Untyped Arithmetic Expressions

Principles of Programming Languages

CSE 526

Compiled at 13:45 on 2018/02/15

Formal Description of Programming Languages

- Formal Definition of Syntax
  - Grammars to define the set of *strings* that define a syntactically valid program
  - Inductive definitions of *abstract syntax trees*.
- Formal Definition of Semantics
  - Structural operational semantics

Syntax

Example: A language of untyped arithmetic expressions

$$
\begin{array}{lcl}
t & ::= & \texttt{true} \\
 & | & \texttt{false} \\
 & | & \texttt{if}(t, t, t) \\
 & | & \texttt{0} \\
 & | & \texttt{succ}\ \ t \\
 & | & \texttt{pred}\ \ t \\
 & | & \texttt{iszero}\ \ t
\end{array}
$$

Inductive Definition: The set $\mathcal{T}$ of *terms* is the smallest set such that:

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$
2. if $t_1 \in \mathcal{T}$ then $\{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1\} \subseteq \mathcal{T}$.
3. if $t_1, t_2, t_3 \in \mathcal{T}$ then $\texttt{if}(t_1, t_2, t_3) \in \mathcal{T}$.

Alternative Definitions of Terms

Inductive Definition: The set $\mathcal{T}$ of *terms* is the smallest set such that:

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$
2. if $t_1 \in \mathcal{T}$ then $\{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1\} \subseteq \mathcal{T}$.
3. if $t_1, t_2, t_3 \in \mathcal{T}$ then $\texttt{if}(t_1, t_2, t_3) \in \mathcal{T}$.

Inference Rules: The set $\mathcal{T}$ is defined by the following rules:

$$\texttt{true} \in \mathcal{T} \qquad \texttt{false} \in \mathcal{T} \qquad \texttt{0} \in \mathcal{T}$$

$$\frac{t_1 \in \mathcal{T}}{\texttt{succ}\ t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\texttt{pred}\ t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\texttt{iszero}\ t_1 \in \mathcal{T}}$$

$$\frac{t_1, t_2, t_3 \in \mathcal{T}}{\texttt{if}(t_1, t_2, t_3) \in \mathcal{T}}$$

Alternative Definitions of Terms (contd.)

Inductive Definition: The set $\mathcal{T}$ of *terms* is the smallest set such that:

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$
2. if $t_1 \in \mathcal{T}$ then $\{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1\} \subseteq \mathcal{T}$.
3. if $t_1, t_2, t_3 \in \mathcal{T}$ then $\texttt{if}(t_1, t_2, t_3) \in \mathcal{T}$.

Constructive Definition: For each natural number i define set S_i as follows:

$$
\begin{array}{rcl}
S_0 & = & \emptyset \\
S_{i+1} & = & \left\{ \begin{array}{ll} & \{\texttt{true}, \texttt{false}, \texttt{0}\} \\ \cup & \{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1 \mid t_1 \in S_i\} \\ \cup & \{\texttt{if}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in S_i\} \end{array} \right.
\end{array}
$$

$\mathcal{S} = \bigcup_i S_i$

Alternative Definitions of Terms (contd.)

Properties:

- The sets S_i are cumulative, i.e., $\forall i\ S_i \subseteq S_{i+1}$
- $\mathcal{T} = \mathcal{S}$
  1. $\mathcal{S}$ satisfies the conditions on $\mathcal{T}$
  2. Let $\mathcal{S}'$ be a set that satisfies the conditions on $\mathcal{T}$. Then $\mathcal{S} \subseteq \mathcal{S}'$.

Equivalence of $\mathcal{S}$ and $\mathcal{T}$

$$S_0 = \emptyset$$
$$S_{i+1} = \begin{cases} & \{\texttt{true}, \texttt{false}, \texttt{0}\} \\ \cup & \{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1 \mid t_1 \in S_i\} \\ \cup & \{\texttt{if}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in S_i\} \end{cases}$$

1. $\forall i\ S_i \subseteq S_{i+1}$

Proof is by *ordinary* induction on i: $P(0)$ and $\forall k.P(k) \implies P(k+1)$, where

$P(i) : S_i \subseteq S_{i+1}$

$P(0)$: S_0 is empty, and hence is a subset of S_1.

