CS520, Handout 06, 19 Sept 2006

Typed Arithmetic Expressions

Reading: [Pierce, Chapter 8]

Syntax of Terms

From Chapt 3, without distinguishing terms that are **values**:

t ::=		*terms*
	`true`	*constant true*
	`false`	*constant false*
	`0`	*constant value*
	`if` t `then` t `else` t	*conditional*
	`succ` t	*successor*
	`pred` t	*predecessor*
	`iszero` t	*zero test*

Syntax of Types

T ::=		*types*
	`Bool`	*type of booleans*
	`Nat`	*type of natural numbers*

Typing Rules

Putting **terms** and **types** together.

Sometimes called **static semantics**, as opposed to **dynamic semantics**.
The typing rules are **syntax-directed**, i.e., they follow the inductive definition of terms.

$$\frac{}{\texttt{true} : \texttt{Bool}} \quad \text{(T-True)}$$

$$\frac{}{\texttt{false} : \texttt{Bool}} \quad \text{(T-False)}$$

$$\frac{}{0 : \texttt{Nat}} \quad \text{(T-Zero)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \quad \text{(T-If)}$$

$$\frac{t : \texttt{Nat}}{\texttt{succ}\, t : \texttt{Nat}} \quad \text{(T-Succ)}$$

$$\frac{t : \texttt{Nat}}{\texttt{pred}\, t : \texttt{Nat}} \quad \text{(T-Pred)}$$

$$\frac{t : \texttt{Nat}}{\texttt{iszero}\, t : \texttt{Bool}} \quad \text{(T-IsZero)}$$

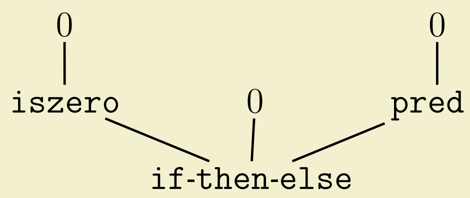
Example of a Typing Derivation

$$\dfrac{\dfrac{\dfrac{}{0 : \texttt{Nat}}\,\text{T-Zero}}{\texttt{iszero}\,0 : \texttt{Bool}}\,\text{T-IsZero} \qquad \dfrac{}{0 : \texttt{Nat}}\,\text{T-Zero} \qquad \dfrac{\dfrac{}{0 : \texttt{Nat}}\,\text{T-Zero}}{\texttt{pred}\,0 : \texttt{Nat}}\,\text{T-Pred}}{\texttt{if }(\texttt{iszero}\,0)\texttt{ then }0\texttt{ else }(\texttt{pred}\,0) : \texttt{Nat}}\,\text{T-If}$$

Example of a Typing Derivation

$$\dfrac{\dfrac{}{0 : \mathtt{Nat}}\,\text{T-Zero}}{\mathtt{iszero}\,0 : \mathtt{Bool}}\,\text{T-IsZero} \quad \dfrac{}{0 : \mathtt{Nat}}\,\text{T-Zero} \quad \dfrac{\dfrac{}{0 : \mathtt{Nat}}\,\text{T-Zero}}{\mathtt{pred}\,0 : \mathtt{Nat}}\,\text{T-Pred}$$
$$\overline{\mathtt{if}\ (\mathtt{iszero}\,0)\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{pred}\,0) : \mathtt{Nat}}\ \text{T-If}$$

Observe the correspondence with the **abstract syntax tree** (upside down):



Digression

A useful connection between the **typing derivation** of a term t and the **evaluation tree** of t according to an (appropriately adjusted) big-step operational semantics.

Big-step operational semantics [Pierce, Ex. 3.5.17, pp 42-43] adjusted by replacing rules **B-IfTrue** and **B-IfFalse** with two new rules:

$$\frac{t_1 \Downarrow \mathtt{true} \qquad t_2 \Downarrow v_2 \qquad t_3 \Downarrow v_3}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \Downarrow v_2} \quad \text{(B-IfTrue*)}$$

$$\frac{t_1 \Downarrow \mathtt{false} \qquad t_2 \Downarrow v_2 \qquad t_3 \Downarrow v_3}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \Downarrow v_3} \quad \text{(B-IfFalse*)}$$

The new rules **B-IfTrue*** and **B-IfFalse*** are wasteful: they do more evaluation than required to produce the final value. **B-IfTrue*** and **B-IfFalse*** are introduced only for purposes of making a comparison between **typing derivations** and **evaluation trees**.

Digression

Evaluation tree, according to adjusted big-step operational semantics, for the term $\mathtt{if}\ (\mathtt{iszero}\,0)\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{pred}\,0)$ from pages 4-5:

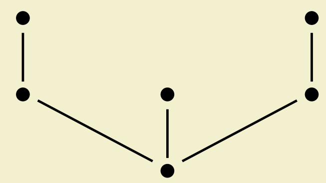
$$\dfrac{\dfrac{\dfrac{}{0 \Downarrow 0}\,\text{B-Value}}{\mathtt{iszero}\,0 \Downarrow \mathtt{true}}\,\text{B-IsZeroZero} \qquad \dfrac{}{0 \Downarrow 0}\,\text{B-Value} \qquad \dfrac{\dfrac{}{0 \Downarrow 0}\,\text{B-Value}}{\mathtt{pred}\,0 \Downarrow 0}\,\text{B-PredZero}}{\mathtt{if}\ (\mathtt{iszero}\,0)\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{pred}\,0) \Downarrow 0}\,\text{B-IfTrue*}$$

Digression

Evaluation tree, according to adjusted big-step operational semantics, for the term $\mathtt{if}\ (\mathtt{iszero}\,0)\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{pred}\,0)$ from pages 4-5:

$$\dfrac{\dfrac{\dfrac{}{0 \Downarrow 0}\,\text{B-Value}}{\mathtt{iszero}\,0 \Downarrow \mathtt{true}}\,\text{B-IsZeroZero} \qquad \dfrac{}{0 \Downarrow 0}\,\text{B-Value} \qquad \dfrac{\dfrac{}{0 \Downarrow 0}\,\text{B-Value}}{\mathtt{pred}\,0 \Downarrow 0}\,\text{B-PredZero}}{\mathtt{if}\ (\mathtt{iszero}\,0)\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{pred}\,0) \Downarrow 0}\,\text{B-IfTrue*}$$

The **evaluation tree** (this page), the **typing derivation** and the **abstract syntax tree** (page 5), all have the same shape:



Safety = Progress + Preservation [Pierce, Section 8.3]

Safety of a type system is also called its **Soundness**.

Progress: *A well-typed term is not stuck, i.e., it is a value or it can take a step according to the evaluation rules.* [Pierce, Theorem 8.3.2]

Preservation: *If a well-typed term takes a step of evaluation, then the resulting term is also well-typed.* [Pierce, Theorem 8.3.3]