University of Cambridge 2017 MPhil ACS / CST Part III *Category Theory and Logic* (L108)

Brief Notes on the Category Theoretic Semantics of Simply Typed Lambda Calculus

Andrew Pitts

Notation: comma-separated snoc lists

When presenting logical systems and type theories, it is common to write finite lists of things using a comma to indicate the cons-operation and with the head of the list at the right. With this convention there is no common notation for the empty list; we will use the symbol "$\diamond$". Thus ML-style list notation

$$\text{nil} \qquad a :: \text{nil} \qquad b :: a :: \text{nil} \qquad \textit{etc}$$

becomes

$$\diamond \qquad \diamond, a \qquad \diamond, a, b \qquad \textit{etc}$$

For non-empty lists, it is very common to leave the initial part "$\diamond$," of the above notation implicit, for example just writing a, b instead of $\diamond, a, b$.

Write $\boxed{X^*}$ for the set of such finite lists with elements from the set X.

1 Syntax of the simply typed λ-calculus

Fix a countably infinite set $\boxed{\mathbb{V}}$ whose elements are called **variables** and are typically written $x, y, z, \ldots$

The **simple types** (with product types) A over a set Gnd of **ground types** are given by the following grammar, where G ranges over Gnd:

$$A ::= G \mid \texttt{unit} \mid A \times A \mid A \to A$$

Write $\boxed{ST(Gnd)}$ for the set of simple types over Gnd.

The syntax trees t of the **simply typed λ-calculus** (STLC) over Gnd with **constants** drawn from a set Con are given by the following grammar, where $\texttt{c}$ ranges over Con, x over $\mathbb{V}$ and A over $ST(Gnd)$:

$$t ::= \texttt{c} \mid x \mid \texttt{()} \mid \texttt{(}t\,\texttt{,}\,t\texttt{)} \mid \texttt{fst}\,t \mid \texttt{snd}\,t \mid \lambda x : A.\, t \mid t\, t$$

We identify such syntax trees modulo remaning of λ-bound variables. More formally a **simply typed λ-term** is an equivalence class of syntax trees for the following, inductively defined relation of **α-equivalence** $\boxed{=_\alpha}$

$$\frac{}{\mathtt{c} =_\alpha \mathtt{c}} \qquad \frac{}{x =_\alpha x} \qquad \frac{}{() =_\alpha ()} \qquad \frac{t_1 =_\alpha t_1' \qquad t_2 =_\alpha t_2'}{(t_1\ ,\ t_2) =_\alpha (t_1'\ ,\ t_2')} \qquad \frac{t =_\alpha t'}{\mathtt{fst}\, t =_\alpha \mathtt{fst}\, t'}$$

$$\frac{t =_\alpha t'}{\mathtt{snd}\, t =_\alpha \mathtt{snd}\, t'} \qquad \frac{t_1 =_\alpha t_1' \qquad t_2 =_\alpha t_2'}{t_1\, t_2 =_\alpha t_1'\, t_2'}$$

$$\frac{(y\ x) \cdot t =_\alpha (y\ x') \cdot t' \qquad y \text{ does not occur in } \{x, x', t, t'\}}{\lambda x : A.\, t =_\alpha \lambda x' : A.\, t'}$$

In the last rule $(y\ x) \cdot t$ indicates the syntax tree obtained from t by swapping occurrences of y and x; given the condition that y does not occur in t, this is the same as replacing all occurrences of x in t by y. Thus the last rule says that $\lambda x : A.\, t$ and $\lambda x' : A.\, t'$ are α-equivalent if t and t' become α-equivalent once we replace all occurrences of x in t and all occurrences of x' in t' by some common "fresh" variable y.

It is conventional to not make a notational distinction between a tree t and the α-equivalence class that it determines. That convention can be made mathematically precise via the use of nominal sets; see for example Pitts [2013, Chapter 8]. An alternative to working with λ-terms as α-equivalence classes of abstract syntax trees is to use a nameless representation due to de Bruijn [1972] instead of explicitly named bound variables. For typed λ-calculi, especially when using systems like Agda [`wiki.portal.chalmers.se/agda/agda.php`] or Coq [`coq.inria.fr`], so-called *well-scoped de Bruijn indices* are very convenient (if not very humam-readable); see for example Keller and Altenkirch [2010, Section 2].

2 Typing relation

We assume that the set *Con* comes with a function mapping each constant $\mathtt{c} \in Con$ to its type $A \in ST(Gnd)$. We some times write $\mathtt{c}$ as $\mathtt{c}^A$ to indicate that A is its type.

In order to extend this typing function from constants to compound simply typed λ-terms we have to assign types to (free) variables. We do so via **typing environments** Γ:

$$\Gamma ::= \diamond \mid \Gamma, x : A \qquad (\text{where } x \in \mathbb{V}, A \in ST(Gnd))$$

Thus the set of typing environments is in bijection with $(\mathbb{V} \times ST(Gnd))^*$, the set of finite lists of (variable,type)-pairs. The domain $\boxed{\operatorname{dom}\Gamma}$ of a typing environment Γ is the finite set of variables occurring in it:

$$\begin{aligned} \operatorname{dom} \diamond &= \emptyset \\ \operatorname{dom}(\Gamma, x : A) &= \operatorname{dom}\Gamma \cup \{x\} \end{aligned}$$

We only use the Γ that are well-formed $\boxed{\Gamma \text{ ok}}$ in the sense that no variable occurs more than once in the list:

$$\frac{}{\diamond \text{ ok}} \qquad \frac{\Gamma \text{ ok} \qquad x \notin \operatorname{dom}\Gamma}{\Gamma, x : A \text{ ok}}$$

