Canonical Forms LF for Computation and Deduction 2009

Chris Martens

February 12, 2009

$$\begin{array}{rrcl} \text{kinds} & K & ::= & \mathsf{type} \mid \Pi x{:}A.K \\ \text{atomic families} & P & ::= & a \mid P\ M \\ \text{families} & A & ::= & P \mid \Pi x{:}A.A \\ \text{atoms} & R & ::= & x \mid c \mid R\ M \\ \text{terms} & M & ::= & R \mid \lambda x.M \end{array}$$

1 Typing

1.1 Kind Well-Formedness

$$\frac{}{\Gamma \vdash \mathsf{type}\ \mathsf{wf}} \qquad \frac{\Gamma, x{:}A \vdash K\ \mathsf{wf}}{\Gamma \vdash \Pi x{:}A.K\ \mathsf{wf}}$$

1.2 Atomic Family Kinding

$$\frac{\Sigma(a) = K \quad \Gamma \vdash K\ \mathsf{wf}}{\Gamma \vdash a : K} \qquad \frac{\Gamma \vdash P : \Pi x{:}A.K \quad \Gamma \vdash M \Leftarrow A \quad [M/x]\,K\ \mathsf{is}\ K'}{\Gamma \vdash P\ M : K'}$$

1.3 Type Well-Formedness

$$\frac{\Gamma \vdash P : \mathsf{type}}{\Gamma \vdash P\ \mathsf{wf}} \qquad \frac{\Gamma \vdash A\ \mathsf{wf} \quad \Gamma, x{:}A \vdash B\ \mathsf{wf}}{\Gamma \vdash \Pi x{:}A.B\ \mathsf{wf}}$$

1.4 Type Synthesis (for Atoms)

$$\frac{\Sigma(c) = A \quad \Gamma \vdash A\ \mathsf{wf}}{\Gamma \vdash c \Rightarrow A} \qquad \frac{\Gamma(x) = A \quad \Gamma \vdash A\ \mathsf{wf}}{\Gamma \vdash x \Rightarrow A} \qquad \frac{\Gamma \vdash R \Rightarrow \Pi x{:}A.B \quad \Gamma \vdash M \Leftarrow A \quad [M/x]\,B\ \mathsf{is}\ B'}{\Gamma \vdash R\ M \Rightarrow B'}$$

1.5 Type Checking (for Terms)

$$\frac{\Gamma \vdash R \Rightarrow P}{\Gamma \vdash R \Leftarrow P} \qquad \frac{\Gamma \vdash A\ \mathsf{wf} \quad \Gamma, x{:}A \vdash M \Leftarrow B}{\Gamma \vdash \lambda x.M \Leftarrow \Pi x{:}A.B}$$

In type synthesis, the type is thought of as an output, while in type checking, the type is thought of as an input to be verified.

Note that anytime a rule induces a transition from something atomic to something normal, the something must have a base type or kind.

2 Substitution

Take a moment to note that in the grammar given, variables (x) are in the syntactic category of atoms; that is, the things they stand for (and thus the things that we may substitute for them) are atoms. But note that all forms of application involve applying something to a *term*, and so in the typing rules for an application, we have to substitute the *term* in for the variable in the body of the dependent type (or kind) of the atomic applicand. Such a notion has not yet been defined.

Defining the notion of substitution we want is actually nontrivial. The problem arises because, given the usual definition of substitution, it could create beta reductions. A trivial example is

$$[\lambda x.x/y]\,(y\ z)\ \mathsf{is}\ (\lambda x.x)\ z$$

for some constant z.

One approach would be simply to perform this substitution in one phase and then normalize the resulting term in a second phase. But since we would like to work entirely within the grammar of canonical forms, we instead do the normalization *as we substitute*. This formulation of substitution is called *hereditary substitution*.

2.1 Hereditary Substitution

There are several judgments, each one a 4-place relation (including the variable):

$[M/x]_{\mathrm{RR}}\,R\ \mathsf{is}\ R'$	Substitute M into an atom R and get an atom R'
$[M/x]_{\mathrm{RM}}\,R\ \mathsf{is}\ N$	Substitute M into an atom R and get a term N
$[M/x]_{\mathrm{M}}\,N\ \mathsf{is}\ N'$	Substitute M into a term N and get a term N'
$[M/x]_{\mathrm{P}}\,P\ \mathsf{is}\ P'$	Substitute M into an atomic family P and get an atomic family P'
$[M/x]_{\mathrm{A}}\,A\ \mathsf{is}\ A'$	Substitute M into a family A and get a family A'
$[M/x]_{\mathrm{K}}\,K\ \mathsf{is}\ K'$	Substitute M into a kind K and get a kind K'

Substitution into an atom results in a term (thus the need for the RM case) when the variable being substituted for is the *head variable*, which can be defined by the following predicate:

$$\mathsf{headvar}(x) = x$$
$$\mathsf{headvar}(R\ M) = \mathsf{headvar}(R)$$

That is, if the variable being substituted for is at the head of a (possibly nullary) application, then plugging in a term for it will result in a term. The reader should mentally check that none of the other kinds of substitution can incur a change in the syntactic class of the output.

