq_H, \top_H, \bot_H, \&_H, \vee_H, \to_H, \neg_H, \leftrightarrow_H)$$

by *defining them* as $\neg P := P \to \bot$ and $P \leftrightarrow Q := (P \to Q) \& (Q \to P)$
(i.e., $\neg_H P := P \to_H \bot_H$
and $P \leftrightarrow_H Q := (P \to_H Q) \&_H (Q \to_H P)$).

This abuse of language is very common:
R "=" $(R, 0_R, 1_R, +_R, -_R, \cdot_R)$.

Protocategories

A *protocategory* is a 4-uple

$$\mathbf{C} = (\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}})$$

where

$\mathbf{C}_0$ is a set (more precisely a “class”),
$\mathrm{Hom}_{\mathbf{C}} : \mathbf{C}_0 \times \mathbf{C}_0 \to \mathbf{Sets}$,
$\mathrm{id}_{\mathbf{C}}(A) \in \mathrm{Hom}_{\mathbf{C}}(A, A)$,
$(\circ_{\mathbf{C}})_{ABC} : \mathrm{Hom}_{\mathbf{C}}(B, C) \times \mathrm{Hom}_{\mathbf{C}}(A, B) \to \mathrm{Hom}_{\mathbf{C}}(A, C)$.

A *categoru* is a protocategory plus the assurance that identities behave as expected and composition is associative.

Sometimes we add an operation ‘;’ to a category,

$$\mathbf{C} = (\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}}, ;_{\mathbf{C}})$$

where ‘;’ is the composition in other order: $f \circ g = g; f$.