$P(k) \implies P(k+1)$: We'll show that every $t \in S_{k+1}$ is also $\in S_{k+2}$.

Consider $t \in S_{k+1}$. Then t is of one of the following forms:

1. $t \in \{\texttt{true}, \texttt{false}, \texttt{0}\}$. Then $t \in S_{k+2}$ by definition.
2. $t = \texttt{succ}(t_1)$ for some $t_1 \in S_k$. By ind. hyp., $t_1 \in S_{k+1}$ and hence $t \in S_{k+2}$.

3–5. proof steps for terms of the form $\texttt{pred}(t_1)$ etc. are similar to case 2.

Equivalence of $\mathcal{S}$ and $\mathcal{T}$

$\mathcal{T}$ is the smallest set such that 1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$ 2. if $t_1 \in \mathcal{T}$ then $\{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1\} \subseteq \mathcal{T}$. 3. if $t_1, t_2, t_3 \in \mathcal{T}$ then $\texttt{if}(t_1, t_2, t_3) \in \mathcal{T}$.	$S_0 = \emptyset$ $S_{i+1} = \begin{cases} & \{\texttt{true}, \texttt{false}, \texttt{0}\} \\ \cup & \{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1 \mid t_1 \in S_i\} \\ \cup & \{\texttt{if}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in S_i\} \end{cases}$ $\mathcal{S} = \bigcup_{i \geq 0} S_i$

2a. $\mathcal{S}$ satisfies the conditions 1, 2, and 3 on $\mathcal{T}$

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\}$ are in S_1 and hence in $\mathcal{S}$.
2. If $t_1 \in \mathcal{S}$ then $t_1 \in S_k$ for some $k \geq 0$. Hence, $\{\texttt{succ}(t_1), \texttt{pred}(t_1), \texttt{iszero}(t_1)\} \subseteq S_{k+1}$ and consequently $\subseteq \mathcal{S}$.
3. If $t_1, t_2, t_3 \in \mathcal{S}$ then there are k_1, k_2, k_3 such that $t_1 \in S_{k_1}$, $t_2 \in S_{k_2}$, and $t_3 \in S_{k_3}$. From Lemma 1, t_1, t_2, t_3 are all $\in S_k$ for $k \geq \max(k_1, k_2, k_3)$. Hence $\texttt{if}(t_1, t_2, t_3) \in S_{k+1}$ and consequently $\in \mathcal{S}$.

Equivalence of $\mathcal{S}$ and $\mathcal{T}$ (Contd.)

$\mathcal{T}$ is the smallest set such that	$S_0 = \emptyset$
1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$ 2. if $t_1 \in \mathcal{T}$ then $\{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1\} \subseteq \mathcal{T}$. 3. if $t_1, t_2, t_3 \in \mathcal{T}$ then $\texttt{if}(t_1, t_2, t_3) \in \mathcal{T}$.	$S_{i+1} = \begin{cases} & \{\texttt{true}, \texttt{false}, \texttt{0}\} \\ \cup & \{\texttt{succ}\ t_1, \texttt{pred}\ t_1, \texttt{iszero}\ t_1 \mid t_1 \in S_i\} \\ \cup & \{\texttt{if}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in S_i\} \end{cases}$ $\mathcal{S} = \bigcup_{i \geq 0} S_i$

2b. If $\mathcal{S}'$ satisfies the conditions 1, 2, and 3 on $\mathcal{T}$, then $\mathcal{S} \subseteq \mathcal{S}'$

We will show this by proving that S_i is a subset of $\mathcal{S}'$ by *complete* induction: $(\forall j < i\ P(j)) \implies P(i)$.

