CIS 500

Software Foundations

Fall 2003

17 September

Administrivia

- Reading for (before!) next week's lectures: TAPL Chapter 5

Review (and a few more details)

Simple Arithmetic Expressions

The set $\mathcal{T}$ of terms is defined by the following abstract grammar:

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

Inference Rule Notation

More explicitly: The set $\mathcal{T}$ is the smallest set closed under the following rules.

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

$$\frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{succ t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{pred t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{iszero t}_1 \in \mathcal{T}}$$

$$\frac{\texttt{t}_1 \in \mathcal{T} \qquad \texttt{t}_\in \in \mathcal{T} \qquad \texttt{t}_\ni \in \mathcal{T}}{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}}$$

Generating Functions

Each of these rules can be thought of as a generating function that, given some elements from $\mathcal{T}$, generates some other element of $\mathcal{T}$. Saying that $\mathcal{T}$ is closed under these rules means that $\mathcal{T}$ cannot be made any bigger using these generating functions — it already contains everything "justified by its members."

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

$$\frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{succ t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{pred t}_1 \in \mathcal{T}} \qquad \frac{\texttt{t}_1 \in \mathcal{T}}{\texttt{iszero t}_1 \in \mathcal{T}}$$

$$\frac{\texttt{t}_1 \in \mathcal{T} \qquad \texttt{t}_\in \in \mathcal{T} \qquad \texttt{t}_\ni \in \mathcal{T}}{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}}$$

Let's write these generating functions explicitly.

$$
\begin{aligned}
F_1(\mathcal{U}) &= \{\texttt{true}\} \\
F_2(\mathcal{U}) &= \{\texttt{false}\} \\
F_3(\mathcal{U}) &= \{\texttt{0}\} \\
F_4(\mathcal{U}) &= \{\texttt{succ } \texttt{t}_1 \mid \texttt{t}_1 \in \mathcal{U}\} \\
F_5(\mathcal{U}) &= \{\texttt{pred } \texttt{t}_1 \mid \texttt{t}_1 \in \mathcal{U}\} \\
F_6(\mathcal{U}) &= \{\texttt{iszero } \texttt{t}_1 \mid \texttt{t}_1 \in \mathcal{U}\} \\
F_7(\mathcal{U}) &= \{\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \mid \texttt{t}_1, \texttt{t}_2, \texttt{t}_3 \in \mathcal{U}\}
\end{aligned}
$$

Each one takes a set of terms $\mathcal{U}$ as input and produces a set of "terms justified by $\mathcal{U}$" as output.

If we now define a generating function for the whole set of inference rules (by combining the generating functions for the individual rules),

$$F(U) = F_1(U) \cup F_2(U) \cup F_3(U) \cup F_4(U) \cup F_5(U) \cup F_6(U) \cup F_7(U)$$

then we can restate the previous definition of the set of terms $\mathcal{T}$ like this:

Definition:

- A set U is said to be "closed under F" (or "F-closed") if $F(U) \subseteq U$.
- The set of terms $\mathcal{T}$ is the smallest F-closed set.
  (I.e., if O is another set such that $F(O) \subseteq O$, then $\mathcal{T} \subseteq \mathcal{O}$.)

Another definition by generating functions

Our alternate definition of the set of terms can also be stated using the generating function F:

$$
\begin{aligned}
\mathcal{S}_0 &= \emptyset \\
\mathcal{S}_{i+1} &= \mathsf{F}(\mathcal{S}_i) \\
\\
\mathcal{S} &= \bigcup_i \mathcal{S}_i
\end{aligned}
$$

Compare this definition of $\mathcal{S}$ with the one we saw last time:

$$
\begin{aligned}
\mathcal{S}_0 &= \emptyset \\
\mathcal{S}_{i+1} &= \{\mathtt{true}, \mathtt{false}, 0\} \\
&\quad \cup \{\mathtt{succ}\ \mathtt{t}_1, \mathtt{pred}\ \mathtt{t}_1, \mathtt{iszero}\ \mathtt{t}_1 \mid \mathtt{t}_1 \in \mathcal{S}_i\} \\
&\quad \cup \{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \mid \mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_3 \in \mathcal{S}_i\}
\end{aligned}
$$

$$
\mathcal{S} = \bigcup_i \mathcal{S}_i
$$

The only difference is that we have “pulled out” F and given it a name.

Note that our two definitions of terms characterize the same set from different directions:

- "from above," as the intersection of all F-closed sets;
- "from below," as the limit (union) of a series of sets that start from $\emptyset$ and get "closer and closer to being F-closed."

Proposition 3.2.6 in the book shows that these two definitions actually define the same set.

Warning: Hard hats on for the next slide!