Then the **typing relation** $\boxed{\Gamma \vdash t : A}$ for assigning types A to terms t in a given typing environment Γ is inductively defined by:

$$\frac{\Gamma\ \mathrm{ok} \qquad x \notin \mathrm{dom}\,\Gamma}{\Gamma, x : A \vdash x : A}\ (\textsc{var}) \qquad \frac{\Gamma \vdash x : A \qquad x' \notin \mathrm{dom}\,\Gamma}{\Gamma, x' : A' \vdash x : A}\ (\textsc{var}')$$

$$\frac{\Gamma\ \mathrm{ok}}{\Gamma \vdash \mathtt{c}^A : A}\ (\textsc{const}) \qquad \frac{\Gamma\ \mathrm{ok}}{\Gamma \vdash () : \mathtt{unit}}\ (\textsc{unit})$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash t' : A'}{\Gamma \vdash (t\ ,\ t') : A \times A'}\ (\textsc{pair}) \qquad \frac{\Gamma \vdash t : A \times A'}{\Gamma \vdash \mathtt{fst}\,t : A}\ (\textsc{fst}) \qquad \frac{\Gamma \vdash t : A \times A'}{\Gamma \vdash \mathtt{snd}\,t : A'}\ (\textsc{snd})$$

$$\frac{\Gamma, x : A \vdash t : A'}{\Gamma \vdash \lambda x : A.\, t : A \rightarrow A'}\ (\lambda) \qquad \frac{\Gamma \vdash t : A \rightarrow A' \qquad \Gamma \vdash t' : A}{\Gamma \vdash t\, t' : A'}\ (\textsc{app})$$

Here are some simple properties of the typing relation $\Gamma \vdash t : A$, proved by induction on its derivation. The second property makes use of the finite set $\boxed{\mathrm{fv}\, t}$ of **free variables** of a term t, which is well-defined by:

$$\begin{aligned} &\mathrm{fv}\,\mathtt{c} = \mathrm{fv}\,() = \emptyset && \mathrm{fv}\,(t\ ,\ t') = \mathrm{fv}\,t\,t' = \mathrm{fv}\,t \cup \mathrm{fv}\,t' \\ &\mathrm{fv}\,x = \{x\} && \mathrm{fv}\,\lambda x : A.\,t = \{x' \in \mathrm{fv}\,t \mid x' \neq x\} \end{aligned}$$

Lemma 2.1. 1. *If* $\Gamma \vdash t : A$*, then* Γ ok.

2. *If* $\Gamma \vdash t : A$*, then* $\mathrm{fv}\,t \subseteq \mathrm{dom}\,\Gamma$.

3. *If* $\Gamma \vdash t : A$ *and* $\Gamma \vdash t : A'$*, then* $A = A'$.

□

Property 3 says that terms have at most one type in any (well-formed) typing environment. Of course some terms have no type; for example $\diamond \vdash ()\,() : A$ is not derivable from the rules for any type A (why?).

Because we have formulated typing environments as ordered lists (rather than, say, finite maps from variables to types), the important property of the typing relation that it is preserved under **weakening** typing environments (that is, adding extra (variable, type)-pairs while preserving the property of being well-formed) has to be formulated carefully. Here is a particular inductive definition of a **weakening relation** $\boxed{w : \Gamma' \rhd \Gamma}$ (where $w ::= \iota \mid w\,\pi \mid w\,\mathsf{x}$), inspired by Chapman [2009, Section 4.5], that interacts well with the typing relation:

$$\frac{\Gamma\ \mathrm{ok}}{\iota : \Gamma \rhd \Gamma} \qquad \frac{w : \Gamma' \rhd \Gamma \qquad x \notin \mathrm{dom}\,\Gamma'}{w\,\pi : (\Gamma', x : A) \rhd \Gamma} \qquad \frac{w : \Gamma' \rhd \Gamma \qquad x \notin \mathrm{dom}\,\Gamma'}{w\,\mathsf{x} : (\Gamma', x : A) \rhd \Gamma, x : A}$$

Lemma 2.2. 1. *If* $w : \Gamma' \rhd \Gamma$ *and* Γ ok*, then* Γ' ok.

2. *If* $\Gamma \vdash t : A$ *and* $w : \Gamma' \rhd \Gamma$*, then* $\Gamma' \vdash t : A$.

Proof. Property 1 is proved by induction on the derivation of $w : \Gamma' \triangleright \Gamma$.

For property 2, which is the desired weakening property of the typing relation, one proceeds by induction on the derivation of $\Gamma \vdash t : A$. For the base case when t is a variable, one proves

$$\Gamma \vdash x : A \quad \text{and} \quad w : \Gamma' \triangleright \Gamma \quad \text{implies} \quad \Gamma' \vdash x : A$$

by induction on the derivation of $w : \Gamma' \triangleright \Gamma$, using part 1; for the induction step when t is a λ-abstraction one uses the fact that λ-terms are α-equivalence classes of syntax trees, so that a representative λ-bound variable can chosen to not be in $\operatorname{dom}\Gamma'$, allowing the third rule for the $w : \Gamma' \triangleright \Gamma$ relation to be applied. □