We get two cases from the definition of S_i:

$i = 0$: $S_0 \subseteq \mathcal{S}'$.

$\exists j.\ i = j + 1$: Every $t \in S_{j+1}$ is also $\in \mathcal{S}'$.

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\}$ are in $\mathcal{S}'$ by condition 1.
2. If $t = \texttt{succ}(t_1) \in S_{j+1}$ for $t_1 \in S_j$, then by ind. hyp. $t_1 \in \mathcal{S}'$, and $t \in \mathcal{S}'$ by condition 2.

3–5. Proof steps are similar to case 2 for $t = \texttt{pred}(t_1)$ etc.

Inductive Definitions

The following recursive definition is "well-defined" since the function on a term is defined based on that on *smaller* terms.

$$
\begin{array}{rcl}
Const(\texttt{true}) & = & \{\texttt{true}\} \\
Const(\texttt{false}) & = & \{\texttt{false}\} \\
Const(\texttt{0}) & = & \{\texttt{0}\} \\
Const(\texttt{succ}\ t_1) & = & Const(t_1) \\
Const(\texttt{pred}\ t_1) & = & Const(t_1) \\
Const(\texttt{iszero}\ t_1) & = & Const(t_1) \\
Const(\texttt{if}(t_1, t_2, t_3)) & = & Const(t_1) \cup Const(t_2) \cup Const(t_3)
\end{array}
$$

Inductive Definitions (contd.)

The size of a term is also defined inductively:

$$
\begin{aligned}
size(\texttt{true}) &= 1 \\
size(\texttt{false}) &= 1 \\
size(\texttt{0}) &= 1 \\
size(\texttt{succ}\ t_1) &= size(t_1) + 1 \\
size(\texttt{pred}\ t_1) &= size(t_1) + 1 \\
size(\texttt{iszero}\ t_1) &= size(t_1) + 1 \\
size(\texttt{if}(t_1, t_2, t_3)) &= size(t_1) + size(t_2) + size(t_3) + 1
\end{aligned}
$$

The *depth* of terms can be defined similarly.

Induction on terms

- **Structural Induction:**

  If, for each term s,
  given $P(r)$ for all immediate sub-terms r of s
  we can show $P(s)$
  then $P(s)$ holds for all s.

- **Induction on size:**

  If, for each term s,
  given $P(r)$ for all terms r such that $size(r) < size(s)$
  we can show $P(s)$
  then $P(s)$ holds for all s.

Operational Semantics

Example: A language of untyped boolean expressions $\mathcal{B}$:

t	::=	true | false | if(t, t, t)	Terms
v	::=	true | false	Values

Evaluation:

$$\texttt{if}(\texttt{true}, t_2, t_3) \to t_2 \qquad \text{E-IfTrue}$$

$$\texttt{if}(\texttt{false}, t_2, t_3) \to t_3 \qquad \text{E-IfFalse}$$

$$\frac{t_1 \to t_1'}{\texttt{if}(t_1, t_2, t_3) \to \texttt{if}(t_1', t_2, t_3)} \qquad \text{E-If}$$

The Inference Rule Notation

$$\frac{\textit{Premises}}{\textit{Conclusion}} \qquad \text{Name}$$

- Inference rules without premises are called *axioms*.
- Inference rules (more precisely *rule schema*) may have meta-variables. E.g., t_1, t_2, t_3, t_1' in:

$$\frac{t_1 \to t_1'}{\texttt{if}(t_1, t_2, t_3) \to \texttt{if}(t_1', t_2, t_3)} \qquad \text{E-If}$$

A *rule instance* is obtained by consistently replacing each meta-variable by the same term in the premises as well as the conclusion.

