Chapter 8: Typed Arithmetic Expressions

Types

The Typing Relation

Safety = Progress + Preservation

Reall: Syntax and Semantics

t ::=

- true
- false
- if t then t else t
- 0
- succ t
- pred t
- iszero t

Evaluation $\boxed{t \longrightarrow t'}$

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad \text{(E-If)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{succ } t_1 \longrightarrow \texttt{succ } t_1'} \qquad \text{(E-Succ)}$$

$$\texttt{pred } 0 \longrightarrow 0 \qquad \text{(E-PredZero)}$$

$$\texttt{pred (succ } nv_1\texttt{)} \longrightarrow nv_1 \qquad \text{(E-PredSucc)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{pred } t_1 \longrightarrow \texttt{pred } t_1'} \qquad \text{(E-Pred)}$$

$$\texttt{iszero } 0 \longrightarrow \texttt{true} \qquad \text{(E-IszeroZero)}$$

$$\texttt{iszero (succ } nv_1\texttt{)} \longrightarrow \texttt{false} \qquad \text{(E-IszeroSucc)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{iszero } t_1 \longrightarrow \texttt{iszero } t_1'} \qquad \text{(E-IsZero)}$$

Evaluation Results

- Values

v	::=		*values:*
		true	*true value*
		false	*false value*
		nv	*numeric value*
nv	::=		*numeric values:*
		0	*zero value*
		succ nv	*successor value*

- Get stuck (i.e., pred false)

Types of Terms

- Can we tell, without actually evaluating a term, that the term evaluation will not get stuck?

- Distinguish two types of terms:
  - Nat: terms whose results will be a numeric value
  - Bool: terms whose results will be a Boolean value

- "a term t has type T" means that t "obviously" (statically) evaluates to a value of T
  - if true then false else true has type Bool
  - pred (succ (pred (succ 0))) has type Nat

The Typing Relation: t : T

Typing Rule for Booleans

New syntactic forms

`T`	::=		*types:*
		`Bool`	*type of booleans*

New typing rules $\boxed{\mathtt{t} : \mathtt{T}}$

$$\mathtt{true} : \mathtt{Bool} \qquad \text{(T-True)}$$

$$\mathtt{false} : \mathtt{Bool} \qquad \text{(T-False)}$$

$$\frac{\mathtt{t}_1 : \mathtt{Bool} \qquad \mathtt{t}_2 : \mathtt{T} \qquad \mathtt{t}_3 : \mathtt{T}}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{T}} \qquad \text{(T-If)}$$

Typing Rules for Numbers

New syntactic forms

T	::=	...	*types:*
		Nat	*type of natural numbers*

New typing rules $\boxed{\texttt{t} : \texttt{T}}$

$$\texttt{0} : \texttt{Nat} \qquad \text{(T-ZERO)}$$

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{succ } \texttt{t}_1 : \texttt{Nat}} \qquad \text{(T-SUCC)}$$

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{pred } \texttt{t}_1 : \texttt{Nat}} \qquad \text{(T-PRED)}$$

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{iszero } \texttt{t}_1 : \texttt{Bool}} \qquad \text{(T-ISZERO)}$$

Typing Relation: Formal Definition

- **Definition**: the typing relation for arithmetic expressions is the smallest binary relation between terms and types satisfying all instances of the typing rules.

- A term t is typable (or well typed) if there is some T such that t : T.

Inversion Lemma (Generation Lemma)

- Given a valid typing statement, it shows
  - how a proof of this statement could have been generated;
  - a recursive algorithm for calculating the types of terms.

LEMMA [INVERSION OF THE TYPING RELATION]:

1. If $\mathtt{true} : \mathsf{R}$, then $\mathsf{R} = \mathtt{Bool}$.
2. If $\mathtt{false} : \mathsf{R}$, then $\mathsf{R} = \mathtt{Bool}$.
3. If $\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathsf{R}$, then $\mathtt{t}_1 : \mathtt{Bool}$, $\mathtt{t}_2 : \mathsf{R}$, and $\mathtt{t}_3 : \mathsf{R}$.
4. If $0 : \mathsf{R}$, then $\mathsf{R} = \mathtt{Nat}$.
5. If $\mathtt{succ}\ \mathtt{t}_1 : \mathsf{R}$, then $\mathsf{R} = \mathtt{Nat}$ and $\mathtt{t}_1 : \mathtt{Nat}$.
6. If $\mathtt{pred}\ \mathtt{t}_1 : \mathsf{R}$, then $\mathsf{R} = \mathtt{Nat}$ and $\mathtt{t}_1 : \mathtt{Nat}$.
7. If $\mathtt{iszero}\ \mathtt{t}_1 : \mathsf{R}$, then $\mathsf{R} = \mathtt{Bool}$ and $\mathtt{t}_1 : \mathtt{Nat}$.

Typing Derivation

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

Statements are formal assertions about the typing of programs.
Typing rules are implications between statements

Derivations are deductions based on typing rules.

Uniqueness of Types

- **Theorem** [Uniqueness of Types]: Each term t has at most one type. That is, if t is typable, then its type is unique.

- Note: later on, we may have a type system where a term may have many types.

Safety = Progress + Preservation

Safety (Soundness)

- By safety, it means well-typed terms do not “go wrong”.

- By “go wrong”, it means reaching a “stuck state” that is not a final value but where the evaluation rules do not tell what to do next.

Safety = Progress + Preservation

Well-typed terms do not get stuck

- Progress: A well-typed term is not stuck (either it is a value or it can take a step according to the evaluation rules).
- Preservation: If a well-typed term takes a step of evaluation, then the resulting term is also well typed.

Progress

- **Theorem** [Progress]: Suppose t is a well-typed term (that is, t : T for some T). Then either t is a value or else there is some t′ with t → t′.

Proof: By induction on a derivation of t : T.

- case T-True: true : Bool   OK?

- case T-If:

$$\frac{t1 : Bool,\ t2 : T,\ t3 : T}{\text{if } t1 \text{ then } t2 \text{ else } t3 : T} \quad \text{OK?}$$

- …

Preservation

- **Theorem** [Preservation]:

If t : T and t → t′, then t′ : T.

Proof: By induction on a derivation of t : T.

- case T-True: true : Bool OK?

- case T-If:

$$\frac{t1 : Bool,\ t2 : T,\ t3 : T}{\text{if } t1 \text{ then } t2 \text{ else } t3 : T} \quad \text{OK?}$$

- ...

Note: The preservation theorem is often called subject reduction property (or subject evaluation property)

Homework

- Read Chapter 8.
- Do Exercises 8.3.7

8.3.7 EXERCISE [RECOMMENDED, ★★]: Suppose our evaluation relation is defined in the big-step style, as in Exercise 3.5.17. How should the intuitive property of type safety be formalized? □