3 Cartesian closed categories

Recall that a category $\mathbf{C}$ is **cartesian closed** if it has

A terminal object: a $\mathbf{C}$-object $\top$ with the property that for every $Z \in \operatorname{obj}\mathbf{C}$ there is a unique morphism $\langle\rangle \in \mathbf{C}(Z, \top)$. The uniqueness part of this property is:

$$f \in \mathbf{C}(Z, \top) \Rightarrow f = \langle\rangle$$

Binary products: for all $X, Y \in \operatorname{obj}\mathbf{C}$ there is a $\mathbf{C}$-object $X \times Y$ and morphisms $\pi_1 \in \mathbf{C}(X \times Y, X)$, $\pi_2 \in \mathbf{C}(X \times Y, Y)$ with the property that for every $Z \in \operatorname{obj}\mathbf{C}$, $f \in \mathbf{C}(Z, X)$ and $g \in \mathbf{C}(Z, Y)$, there is a unique morphism $\langle f, g\rangle \in \mathbf{C}(Z, X \times Y)$ satisfying $\pi_1 \circ \langle f, g\rangle = f$ and $\pi_2 \circ \langle f, g\rangle = g$. The uniqueness part of this property is equivalent to requiring:

$$h \in \mathbf{C}(Z, X \times Y) \Rightarrow h = \langle \pi_1 \circ h, \pi_2 \circ h\rangle$$

As a matter of notation, if $f \in \mathbf{C}(Z, X)$ and $g \in \mathbf{C}(W, Y)$ we define $f \times g \in \mathbf{C}(Z \times W, X \times Y)$ to be $f \times g \triangleq \langle f \circ \pi_1, g \circ \pi_2\rangle$.

Exponentials: for all $X, Y \in \operatorname{obj}\mathbf{C}$ there is a $\mathbf{C}$-object Y^X and a morphism $\mathrm{app} \in \mathbf{C}(Y^X \times X, Y)$ with the property that for every $Z \in \operatorname{obj}\mathbf{C}$ and $f \in \mathbf{C}(Z \times X, Y)$ there is a unique morphism $\mathrm{cur}\, f \in \mathbf{C}(Z, Y^X)$ satisfying $\mathrm{app} \circ (\mathrm{cur}\, f \times \mathrm{id}_X) = f$. The uniqueness part of this property is equivalent to requiring:

$$h \in \mathbf{C}(Z, Y^X) \Rightarrow h = \mathrm{cur}(\mathrm{app} \circ (h \times \mathrm{id}_X))$$

4 Semantics in a cartesian closed category

Let $\mathbf{C}$ be a cartesian closed category. Any function $M : \mathit{Gnd} \to \operatorname{obj}\mathbf{C}$ assigning $\mathbf{C}$-objects to ground types can be extended to a function mapping types $A \in \mathit{ST}(\mathit{Gnd})$ to objects

$M[\![A]\!] \in \mathrm{obj}\,\mathbf{C}$, by recursion over the structure of A:

$$
\begin{aligned}
M[\![G]\!] &= M(G) \\
M[\![\texttt{unit}]\!] &= 1 && \text{(terminal object in } \mathbf{C}\text{)} \\
M[\![A \times A']\!] &= M[\![A]\!] \times M[\![A']\!] && \text{(product in } \mathbf{C}\text{)} \\
M[\![A \to A']\!] &= M[\![A']\!]^{M[\![A]\!]} && \text{(exponential in } \mathbf{C}\text{)}
\end{aligned}
$$

Typing environments also denote $\mathbf{C}$-objects, by recursion over the length of the list Γ:

$$
\begin{aligned}
M[\![\diamond]\!] &= 1 \\
M[\![\Gamma, x : A]\!] &= M[\![\Gamma]\!] \times M[\![A]\!]
\end{aligned}
$$

Finally, if in addition to $M : \mathit{Gnd} \to \mathrm{obj}\,\mathbf{C}$ we also have a function assigning to each constant $\texttt{c} \in \mathit{Con}$, of type A say, a global section[1] $M(c) \in \mathbf{C}(1, M[\![A]\!])$, then for each derivable instance of the typing relation $\Gamma \vdash t : A$ we define a $\mathbf{C}$-morphism

$$\boxed{M[\![\Gamma \vdash t : A]\!] \in \mathbf{C}(M[\![\Gamma]\!], M[\![A]\!])}$$

as follows:

$$
\begin{aligned}
M[\![\Gamma, x : A \vdash x : A]\!] &= M[\![\Gamma]\!] \times M[\![A]\!] \xrightarrow{\pi_2} M[\![A]\!] \\
M[\![\Gamma, x' : A' \vdash x : A]\!] &= M[\![\Gamma]\!] \times M[\![A']\!] \xrightarrow{\pi_1} M[\![\Gamma]\!] \xrightarrow{M[\![\Gamma \vdash x : A]\!]} M[\![A]\!] \quad \text{if } x' \notin \mathrm{dom}\,\Gamma \\
M[\![\Gamma \vdash c^A : A]\!] &= M[\![\Gamma]\!] \xrightarrow{\langle\rangle} 1 \xrightarrow{M(c)} M[\![A]\!] \\
M[\![\Gamma \vdash \texttt{()} : \texttt{unit}]\!] &= M[\![\Gamma]\!] \xrightarrow{\langle\rangle} 1 \\
M[\![\Gamma \vdash \texttt{(}t\texttt{ , }t'\texttt{)} : A \times A']\!] &= M[\![\Gamma]\!] \xrightarrow{\langle M[\![\Gamma \vdash t : A]\!], M[\![\Gamma \vdash t' : A']\!]\rangle} M[\![A]\!] \times M[\![A']\!] \\
M[\![\Gamma \vdash \texttt{fst}\, t : A]\!] &= M[\![\Gamma]\!] \xrightarrow{M[\![\Gamma \vdash t : A \times A']\!]} M[\![A]\!] \times M[\![A']\!] \xrightarrow{\pi_1} M[\![A]\!] \\
&\quad \text{where } A' \text{ is the unique type for which } \Gamma \vdash t : A \times A' \text{ holds} \\
M[\![\Gamma \vdash \texttt{snd}\, t : A']\!] &= M[\![\Gamma]\!] \xrightarrow{M[\![\Gamma \vdash t : A \times A']\!]} M[\![A]\!] \times M[\![A']\!] \xrightarrow{\pi_2} M[\![A']\!] \\
&\quad \text{where } A \text{ is the unique type for which } \Gamma \vdash t : A \times A' \text{ holds} \\
M[\![\Gamma \vdash \lambda x : A.\, t : A \to A']\!] &= \mathrm{cur}\left(M[\![\Gamma]\!] \times M[\![A]\!] \xrightarrow{M[\![\Gamma, x : A \vdash t : A']\!]} M[\![A']\!] \right) \\
M[\![\Gamma t\, t' : A']\!] &= M[\![\Gamma]\!] \xrightarrow{\langle f, f' \rangle} M[\![A']\!]^{M[\![A]\!]} \times M[\![A]\!] \xrightarrow{\mathrm{app}} M[\![A']\!] \\
&\quad \text{where } A \text{ is the unique type for which } \Gamma \vdash t : A \to A' \text{ holds} \\
&\quad \text{and where } f = M[\![\Gamma \vdash t : A \to A']\!] \text{ and } f' = M[\![\Gamma \vdash t' : A]\!].
\end{aligned}
$$