Operational Semantics of Boolean Expressions

$$\texttt{if}(\texttt{true}, t_2, t_3) \rightarrow t_2 \qquad \text{E-IfTrue}$$

$$\texttt{if}(\texttt{false}, t_2, t_3) \rightarrow t_3 \qquad \text{E-IfFalse}$$

$$\frac{t_1 \rightarrow t_1'}{\texttt{if}(t_1, t_2, t_3) \rightarrow \texttt{if}(t_1', t_2, t_3)} \qquad \text{E-If}$$

- The *one-step evaluation relation* is the smallest relation "$\rightarrow$" on terms satisfying the above rules.
- When (t, t') is in the evaluation relation, we say that

  the evaluation statement (or judgment) $t \rightarrow t'$ is derivable.

- **Determinacy:** If $t \rightarrow t'$ and $t \rightarrow t''$ then $t' = t''$.

Example Evaluations

Step 1:

$$\frac{\dfrac{}{\texttt{if}(\texttt{true}, \texttt{false}, \texttt{true}) \rightarrow \texttt{false}}\ \text{E-IfTrue}}{\texttt{if}(\texttt{if}(\texttt{true}, \texttt{false}, \texttt{true}), \texttt{true}, \texttt{false}) \rightarrow \texttt{if}(\texttt{false}, \texttt{true}, \texttt{false})} \qquad \text{E-If}$$

Step 2:

$$\frac{}{\texttt{if}(\texttt{false}, \texttt{true}, \texttt{false}) \rightarrow \texttt{false}} \qquad \text{E-IfFalse}$$

$$\begin{aligned} &\texttt{if}(\texttt{if}(\texttt{true}, \texttt{false}, \texttt{true}), \texttt{true}, \texttt{false}) \\ &\quad \rightarrow \texttt{if}(\texttt{false}, \texttt{true}, \texttt{false}) \\ &\quad \rightarrow \texttt{false} \end{aligned}$$

Small-Step Semantics

Properties of (previously defined) operational semantics

- **Determinacy:** If $t \to t'$ and $t \to t''$ then $t' = t''$.
- Proof: by induction on the derivation of $t \to t'$.
- This proof is also identical to induction on structure of t
- The operational semantics defined previously is said to be "*Structural Operational Semantics* (SOS)", where the evaluation derivation follows the structure of the term being reduced.

Proof of Determinacy

$\texttt{if}(\texttt{true}, t_2, t_3) \to t_2$	E-IfTrue	$\dfrac{t_1 \to t_1'}{\texttt{if}(t_1, t_2, t_3) \to \texttt{if}(t_1', t_2, t_3)}$	E-If
$\texttt{if}(\texttt{false}, t_2, t_3) \to t_3$	E-IfFalse		

If $t \to t'$ and $t \to t''$ then $t' = t''$.

Consider the last rule used in the derivation of $t \to t'$.

- E-IfTrue: Then $t = \texttt{if}(\texttt{true}, t_2, t_3)$ for some terms t_2 and t_3, and $t' = t_2$. Consider the derivation $t \to t''$. The last rule used here cannot be E-IfFalse (does not match) or E-If (premise does not hold). Hence the last rule used in $t \to t''$ must be E-IfTrue, and $t'' = t_2 = t'$.
- E-IfFalse: Similar to above case.
- E-If: Then $t = \texttt{if}(t_1, t_2, t_3)$, $t_1 \to t_1'$ (premise), and $t' = \texttt{if}(t_1', t_2, t_3)$. The last rule used in derivation of $t \to t''$ must also be E-If, with premise $t_1 \to t_1''$, resulting in $t'' = \texttt{if}(t_1'', t_2, t_3)$. By induction hypotheses, we know $t_1' = t_1''$. Consequently, $t' = t''$.

Normal Form

- A term t is in normal form if there is no t' such that $t \to t'$.
- Every *value* is in normal form.
- If t is in normal form, then t is a value.
- Let "$\to^*$" relation be the reflexive, transitive closure of "$\to$" relation in the following:
- **Uniqueness:** If $t \to^* u$ and $t \to^* u'$ where u and u' are normal forms, then $u = u'$.
  If $t \to^* u$, and u is in normal form, we say u is the normal form of t.
- **Termination:** For every term t, there is some normal form t' such that $t \to^* t'$.