Structural Induction

The principle of structural induction on terms can also be re-stated using generating functions:

Suppose T is the smallest F-closed set.

If, for each set U,
from the assumption "$P(u)$ holds for every $u \in U$"
we can show "$P(v)$ holds for any $v \in F(U)$,"
then $P(t)$ holds for all $t \in T$.

Structural Induction

The principle of structural induction on terms can also be re-stated using generating functions:

Suppose T is the smallest F-closed set.

If, for each set U,
from the assumption "$P(u)$ holds for every $u \in U$"
we can show "$P(v)$ holds for any $v \in F(U)$,"
then $P(t)$ holds for all $t \in T$.

Why?

Structural Induction

The principle of structural induction on terms can also be re-stated using generating functions:

> Suppose T is the smallest F-closed set.
>
> If, for each set U,
>
> > from the assumption "$P(u)$ holds for every $u \in U$"
> >
> > we can show "$P(v)$ holds for any $v \in F(U)$,"
>
> then $P(t)$ holds for all $t \in T$.

Why? Because we assumed that T was the smallest F-closed set, i.e., that $T \subseteq O$ for any other F-closed set O. But showing

> for each set U,
>
> > given $P(u)$ for all $u \in U$
> >
> > we can show $P(v)$ for all $v \in F(U)$

amounts to showing that O = "the set of all terms satisfying P" is itself an F-closed set, i.e. $T \subseteq O$, i.e., every element of T satisfies P.

Structural Induction

Compare this with the structural induction principle for terms from last lecture:

> If, for each term $\mathtt{s}$,
> given $\mathbf{P}(\mathtt{r})$ for all immediate subterms $\mathtt{r}$ of $\mathtt{s}$
> we can show $\mathbf{P}(\mathtt{s})$,
> then $\mathbf{P}(\mathtt{t})$ holds for all $\mathtt{t}$.

Operational Semantics

Abstract Machines

An abstract machine consists of:

- a set of states
- a transition relation on states, written $\longrightarrow$

Operational semantics for Booleans

Syntax of terms and values

`t ::=`		terms
	`true`	constant true
	`false`	constant false
	`if t then t else t`	conditional
`v ::=`		values
	`true`	true value
	`false`	false value

The evaluation relation $\mathtt{t} \longrightarrow \mathtt{t}'$ is the smallest relation closed under the following rules:

$$\texttt{if true then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3 \longrightarrow \mathtt{t}_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3 \longrightarrow \mathtt{t}_3 \qquad \text{(E-IfFalse)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\texttt{if } \mathtt{t}_1 \texttt{ then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3 \longrightarrow \texttt{if } \mathtt{t}_1' \texttt{ then } \mathtt{t}_2 \texttt{ else } \mathtt{t}_3} \qquad \text{(E-If)}$$

Digression

Suppose we wanted to change our evaluation strategy so that the `then` and `else` branches of an `if` get evaluated (in that order) before the guard. How would we need to change the rules?

Digression

Suppose we wanted to change our evaluation strategy so that the `then` and `else` branches of an `if` get evaluated (in that order) before the guard. How would we need to change the rules?

Suppose, moreover, that if the evaluation of the `then` and `else` branches leads to the same value, we want to immediately produce that value ("short-circuiting" the evaluation of the guard). How would we need to change the rules?

Digression

Suppose we wanted to change our evaluation strategy so that the `then` and `else` branches of an `if` get evaluated (in that order) before the guard. How would we need to change the rules?

Suppose, moreover, that if the evaluation of the `then` and `else` branches leads to the same value, we want to immediately produce that value ("short-circuiting" the evaluation of the guard). How would we need to change the rules?

Of the rules we just invented, which are computation rules and which are congruence rules?

Evaluation, more explicitly

$\longrightarrow$ is the smallest two-place relation closed under the following rules:

$$((\texttt{if true then } t_2 \texttt{ else } t_3), t_2) \in \longrightarrow$$

$$((\texttt{if false then } t_2 \texttt{ else } t_3), t_3) \in \longrightarrow$$

$$\frac{(t_1, t_1') \in \longrightarrow}{((\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3), (\texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3)) \in \longrightarrow}$$

Even more explicitly...

What is the generating function corresponding to these rules?

(exercise)

Reasoning about Evaluation

Derivations

We can record the "justification" for a particular pair of terms that are in the evaluation relation in the form of a tree.

(on the board)

Terminology:

- These trees are called derivation trees (or just derivations)
- The final statement in a derivation is its conclusion
- We say that the derivation is a witness for its conclusion (or a proof of its conclusion) — it records all the reasoning steps that justify the conclusion.