Summary: *given an interpretation of ground types as objects of* $\mathbf{C}$ *and constants as global sections of objects in* $\mathbf{C}$*, we give meaning to simple types as* $\mathbf{C}$*-objects and meaning to simply-typed* λ *terms (in a given typing environment) as* $\mathbf{C}$*-morphisms.*

We will need the following property of this semantics with respect to weakening typing environments:

[1] In a category $\mathbf{C}$ with terminal object 1, morphisms $f \in \mathbf{C}(1, X)$ are called **global sections** of the $\mathbf{C}$-object X.

Lemma 4.1 (Semantics of weakening)**.** *For each instance of the weakening relation $w : \Gamma' \rhd \Gamma$ we get a* **C***-morphism*

$$M[\![w : \Gamma' \rhd \Gamma]\!] : M[\![\Gamma']\!] \to M[\![\Gamma]\!]$$

by defining:

$$\begin{aligned} M[\![\iota : \Gamma \rhd \Gamma]\!] &= M[\![\Gamma]\!] \xrightarrow{\mathrm{id}} M[\![\Gamma]\!] \\ M[\![w\,\pi : (\Gamma', x : A) \rhd \Gamma]\!] &= M[\![\Gamma']\!] \times M[\![A]\!] \xrightarrow{\pi_1} M[\![\Gamma']\!] \xrightarrow{M[\![w:\Gamma' \rhd \Gamma]\!]} M[\![\Gamma]\!] \\ M[\![w\,x : (\Gamma', x : A) \rhd \Gamma, x : A]\!] &= M[\![\Gamma']\!] \times M[\![A]\!] \xrightarrow{M[\![w:\Gamma' \rhd \Gamma]\!] \times \mathrm{id}} M[\![\Gamma]\!] \times M[\![A]\!] \end{aligned}$$

If $w : \Gamma' \rhd \Gamma$ holds, then for all derivable $\Gamma \vdash t : A$, the meaning of $\Gamma' \vdash t : A$ (valid by Lemma 2.2(2)) in **C** *is the morphism $M[\![\Gamma']\!] \to M[\![A]\!]$ equal to the morphism given by composing $M[\![w : \Gamma' \rhd \Gamma]\!]$ with $M[\![\Gamma \vdash t : A]\!]$.*

Proof. By induction on the derivation of $\Gamma \vdash t : A$, following the proof of Lemma 2.2(2). For the induction step for λ-abstractions, one uses the fact that in a cartesian closed category the Currying operation satisfies $\mathrm{cur}(f \circ (g \times \mathrm{id})) = (\mathrm{cur}\, f) \circ g$. $\square$

When M is understood from the context one sometimes just writes $[\![A]\!]$ for $M[\![A]\!]$ and similarly for $[\![\Gamma]\!]$ and $[\![\Gamma \vdash t : A]\!]$. Also, since the type A in $\Gamma \vdash t : A$ is uniquely determined (Lemma 2.1(3)), it is common to just write $[\![\Gamma \vdash t]\!]$ for $[\![\Gamma \vdash t : A]\!]$.

If $\Gamma \vdash t : A$ and $\Gamma \vdash t' : A$, then a typed equation

$$\Gamma \vdash t = t' : A$$

is satisfied by this semantics if $M[\![\Gamma \vdash t : A]\!]$ and $M[\![\Gamma \vdash t' : A]\!]$ are equal morphisms from $M[\![\Gamma]\!]$ to $M[\![A]\!]$ in **C**. It is natural to ask which typed equations are always satisfied, whatever the ccc **C**. This turns out to to be the notion of $\beta\eta$-equality given in Section 6. To describe it we first have to define (capture-avoiding) substitution of terms for free variables and its semantics.