Untyped Arithmetic Expressions

$$
\begin{array}{rcll}
t & ::= & \texttt{true} & \text{Terms} \\
 & | & \texttt{false} & \\
 & | & \texttt{if}(t, t, t) & \\
 & | & \texttt{0} & \\
 & | & \texttt{succ}\ \ t & \\
 & | & \texttt{pred}\ \ t & \\
 & | & \texttt{iszero}\ \ t & \\
v & ::= & \texttt{true} \mid \texttt{false} \mid nv & \text{Values} \\
nv & ::= & \texttt{0} \mid \texttt{succ}\ nv & \text{Numeric Values}
\end{array}
$$

Operational Semantics of Untyped Arithmetic Expressions

$$\frac{t_1 \to t_1'}{\texttt{succ}\ t_1 \to \texttt{succ}\ t_1'} \quad \text{E-Succ}$$

$$\texttt{pred 0} \to \texttt{0} \quad \text{E-PredZero}$$

$$\texttt{pred succ}\ nv_1 \to nv_1 \quad \text{E-PredSucc}$$

$$\frac{t_1 \to t_1'}{\texttt{pred}\ t_1 \to \texttt{pred}\ t_1'} \quad \text{E-Pred}$$

$$\texttt{iszero 0} \to \texttt{true} \quad \text{E-IsZeroZero}$$

$$\texttt{iszero succ}\ nv_1 \to \texttt{false} \quad \text{E-IsZeroSucc}$$

$$\frac{t_1 \to t_1'}{\texttt{iszero}\ t_1 \to \texttt{iszero}\ t_1'} \quad \text{E-IsZero}$$

Properties of the operational semantics (prev. slide)

- **Determinacy**
- **Uniqueness of normal forms**
- **Termination**
- Not all normal forms are values!

  A term is stuck if it is in normal form but not a value.

  Stuck terms correspond to “run-time errors”.

Reflexive Transitive Closure

Let $R \subseteq D \times D$ be a binary relation. The reflexive transitive closure R^* of R is the smallest relation such that

- $\forall d \in D \quad (d, d) \in R^*$
- $R \subseteq R^*$
- $\forall x, y, z \in D \quad (x, y) \in R^* \wedge (y, z) \in R^* \Rightarrow (x, z) \in R^*$

Inference rules for $\rightarrow$, the small-step transition relation (Ex. 3.5.10):

$$t \rightarrow^* t$$

$$\frac{t \rightarrow t'}{t \rightarrow^* t'}$$

$$\frac{t \rightarrow^* t' \quad t' \rightarrow^* t''}{t \rightarrow^* t''}$$

Big-Step Semantics

Small-Step Semantics for **B**:

$$\texttt{if}(\texttt{true}, t_2, t_3) \rightarrow t_2 \qquad \text{E-IfTrue}$$

$$\texttt{if}(\texttt{false}, t_2, t_3) \rightarrow t_3 \qquad \text{E-IfFalse}$$

$$\frac{t_1 \rightarrow t_1'}{\texttt{if}(t_1, t_2, t_3) \rightarrow \texttt{if}(t_1', t_2, t_3)} \qquad \text{E-If}$$

$\forall \boldsymbol{t}\ \exists \boldsymbol{v}.\ \boldsymbol{t} \rightarrow^* \boldsymbol{v}$ (Uniqueness of N.F. & Termination)

Big-Step Semantics for **B**:

$$v \Downarrow v \qquad \text{B-Value}$$

$$\frac{t_1 \Downarrow \texttt{true} \quad t_2 \Downarrow v_2}{\texttt{if}(t_1, t_2, t_3) \Downarrow v_2} \qquad \text{B-IfTrue}$$

$$\frac{t_1 \Downarrow \texttt{false} \quad t_3 \Downarrow v_3}{\texttt{if}(t_1, t_2, t_3) \Downarrow v_3} \qquad \text{B-IfFalse}$$

