CS101C
Type Theory and Formal Methods

Lecture 10

May 5, 2003

"Canonical" Operators

For each operator in type theory, we will say whether it is "*canonical*" or not. Informally, a "canonical" operators represent the end-results of computation and non-canonical ones represent intermediate results.

Operator	Canonical?	Reduction for non-canonical case
`apply`	No	$\texttt{apply}\{\texttt{lambda}\{x.t[x]\}; a\} \rightsquigarrow t[a]$
`lambda`	Yes	
`pair`	Yes	
`fst`, `snd`	No	$\texttt{fst}\{\texttt{pair}\{a; b\}\} \rightsquigarrow a$
`inl`, `inr`	Yes	
`decide`	No	$\texttt{decide}\{\texttt{inl}\{a\}; x.l[x]; y.r[y]\} \rightsquigarrow l[a]$ $\texttt{decide}\{\texttt{inr}\{a\}; x.l[x]; y.r[y]\} \rightsquigarrow r[a]$
`union`	Yes	

Canonical Terms and Types

A *closed* term is *canonical* if its top level operator is canonical.

Types are defined based on what *canonical* terms have that type. A non-canonical *closed* term t that evaluates to a canonical t' has whatever types t' has.

In our type theory, non-terminating computations do not have types.

Examples.

$A \times B$ is a type of pairs $\texttt{pair}\{a; b\}$ such that $a \in A$ and $b \in B$.

$A + B$ is a type of terms $\texttt{inl}\{a\}$ where $a \in A$ and terms $\texttt{inr}\{b\}$ where $b \in B$.

$A \rightarrow B$ is a type of lambdas $\texttt{lambda}\{x.t[x]\}$ such that for any $x \in A$, $t[x] \in B$

What is equality?

When can we say that $\lambda x.t_1[x] = \lambda x.t_2[x]$?

Intensional equality: when they are the same terms (e.g. α-equal or similar).

Extensional equality: when they compute the same function.

Equality

We have defined "$\lambda x.t_1[x] = \lambda x.t_2[x] \in (A \to B)$" as "for any $a_1 = a_2 \in A$, it must be the case that $t_1[a_1] = t_2[a_2] \in B$".

We would expect that when $\lambda x.t[x] \in (A \to B)$, then also $\lambda x.t[x] = \lambda x.t[x] \in (A \to B)$.

This means that when $\lambda x.t[x] \in (A \to B)$, then whenever $a = a' \in A$, then also $t[a] = t[a'] \in B$!

Equality and Dependent Types

How should we define $\lambda x.t_1[x] = \lambda x.t_2[x] \in (x : A \rightarrow B[x])$?

Answer: as "for any $a_1 = a_2 \in A$, it must be the case that $t_1[a_1] = t_2[a_2] \in B[a_1]$" and the type "$x : A \rightarrow B[x]$" is only well-formed when for any $a_1 = a_2 \in A$, $B[a_1] = B[a_2]$.

We will write "$r_1 = r_2 \in T_1 = T_2$" as an abbreviation for "$T_1 = T_2$ and $r_1 = r_2 \in T_1$" and "$r_1 = r_2 \in T$" will mean "T is a well-formed type expression and r_1 and r_2 are well-formed elements of T that are equal in T."

Equality and Sequents

We expect the following sequents to mean the same thing:

$$\Gamma \vdash \lambda x.t[x] \in (A \rightarrow B[x])$$
$$\Gamma; x : A \vdash t[x] \in B[x]$$
$$\Gamma; x : A \vdash B[x] \texttt{ ext } t[x]$$

where $\texttt{ext}\ t[x]$ is a notation for “the evidence of this is $t[x]$” ($\texttt{ext}$ stands for “extract”, as in “the evidence we’ll extract from the proof”).

Semantics of sequents

Therefore we need to add the "equality" part to the definition of sequent semantics:

$$x_1 : A_1;\ x_2 : A_2[x_1];\ \cdots;\ x_n : A_n[x_1; \cdots ; x_{n-1}] \vdash C[x_1; \cdots ; x_n]\ \texttt{ext}\ t[x_1; \cdots ; x_n]$$

is "true" when for any $a_1, \ldots, a_n, a'_1, \ldots, a'_n$, whenever

$$a_1 = a'_1 \in A_1 \text{ and } a_2 = a'_2 \in A_2[a_1] = A_2[a'_1] \text{ and } \cdots$$
$$a_n = a'_n \in A_n[a_1; \cdots ; a_{n-1}] = A_n[a'_1; \cdots ; a'_{n-1}],$$

then also

$$t[a_1; \cdots ; a_n] = t[a'_1; \cdots ; a'_{n-1}] \in C[a_1; \cdots ; a_n] = C[a'_1; \cdots ; a'_{n-1}].$$