5 Substitution

Substitutions σ are finite lists of (variable, term)-pairs,written with the following notation:

$$\sigma ::= \diamond \mid \sigma, x := t$$

The domain $\boxed{\mathrm{dom}\,\sigma}$ of a substitution is given by

$$\begin{aligned} \mathrm{dom}\,\diamond &= \emptyset \\ \mathrm{dom}(\sigma, x := t) &= \mathrm{dom}\,\sigma \cup \{x\} \end{aligned}$$

and its set of free variables $\boxed{\mathrm{fv}\,\sigma}$ by

$$\begin{aligned} \mathrm{fv}\,\diamond &= \emptyset \\ \mathrm{fv}(\sigma, x := t) &= \mathrm{fv}\,\sigma \cup \mathrm{fv}\,t \end{aligned}$$

Write $\boxed{x \# \sigma}$ to mean that $x \notin \operatorname{dom}\sigma \cup \operatorname{fv}\sigma$.

Then the simply-typed λ-term $\boxed{t[\sigma]}$ resulting from applying the substitution σ to the simply-typed λ-term t is well-defined by:

$$
\begin{aligned}
x[\diamond] &= x \\
x[\sigma, x := t] &= t \\
x[\sigma, x' := t] &= x[\sigma] && \text{if } x \neq x' \\
\mathtt{c}[\sigma] &= \mathtt{c} \\
(t\ ,\ t')[\sigma] &= (t[\sigma]\ ,\ t'[\sigma]) \\
(\mathtt{fst}\, t)[\sigma] &= \mathtt{fst}(t[\sigma]) \\
(\mathtt{snd}\, t)[\sigma] &= \mathtt{snd}(t[\sigma]) \\
(\lambda x : A.\, t)[\sigma] &= \lambda x : A.\, (t[\sigma]) && \text{if } x \# \sigma \\
(t\, t')[\sigma] &= (t[\sigma])(t'[\sigma])
\end{aligned}
$$

Recall that simply-typed λ-terms are α-equivalence classes of syntax trees. One has to check that not only does the above definition respect α-equivalence, but also it gives a totally defined function; it does so because in the penultimate clause, modulo α-equivalence we can always choose the λ-bound variable x so that $x \# \sigma$ holds.

Note that $t[\diamond, x_1 := t_1, \ldots, x_n := t_n]$ is a **simultaneous substitution** of t_i for free occurrences of x_i in t for all $i = 1, \ldots, n$ and that may be different from an iterated single-substitution. For example $x[\diamond, x := y, y := z] = y$, whereas $(x[\diamond, x := y])[\diamond, y := z] = z$. We write $\boxed{t'[t/x]}$ for the **single-substitution** $t'[\diamond, x := t]$.

The relation $\boxed{\Gamma' \vdash \sigma : \Gamma}$ that σ is a well-formed substitution between the typing environments Γ' and Γ is inductively defined by:

$$
\frac{\Gamma' \text{ ok}}{\Gamma' \vdash \diamond : \diamond}
\qquad\qquad
\frac{\Gamma' \vdash \sigma : \Gamma \qquad x \notin \operatorname{dom}\Gamma \qquad \Gamma' \vdash t : A}{\Gamma' \vdash (\sigma, x := t) : (\Gamma, x : A)}
$$

Here are some simple properties of this relation that we need, and that can be proved by induction on its derivation:

Lemma 5.1. *If* $\Gamma' \vdash \sigma : \Gamma$*, then*

1. Γ ok *and* Γ' ok
2. $w : \Gamma'' \rhd \Gamma'$ *implies* $\Gamma'' \vdash \sigma : \Gamma$
3. $x \notin \operatorname{dom}\Gamma \cup \operatorname{dom}\Gamma'$ *implies* $\Gamma', x : A \vdash (\sigma, x := x) : (\Gamma, x : A)$

□

Lemma 5.2. *If* $\Gamma \vdash t : A$ *and* $\Gamma' \vdash \sigma : \Gamma$*, then* $\Gamma' \vdash t[\sigma] : A$*.*

Proof. By induction on the derivation of $\Gamma \vdash t : A$. The induction step for λ-abstractions uses Lemma 5.1(3) together with the easily proved property of substitution that $x \# \sigma$ implies $x[\sigma] = x$ and $t[\sigma, x := x] = t[\sigma]$. □

Given a function M mapping ground types and constants to objects and global sections in a ccc $\mathbf{C}$, we can interpret substitutions $\Gamma' \vdash \sigma : \Gamma$ as morphisms $\boxed{M[\![\Gamma' \vdash \sigma : \Gamma]\!] : M[\![\Gamma']\!] \to M[\![\Gamma]\!]}$ like so:

$$M[\![\Gamma' \vdash \diamond : \diamond]\!] = M[\![\Gamma']\!] \xrightarrow{\langle\rangle} 1$$
$$M[\![\Gamma' \vdash (\sigma, x := t) : (\Gamma, x : A)]\!] = M[\![\Gamma']\!] \xrightarrow{\langle M[\![\Gamma' \vdash \sigma : \Gamma]\!], M[\![\Gamma' \vdash t : A]\!]\rangle} M[\![\Gamma]\!] \times M[\![A]\!]$$

Lemma 5.3. *If $\Gamma' \vdash \sigma : \Gamma$ and $x \notin \operatorname{dom}\Gamma \cup \operatorname{dom}\Gamma'$, then the meaning of $\Gamma', x : A \vdash (\sigma, x := x) : (\Gamma, x : A)$ (which is valid by Lemma 5.1(3)) is $M[\![\Gamma' \vdash \sigma : \Gamma]\!] \times \mathrm{id} : M[\![\Gamma']\!] \times M[\![A]\!] \to M[\![\Gamma]\!] \times M[\![A]\!]$.*

Proof. By the definition of $M[\![\Gamma', x : A \vdash (\sigma, x := x) : (\Gamma, x : A)]\!]$, Lemma 4.1 and the fact that in a cartesian category one always has $f \times \mathrm{id} = \langle f \circ \pi_1, \pi_2\rangle$. $\square$