Observation

Lemma: Suppose we are given a derivation tree $\mathcal{D}$ witnessing the pair $(\mathtt{t}, \mathtt{t}')$ in the evaluation relation. Then either

1. the final rule used in $\mathcal{D}$ is E-IfTrue and we have $\mathtt{t} = \mathtt{if\ true\ then\ t_2\ else\ t_3}$ and $\mathtt{t}' = \mathtt{t_2}$, for some $\mathtt{t_2}$ and $\mathtt{t_3}$, or
2. the final rule used in $\mathcal{D}$ is E-IfFalse and we have $\mathtt{t} = \mathtt{if\ false\ then\ t_2\ else\ t_3}$ and $\mathtt{t}' = \mathtt{t_3}$, for some $\mathtt{t_2}$ and $\mathtt{t_3}$, or
3. the final rule used in $\mathcal{D}$ is E-If and we have $\mathtt{t} = \mathtt{if\ t_1\ then\ t_2\ else\ t_3}$ and $\mathtt{t}' = \mathtt{if\ t_1'\ then\ t_2\ else\ t_3}$, for some $\mathtt{t_1}$, $\mathtt{t_1'}$, $\mathtt{t_2}$, and $\mathtt{t_3}$; moreover, the immediate subderivation of $\mathcal{D}$ witnesses $(\mathtt{t_1}, \mathtt{t_1'}) \in \longrightarrow$.

Induction on Derivations

We can now write proofs about evaluation "by induction on derivation trees."

Given an arbitrary derivation $\mathcal{D}$ with conclusion $\texttt{t} \longrightarrow \texttt{t}'$, we assume the desired result for its immediate sub-derivation (if any) and proceed by a case analysis (using the previous lemma) of the final evaluation rule used in constructing the derivation tree.

E.g....

Induction on Derivations — Example

Theorem: If $\mathtt{t} \longrightarrow \mathtt{t}'$ — i.e., if $(\mathtt{t}, \mathtt{t}') \in \longrightarrow$ — then $size(\mathtt{t}) > size(\mathtt{t}')$.

Proof: By induction on a derivation $\mathcal{D}$ of $\mathtt{t} \longrightarrow \mathtt{t}'$.

1. Suppose the final rule used in $\mathcal{D}$ is E-IfTrue, with $\mathtt{t} = \mathtt{if\ true\ then\ t_2\ else\ t_3}$ and $\mathtt{t}' = \mathtt{t_2}$. Then the result is immediate from the definition of size.
2. Suppose the final rule used in $\mathcal{D}$ is E-IfFalse, with $\mathtt{t} = \mathtt{if\ false\ then\ t_2\ else\ t_3}$ and $\mathtt{t}' = \mathtt{t_3}$. Then the result is again immediate from the definition of size.
3. Suppose the final rule used in $\mathcal{D}$ is E-If, with $\mathtt{t} = \mathtt{if\ t_1\ then\ t_2\ else\ t_3}$ and $\mathtt{t}' = \mathtt{if\ t_1'\ then\ t_2\ else\ t_3}$, where $(\mathtt{t_1}, \mathtt{t_1'}) \in \longrightarrow$ is witnessed by a derivation $\mathcal{D}_\infty$. By the induction hypothesis, $size(\mathtt{t_1}) > size(\mathtt{t_1'})$. But then, by the definition of size, we have $size(\mathtt{t}) > size(\mathtt{t}')$.

Normal forms

A normal form is a term that cannot be evaluated any further — i.e., a term $\mathtt{t}$ is a normal form (or "is in normal form") if there is no $\mathtt{t}'$ such that $\mathtt{t} \longrightarrow \mathtt{t}'$.

A normal form is a state where the abstract machine is halted — i.e., it can be regarded as a "result" of evaluation.

Normal forms

A *normal form* is a term that cannot be evaluated any further — i.e., a term `t` is a normal form (or "is in normal form") if there is no `t'` such that `t` $\longrightarrow$ `t'`.

A normal form is a state where the abstract machine is halted — i.e., it can be regarded as a "result" of evaluation.

Recall that we intended the set of *values* (the boolean constants `true` and `false`) to be exactly the possible "results of evaluation."

Did we get this definition right?

Values = normal forms

Theorem: A term `t` is a value iff it is in normal form.

Proof:

Values = normal forms

Theorem: A term t is a value iff it is in normal form.

Proof: The $\Longrightarrow$ direction is immediate from the definition of the evaluation relation.

Values = normal forms

Theorem: A term t is a value iff it is in normal form.

Proof: The $\Longrightarrow$ direction is immediate from the definition of the evaluation relation.

For the $\Longleftarrow$ direction,

Values = normal forms

Theorem: A term t is a value iff it is in normal form.

