IIT CS534: Types and Programming Languages

Rule Induction, Syntax, and Small-step Semantics

Lecture given by Farzaneh Derakhshan, Notes by Stefan Muller

Out: Tuesday, Jan. 16

1 Rule Induction

Below are the proofs done in class, written out formally as you would write them on a homework.

First, here are the rules for our "is XML" judgment.

$$\frac{}{\text{``'' is XML}}\ (\text{X-1}) \qquad \frac{}{\texttt{<T>}\text{ is XML}}\ (\text{X-2}) \qquad \frac{X \text{ is XML}}{\texttt{<T>}X\texttt{</T>}\text{ is XML}}\ (\text{X-3}) \qquad \frac{X \text{ is XML} \quad Y \text{ is XML}}{X\ Y \text{ is XML}}\ (\text{X-4})$$

Let *OpenAngle*(X) be the number of open angle brackets in a string X and *CloseAngle*(X) be the number of close angle brackets in X.

Theorem 1. *If X is XML, then $\mathit{OpenAngle}(X) = \mathit{CloseAngle}(X)$.*

Proof. By induction on the derivation of X is XML.

- Case X-1. Then $X =$ "" and $\mathit{OpenAngle}(X) = \mathit{CloseAngle}(X) = 0$.
- Case X-2. Then $X = \texttt{<T>}$ and $\mathit{OpenAngle}(X) = \mathit{CloseAngle}(X) = 1$.
- Case X-3. Then $X = \texttt{<T>}Y\texttt{</T>}$ and Y is XML. We have $\mathit{OpenAngle}(X) = 2 + \mathit{OpenAngle}(Y)$ and $\mathit{CloseAngle}(X) = 2 + \mathit{CloseAngle}(Y)$. By induction, $\mathit{OpenAngle}(Y) = \mathit{CloseAngle}(Y)$, so $2 + \mathit{OpenAngle}(Y) = 2 + \mathit{CloseAngle}(Y)$.
- Case X-4. Then $X = Y\ Z$ and Y is XML and Z is XML. We have $\mathit{OpenAngle}(X) = \mathit{OpenAngle}(Y) + \mathit{OpenAngle}(Z)$ and $\mathit{CloseAngle}(X) = \mathit{CloseAngle}(Y) + \mathit{CloseAngle}(Z)$. By induction, $\mathit{OpenAngle}(Y) = \mathit{CloseAngle}(Y)$ and $\mathit{OpenAngle}(Z) = \mathit{CloseAngle}(Z)$, so $\mathit{OpenAngle}(X) = \mathit{CloseAngle}(X)$.

□

Now here are the rules for natural numbers (both constructing a natural number and the greater-than judgment)

$$\frac{}{0 \text{ is a natural number}}\ (\text{ZERO}) \qquad \frac{n \text{ is a natural number}}{n+1 \text{ is a natural number}}\ (\text{SUCC}) \qquad \frac{n \text{ is a natural number}}{n \geq n}\ (\text{GE-NAT})$$

$$\frac{m \text{ is a natural number} \quad m \geq n}{m+1 \geq n}\ (\text{GE-SUCC})$$

Theorem 2. *If n is a natural number then $n \geq 0$.*

Proof. By induction on the derivation of n is a natural number.

- Rule ZERO. Then $n = 0$. By GE-NAT, we have $n \geq 0$.
- Rule SUCC. Then $n = m + 1$ and m is a natural number. By induction, $m \geq 0$. By GE-SUCC, we have $m + 1 \geq 0$.

□

2 E Language

2.1 Syntax

We will be working with a small language called E consisting of integer and string expressions. The grammar below is in BNF (Backus-Naur Form). We use $e ::= A \mid B \mid ...$ to mean that an expression (with metavariable e) can look like form A or B, and so on.

$$
\begin{array}{rcll}
e & ::= & \overline{n} & \text{Numbers} \\
 & | & \text{“}s\text{”} & \text{Strings} \\
 & | & e + e & \text{Addition} \\
 & | & e \mathbin{\hat{}} e & \text{Concatenation} \\
 & | & |e| & \text{String Length}
\end{array}
$$

2.2 Small-step semantics

We also studied the small-step semantics of E. There are two judgments, $e\ \mathsf{val}$ meaning that e is a value and can't step anymore, and $e \mapsto e'$ meaning that e steps to e'. The rules for these judgments are below. Note that in rules S-1 and S-3, when we do $n_1 + n_2$ or $|s|$ (as opposed to $\overline{n_1} + \overline{n_2}$ and $|\text{“}s\text{”}|$), these are actually taking the mathematical addition of two integers and the actual number of characters in a string literal. We just use the same symbols for the actual underlying operation and for the syntax of the programming language. Rules S-4 through S-8 are “search” rules that allow us to step subexpressions.

$$
\frac{}{\overline{n}\ \mathsf{val}}\ (\text{V-1}) \qquad
\frac{}{\text{“}s\text{”}\ \mathsf{val}}\ (\text{V-2}) \qquad
\frac{}{\overline{n_1} + \overline{n_2} \mapsto \overline{n_1 + n_2}}\ (\text{S-1}) \qquad
\frac{}{\text{“}s_1\text{”} \mathbin{\hat{}} \text{“}s_2\text{”} \mapsto \text{“}s_1 s_2\text{”}}\ (\text{S-2})
$$

$$
\frac{}{|\text{“}s\text{”}| \mapsto \overline{|s|}}\ (\text{S-3}) \qquad
\frac{e_1 \mapsto e_1'}{e_1 + e_2 \mapsto e_1' + e_2}\ (\text{S-4}) \qquad
\frac{e_2 \mapsto e_2'}{\overline{n_1} + e_2 \mapsto \overline{n_1} + e_2'}\ (\text{S-5}) \qquad
\frac{e_1 \mapsto e_1'}{e_1 \mathbin{\hat{}} e_2 \mapsto e_1' \mathbin{\hat{}} e_2}\ (\text{S-6})
$$

$$
\frac{e_2 \mapsto e_2'}{\text{“}s_1\text{”} \mathbin{\hat{}} e_2 \mapsto \text{“}s_1\text{”} \mathbin{\hat{}} e_2'}\ (\text{S-7}) \qquad
\frac{e \mapsto e'}{|e| \mapsto |e'|}\ (\text{S-8})
$$