Theorem 5.4 (Semantics of simultaneous substitution)**.** *If $\Gamma \vdash t : A$ and $\Gamma' \vdash \sigma : \Gamma$, then then the following diagram commutes in $\mathbf{C}$:*

$$\begin{tikzcd}
M[\![\Gamma']\!] \arrow[r, "M[\![\Gamma' \vdash \sigma : \Gamma]\!]"] \arrow[dr, "M[\![\Gamma' \vdash t[\sigma] : A]\!]"'] & M[\![\Gamma]\!] \arrow[d, "M[\![\Gamma \vdash t : A]\!]"] \\
& M[\![A]\!]
\end{tikzcd}$$

Proof. By induction on the derivation of $\Gamma \vdash t : A$. For the induction step for λ-abstractions one uses Lemma 5.3 and the fact that in a cartesian closed category the Currying operation satisfies $\mathrm{cur}(f \circ (g \times \mathrm{id})) = (\mathrm{cur}\, f) \circ g$. $\square$

Lemma 5.5 (Identity substitution)**.** *For each typing environment Γ, define the substitution id_Γ by:*

$$\begin{aligned} id_\diamond &= \diamond \\ id_{\Gamma, x:A} &= (id_\Gamma, x := x) \end{aligned}$$

1. *If Γ ok, then $\Gamma \vdash \mathrm{id}_\Gamma : \Gamma$.*
2. *If $\Gamma \vdash t : A$ and $\Gamma, x : A \vdash t' : A'$, then*

$$\begin{aligned} & \Gamma \vdash (\mathrm{id}_\Gamma, x := t) : (\Gamma, x : A), \\ & t'[t/x] = t'[\mathrm{id}_\Gamma, x := t] \\ \textit{and}\quad & \Gamma \vdash t'[t/x] : A' \end{aligned}$$

3. *$M[\![\Gamma \vdash \mathrm{id}_\Gamma : \Gamma]\!]$ is equal to the identity morphism on $M[\![\Gamma]\!]$.*

Proof. By induction on the derivation of Γ ok, using Lemma 5.2 for part (2). $\square$

Corollary 5.6 (Semantics of single substitution)**.** *If $\Gamma \vdash t : A$ and $\Gamma, x : A \vdash t' : A'$, then the following diagram commutes in* $\mathbf{C}$*:*

$$\begin{tikzcd}
M[\![\Gamma]\!] \arrow[r, "\langle \mathrm{id}, M[\![\Gamma \vdash t : A]\!]\rangle"] \arrow[dr, "M[\![\Gamma \vdash t'[t/x] : A']\!]"'] & M[\![\Gamma]\!] \times M[\![A]\!] \arrow[d, "M[\![\Gamma, x : A \vdash t' : A']\!]"] \\
& M[\![A']\!]
\end{tikzcd}$$

Proof. The result is a special case of Theorem 5.4 for the simultaneous substitution $\Gamma \vdash (\mathrm{id}_\Gamma, x := t) : (\Gamma, x : A)$, using Lemma 5.5. □

6 $\beta\eta$-Equality of simply-typed λ-terms

The relation $\boxed{\Gamma \vdash t =_{\beta\eta} t' : A}$ is inductively defined by the following rules:

equivalence relation

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t =_{\beta\eta} t : A} \qquad \frac{\Gamma \vdash t_1 =_{\beta\eta} t_2 : A}{\Gamma \vdash t_2 =_{\beta\eta} t_1 : A} \qquad \frac{\Gamma \vdash t_1 =_{\beta\eta} t_2 : A \qquad \Gamma \vdash t_2 =_{\beta\eta} t_3 : A}{\Gamma \vdash t_1 =_{\beta\eta} t_3 : A}$$

β-conversions

$$\frac{\Gamma, x : A \vdash t : A' \qquad \Gamma \vdash t' : A}{\Gamma \vdash (\lambda x : A.\, t)\, t' =_{\beta\eta} t[t'/x] : A'}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash t' : A'}{\Gamma \vdash \mathtt{fst}\,(t\ ,\ t') =_{\beta\eta} t : A} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash t' : A'}{\Gamma \vdash \mathtt{snd}\,(t\ ,\ t') =_{\beta\eta} t' : A'}$$

η-conversions

$$\frac{\Gamma \vdash t : A \to A' \qquad x \notin \mathrm{fv}\, t}{\Gamma \vdash t =_{\beta\eta} \lambda x : A.\, (t\, x) : A \to A'}$$

$$\frac{\Gamma \vdash t : A \times A'}{\Gamma \vdash t =_{\beta\eta} (\mathtt{fst}\, t\ ,\ \mathtt{snd}\, t) : A \times A'} \qquad \frac{\Gamma \vdash t : \mathtt{unit}}{\Gamma \vdash t =_{\beta\eta} () : \mathtt{unit}}$$

congruence rules

$$\frac{\Gamma \vdash t_1 =_{\beta\eta} t_2 : A \qquad \Gamma \vdash t_1' =_{\beta\eta} t_2' : A'}{\Gamma \vdash (t_1\ ,\ t_1') =_{\beta\eta} (t_2\ ,\ t_2') : A \times A'} \qquad \frac{\Gamma \vdash t_1 =_{\beta\eta} t_2 : A \times A'}{\Gamma \vdash \mathtt{fst}\, t_1 =_{\beta\eta} \mathtt{fst}\, t_2 : A}$$

$$\frac{\Gamma \vdash t_1 =_{\beta\eta} t_2 : A \times A'}{\Gamma \vdash \mathtt{snd}\, t_1 =_{\beta\eta} \mathtt{snd}\, t_2 : A'} \qquad \frac{\Gamma, x : A \vdash t_1 =_{\beta\eta} t_2 : A'}{\Gamma \vdash \lambda x : A.\, t_1 =_{\beta\eta} \lambda x : A.\, t_2 : A \to A'}$$

$$\frac{\Gamma \vdash t_1 =_{\beta\eta} t_2 : A \to A' \qquad \Gamma \vdash t_1' =_{\beta\eta} t_2' : A}{\Gamma \vdash t_1\, t_1' =_{\beta\eta} t_2\, t_2' : A'}$$