Soundness of Big-Step Semantics

If $t \Downarrow v$ then $t \longrightarrow^* v$

Proof: by induction on derivation of $t \Downarrow v$:

$$\dfrac{\dfrac{\cdots}{\vdots}}{\dfrac{\cdots}{t \Downarrow v}}$$

} Case-split on the last step

1. B-Value: $t = v$, *trivial*
2. B-IfTrue: $t = \texttt{if}(t_1, t_2, t_3)$, $t_1 \Downarrow \texttt{true}$, $t_2 \Downarrow v$
   - By induction hypothesis, we know $t_1 \longrightarrow^* \texttt{true}$, and $t_2 \longrightarrow^* v$.
   - From evaluation sequence $t_1 \longrightarrow^* \texttt{true}$, we can construct an evaluation sequence $\texttt{if}(t_1, t_2, t_3) \longrightarrow^* \texttt{if}(\texttt{true}, t_2, t_3)$

     Stated as a Lemma and proved separately
   - From E-IfTrue we have $\texttt{if}(\texttt{true}, t_2, t_3) \longrightarrow t_2$
   - Hence we get the following evaluation sequence:
     $\texttt{if}(t_1, t_2, t_3) \longrightarrow^* \texttt{if}(\texttt{true}, t_2, t_3) \longrightarrow t_2 \longrightarrow^* v$
3. B-IfFalse (similar to the above case)

Soundness of Big-Step Semantics (contd.)

Lemma needed for soundness proof:

If $t_1 \longrightarrow^* t_1'$ then $\texttt{if}(t_1, t_2, t_3) \longrightarrow^* \texttt{if}(t_1', t_2, t_3)$

Proof: by induction on the length of evaluation sequence $t_1 \longrightarrow^* t_1'$.

- $t_1 = t_1'$ (i.e zero-length evaluation sequence): trivial.
- $t_1 \longrightarrow \hat{t}_1 \longrightarrow^* t_1'$: Then t_1 is not a value (by defn of small-step semantics)
  - By E-If, $\texttt{if}(t_1, t_2, t_3) \longrightarrow \texttt{if}(\hat{t}_1, t_2, t_3)$
  - By induction hypothesis, $\hat{t}_1 \longrightarrow^* t_1'$ means $\texttt{if}(\hat{t}_1, t_2, t_3) \longrightarrow^* \texttt{if}(t_1', t_2, t_3)$
  - Hence $\texttt{if}(t_1, t_2, t_3) \longrightarrow \texttt{if}(\hat{t}_1, t_2, t_3) \longrightarrow^* \texttt{if}(t_1', t_2, t_3)$

Completeness of Big-Step Semantics

If $t \rightarrow^* v$ then $t \Downarrow v$

Proof: by induction on length of evaluation of $t \rightarrow^* v$

1. $t = v$: $v \Downarrow v$ by B-VALUE.
2. $t \rightarrow \hat{t} \rightarrow^* v$: then $t = \texttt{if}(t_1, t_2, t_3)$.

   Use the following lemma:

   > If $\texttt{if}(t_1, t_2, t_3) \rightarrow^* v$ then
   > $t_1 \rightarrow^* \texttt{true}$ and $t_2 \rightarrow^* v$, or
   > $t_1 \rightarrow^* \texttt{false}$ and $t_3 \rightarrow^* v$
   > and the evaluation sequences for t_1 and t_2 or t_3 are strictly shorter than the given evaluation sequence.

   If $t_1 \rightarrow^* \texttt{true}$, then by induction hypothesis, $t_1 \Downarrow \texttt{true}$ and $t_2 \Downarrow v$

   Hence by applying B-IFTRUE, we get $\texttt{if}(t_1, t_2, t_3) \Downarrow v$.

   Proof if $t_1 \rightarrow^* \texttt{false}$ is similar.