Substitution is defined as follows:

$$\frac{}{[M/x]_{\mathrm{RR}}\,c\ \mathsf{is}\ c} \qquad \frac{}{[M/x]_{\mathrm{RR}}\,y\ \mathsf{is}\ y} \qquad \frac{[M/x]_{\mathrm{RR}}\,R\ \mathsf{is}\ R' \quad [M/x]_{\mathrm{M}}\,N\ \mathsf{is}\ N'}{[M/x]_{\mathrm{RR}}\,(R\ N)\ \mathsf{is}\ R'\ N'}$$

$$\frac{}{[M/x]_{\mathrm{RM}}\,x\ \mathsf{is}\ M} \qquad \frac{[M/x]_{\mathrm{RM}}\,R\ \mathsf{is}\ \lambda y.O \quad [M/x]_{\mathrm{M}}\,N\ \mathsf{is}\ N' \quad [N'/x]_{\mathrm{M}}\,O\ \mathsf{is}\ O'}{[M/x]_{\mathrm{RM}}\,(R\ N)\ \mathsf{is}\ O'}$$

The above are the key rules, particularly the last one: note that the result of substituting into a thing with pi type, if it is a term, must be canonical (a lambda). Thus the rule can pattern match the result of the inductive call to substitution as a lambda and recursively substitute into the body.

The remainder of the rules should contain no surprises.

$$\frac{[M/x]_{\mathrm{RR}}\,R\ \mathsf{is}\ R'}{[M/x]_{\mathrm{M}}\,R\ \mathsf{is}\ R'} \qquad \frac{[M/x]_{\mathrm{RM}}\,R\ \mathsf{is}\ N}{[M/x]_{\mathrm{M}}\,R\ \mathsf{is}\ N} \qquad \frac{[M/x]_{\mathrm{M}}\,N\ \mathsf{is}\ N'}{[M/x]_{\mathrm{M}}\,\lambda y.N\ \mathsf{is}\ \lambda y.N'}$$

$$\frac{}{[M/x]_{\mathrm{P}}\, a \text{ is } a} \qquad \frac{[M/x]_{\mathrm{P}}\, P \text{ is } P' \quad [M/x]_{\mathrm{M}}\, N \text{ is } N'}{[M/x]_{\mathrm{P}}\, (P\ N) \text{ is } P'\ N'}$$

$$\frac{[M/x]_{\mathrm{P}}\, P \text{ is } P'}{[M/x]_{\mathrm{A}}\, P \text{ is } P'} \qquad \frac{[M/x]_{\mathrm{A}}\, A \text{ is } A' \quad [M/x]_{\mathrm{A}}\, B \text{ is } B'}{[M/x]_{\mathrm{A}}\, \Pi y{:}A.B \text{ is } \Pi y{:}A'.B'}$$

$$\frac{}{[M/x]_{\mathrm{K}}\, \mathsf{type} \text{ is } \mathsf{type}} \qquad \frac{[M/x]_{\mathrm{A}}\, A \text{ is } A' \quad [M/x]_{\mathrm{K}}\, K \text{ is } K'}{[M/x]_{\mathrm{K}}\, \Pi y{:}A.K \text{ is } \Pi y{:}A'.K'}$$

2.2 Termination Argument

The argument that hereditary substitution terminates proceeds by lexicographic induction on the thing being substituted into and the *type of the term being substituted*. In most of the rules, the former induction metric works: you always substitute into a smaller term. However, in the key rule mentioned above, suddenly the thing being substituted into is completely unrelated syntactically to the original thing, so we must instead argue that the type of the substitutend gets smaller (assuming all of the inputs are well-typed). It is instructive to think through said argument.

3 Further Reading

See also:

- Harper-Licata '07: *http://www.cs.cmu.edu/ drl/pubs/hl07mechanizing/hl07mechanizing.pdf*
- Kevin Watkins' thesis proposal: *http://www.cs.cmu.edu/ kw/pubs/proposal.ps*