Lemma 6.1. *If $\Gamma \vdash t =_{\beta\eta} t' : A$, then $\Gamma \vdash t : A$ and $\Gamma \vdash t' : A$.*

Proof. By induction on the derivation of $\Gamma \vdash t =_{\beta\eta} t' : A$, using Lemma 5.2 for the first β-conversion rule and Lemma 2.2(2) for first η-conversion rule. □

Theorem 6.2 (Soundness)**.** *For any function M mapping ground types and constants to objects and global sections in a cartesian closed category $\mathbf{C}$, the associated semantics of types and terms (Section 4) satisfies that if $\Gamma \vdash t =_{\beta\eta} t' : A$ is derivable, then $M[\![\Gamma \vdash t : A]\!]$ and $M[\![\Gamma \vdash t' : A]\!]$ are equal morphisms in $\mathbf{C}(M[\![\Gamma]\!], M[\![A]\!])$.*

Proof. One has to check that the relation

$$\Gamma \vdash t : A \text{ and } \Gamma \vdash t' : A \text{ and } M[\![\Gamma \vdash t : A]\!] = M[\![\Gamma \vdash t' : A]\!]$$

is closed under the above rules inductively generating the relation $\beta\eta$-equality relation. Here is the argument for the β-conversion involving λ-abstraction

$$\frac{\Gamma, x : A \vdash t : A' \qquad \Gamma \vdash t' : A}{\Gamma \vdash (\lambda x : A.\, t)\, t' =_{\beta\eta} t[t'/x] : A'}$$

Given $\Gamma, x : A \vdash t : A'$ and $\Gamma \vdash t' : A$, define

$$\begin{aligned}
X &= M[\![A]\!] \\
Y &= M[\![\Gamma]\!] \\
Z &= M[\![A']\!] \\
f &= M[\![\Gamma, x : A \vdash t : A']\!] \\
g &= M[\![\Gamma \vdash t' : A]\!]
\end{aligned}$$

Thus $f : Y \times X \to Z$ and $g : Y \to X$ in the ccc $\mathbf{C}$ and

$$M[\![\Gamma \vdash (\lambda x : A.\, t)\, t' : A']\!] = \mathrm{app} \circ \langle \mathrm{cur}\, f, g \rangle : Y \to Z$$
(by definition of the semantics of terms)
$$M[\![\Gamma \vdash t[t'/x] : A']\!] = f \circ \langle \mathrm{id}_Y, g \rangle : Y \to Z$$
(by Corollary 5.6)

But in any ccc we have $\mathrm{app} \circ \langle \mathrm{cur}\, f, g \rangle = \mathrm{app} \circ (\mathrm{cur}\, f \times \mathrm{id}_X) \circ \langle \mathrm{id}_Y, g \rangle = f \circ \langle \mathrm{id}_Y, g \rangle$. Therefore $M[\![\Gamma \vdash (\lambda x : A.\, t)\, t' : A']\!] = M[\![\Gamma \vdash t[t'/x] : A']\!]$, as required.

Here is the argument for the η-conversion involving λ-abstraction

$$\frac{\Gamma \vdash t : A \to A' \qquad x \notin \mathrm{fv}\, t}{\Gamma \vdash t =_{\beta\eta} \lambda x : A.\, (t\, x) : A \to A'}$$

Given $\Gamma \vdash t : A \to A'$ and $x \notin \mathrm{fv}(t)$, without loss of generality we may assume also that $x \notin \mathrm{dom}\, \Gamma$ (since $\lambda x : A.\, (t\, x) =_\alpha \lambda x' : A.\, (t\, x')$ for any $x' \notin \mathrm{fv}\, t \cup \mathrm{dom}\, \Gamma$). Define

$$\begin{aligned}
X &= M[\![A]\!] \\
Y &= M[\![\Gamma]\!] \\
Z &= M[\![A']\!] \\
h &= M[\![\Gamma \vdash t : A \to A']\!]
\end{aligned}$$

Thus $h : Y \to Z^X$ in $\mathbf{C}$ and

$$M[\![\Gamma, x : A \vdash t : A \to A']\!] = h \circ \pi_1 : Y \times X \to Z^X$$
(by Lemma 4.1)
$$M[\![\Gamma, x : A \vdash x : A]\!] = \pi_2 : Y \times X \to X$$
(by definition of the semantics of terms)

Hence $M[\![\Gamma \vdash \lambda x : A.\,(t\,x) : A \to A']\!] = \mathrm{cur}(\mathrm{app} \circ \langle h \circ \pi_1, \pi_2 \rangle)$. But in any ccc we have $\mathrm{cur}(\mathrm{app} \circ \langle h \circ \pi_1, \pi_2 \rangle) = \mathrm{cur}(\mathrm{app} \circ (h \times \mathrm{id}_X)) = h$ and therefore $M[\![\Gamma \vdash t : A \to A']\!] = M[\![\Gamma \vdash \lambda x : A.\,(t\,x) : A \to A']\!]$, as required.

We leave checking closure under the other rules of $\beta\eta$-equivalence as an exercise. $\square$

7 The internal language of a cartesian closed category

Given a particular cartesian closed category $\mathbf{C}$, we can take $\mathrm{obj}\,\mathbf{C}$ to be the set of ground types and take each global element $f \in \mathbf{C}(1, X)$ (for any $\mathbf{C}$-object X) to be a constant of type X. Taking the interpretation M to be the identity function, then the simple types and the simply typed λ-terms over this collection of ground types and constants provides a convenient language for describing the objects and morphisms of $\mathbf{C}$ and their (equational) properties.

For example if X, Y and Z are three objects in a ccc $\mathbf{C}$, then there is always an isomorphism

$$Z^{X \times Y} \cong (Z^Y)^X$$

One can construct the morphisms that constitute this isomorphism and prove they are mutually inverse only using the universal properties of products and exponentials in $\mathbf{C}$. However, the internal language allows us describe the morphisms and prove that they are inverse via properties of $\beta\eta$-equivalence; furthermore these descriptions look like what one expect when $\mathbf{C}$ is the category of sets and functions:

$$s \triangleq \lambda f : (X \times Y) \to Z.\, \lambda x : X.\, \lambda y : Y.\, f\,(x\,,\,y)$$
$$t \triangleq \lambda g : X \to (Y \to Z).\, \lambda z : X \times Y.\, g\,(\mathtt{fst}\,z)\,(\mathtt{snd}\,z)$$

satisfy

$$\diamond \vdash s : ((X \times Y) \to Z) \to (X \to (Y \to Z))$$
$$\diamond \vdash t : (X \to (Y \to Z)) \to ((X \times Y) \to Z)$$
$$\diamond, f : (X \times Y) \to Z \vdash t\,(s\,f) =_{\beta\eta} f : (X \times Y) \to Z$$
$$\diamond, g : X \to (Y \to Z) \vdash s\,(t\,g) =_{\beta\eta} g : X \to (Y \to Z)$$

8 Free cartesian closed categories

Theorem 6.2 has a converse – a *completeness* theorem: given $\Gamma \vdash t : A$ and $\Gamma \vdash t' : A$, if $M[\![\Gamma \vdash t : A]\!] = M[\![\Gamma \vdash t' : A]\!]$ holds for any interpretation M of the ground types and constants in any ccc, then $\Gamma \vdash t =_{\beta\eta} t' : A$ is derivable. In fact for any set of ground types and constants, there is a particular *freely generated* ccc $\mathbf{F}$ containing an interpretation M of the ground types and constants satisfying

$$M[\![\Gamma \vdash t : A]\!] = M[\![\Gamma \vdash t' : A]\!] \;\Leftrightarrow\; \Gamma \vdash t =_{\beta\eta} t' : A \tag{1}$$

$\mathbf{F}$ is constructed from the syntax of the simply typed λ-calculus quotiented by $\beta\eta$-equivalence. Specifically, one can take $\mathrm{obj}\,\mathbf{F} = ST(Gnd)$. For two such objects $A, A' \in ST(Gnd)$, we take $\mathbf{F}(A, A')$ to be the quotient of the set $\{t \mid \diamond \vdash t : A \to A'\}$ of closed terms (i.e. those with no free variables) of type $A \to A'$ by the equivalence relation relating two such terms t and t' if $\diamond \vdash t =_{\beta\eta} t' : A \to A'$ holds. The identity morphism in $\mathbf{F}$ on A is the equivalence class of $\lambda x : A.\, x$. The composition of two morphisms represented by terms $\diamond : t : A \to A'$ and $\diamond \vdash t' : A' \to A''$ is well-defined by taking the equivalence class of the term $\diamond \vdash \lambda x : A.\, t'\,(t\, x) : A \to A''$. One has to check that this recipe does give a category and that it is cartesian closed; unsurprisingly, the terminal object is $\mathtt{unit}$, the product of objects $A, A' \in ST(Gnd)$ is the simple type $A \times A'$ (equipped with the obvious projection morphisms) and their exponential is the simple type $A \to A'$ (equipped with the obvious application morphism).

Taking M to map each ground type $G \in Gnd$ to $G \in \mathrm{obj}\,\mathbf{F}$ and each constant $\mathtt{c}^A$ to the global element $M\,\mathtt{c} \in \mathbf{F}(\mathtt{unit}, A)$ given by the equivalence class of the term $\diamond \vdash \lambda x : \mathtt{unit}.\, \mathtt{c} : \mathtt{unit} \to A$, one can show that this interpretation has property (1).

$\mathbf{F}$ is a *free* ccc in a similar sense to Σ^* being the free monoid on a set Σ – there is a universal property that characterises it, whose statement in terms of morphisms of cartesian closed categories is beyond the scope of these notes (see Crole [1993, Section 4.8]).

References

J. M. Chapman. *Type Checking and Normalisation*. PhD thesis, University of Nottingham, 2009. URL http://eprints.nottingham.ac.uk/id/eprint/10824. [Cited on page 3.]

R. L. Crole. *Categories for Types*. Cambridge University Press, 1993. [Cited on page 12.]

N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34:381–392, 1972. [Cited on page 2.]

C. Keller and T. Altenkirch. Hereditary substitutions for simple types, formalized. In *Proceedings of the Third ACM SIGPLAN Workshop on Mathematically Structured Functional Programming*, MSFP ’10, pages 3–10, New York, NY, USA, 2010. ACM. URL http://doi.acm.org/10.1145/1863597.1863601. [Cited on page 2.]

A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013. [Cited on page 2.]