Proof: The $\Longrightarrow$ direction is immediate from the definition of the evaluation relation.

For the $\Longleftarrow$ direction, it is convenient to prove the contrapositive: If t is *not* a value, then it is *not* a normal form.

Values = normal forms

Theorem: A term $\mathtt{t}$ is a value iff it is in normal form.

Proof: The $\Longrightarrow$ direction is immediate from the definition of the evaluation relation.

For the $\Longleftarrow$ direction, it is convenient to prove the contrapositive: If $\mathtt{t}$ is *not* a value, then it is *not* a normal form. The argument goes by induction on $\mathtt{t}$.

Note, first, that $\mathtt{t}$ must have the form $\mathtt{if}\ \mathtt{t_1}\ \mathtt{then}\ \mathtt{t_2}\ \mathtt{else}\ \mathtt{t_3}$ (otherwise it would be a value). If $\mathtt{t_1}$ is $\mathtt{true}$ or $\mathtt{false}$, then rule E-IfTrue or E-IfFalse applies to $\mathtt{t}$, and we are done. Otherwise, $\mathtt{t_1}$ is not a value and so, by the induction hypothesis, there is some $\mathtt{t_1'}$ such that $\mathtt{t_1} \longrightarrow \mathtt{t_1'}$. But then rule E-If yields

$$\mathtt{if}\ \mathtt{t_1}\ \mathtt{then}\ \mathtt{t_2}\ \mathtt{else}\ \mathtt{t_3} \longrightarrow \mathtt{if}\ \mathtt{t_1'}\ \mathtt{then}\ \mathtt{t_2}\ \mathtt{else}\ \mathtt{t_3}$$

i.e., $\mathtt{t}$ is not in normal form.

Numbers

New syntactic forms

`t`	::=	...	terms
		`0`	constant zero
		`succ t`	successor
		`pred t`	predecessor
		`iszero t`	zero test
`v`	::=	...	values
		`nv`	numeric value
`nv`	::=		numeric values
		`0`	zero value
		`succ nv`	successor value

New evaluation rules

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

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

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

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

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

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

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

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

Values are normal forms

Our observation a few slides ago that all values are in normal form still holds for the extended language.

Is the converse true? I.e., is every normal form a value?

Stuck terms

Is the converse true? I.e., is every normal form a value?

No: some terms are stuck.

Formally, a stuck term is one that is a normal form but not a value.

Stuck terms model run-time errors.

Multi-step evaluation.

The multi-step evaluation relation, $\longrightarrow^*$, is the reflexive, transitive closure of single-step evaluation.

I.e., it is the smallest relation closed under the following rules:

$$\frac{\mathtt{t} \longrightarrow \mathtt{t}'}{\mathtt{t} \longrightarrow^* \mathtt{t}'}$$

$$\mathtt{t} \longrightarrow^* \mathtt{t}$$

$$\frac{\mathtt{t} \longrightarrow^* \mathtt{t}' \qquad \mathtt{t}' \longrightarrow^* \mathtt{t}''}{\mathtt{t} \longrightarrow^* \mathtt{t}''}$$

Termination of evaluation

Theorem: For every $\mathtt{t}$ there is some normal form $\mathtt{t}'$ such that $\mathtt{t} \longrightarrow^{*} \mathtt{t}'$.

Proof:

Termination of evaluation

Theorem: For every t there is some normal form t' such that $t \longrightarrow^* t'$.

Proof:

- First, recall that single-step evaluation strictly reduces the size of the term:

  if $t \longrightarrow t'$, then $\mathsf{size}(t) > \mathsf{size}(t')$

- Now, assume (for a contradiction) that

  $t_0, t_1, t_2, t_3, t_4, \ldots$

  is an infinite-length sequence such that

  $t_0, \longrightarrow t_1, \longrightarrow t_2, \longrightarrow t_3, \longrightarrow t_4 \longrightarrow \cdots,$

- Then

  $\mathsf{size}(t_0), \mathsf{size}(t_1), \mathsf{size}(t_2), \mathsf{size}(t_3), \mathsf{size}(t_4), \ldots$

  is an infinite, strictly decreasing, sequence of natural numbers.

- But such a sequence cannot exist — contradiction!

Termination Proofs

Most termination proofs have the same basic form:

Theorem: The relation $R \subseteq X \times X$ is terminating — i.e., there are no infinite sequences x_0, x_1, x_2, etc. such that $(x_i, x_{i+1}) \in R$ for each i.

Proof:

1. Choose
   - a well-founded set $(W, <)$ — i.e., a set W with a partial order $<$ such that there are no infinite descending chains $w_0 > w_1 > w_2 > \ldots$ in W
   - a function f from X to W
2. Show $f(x) > f(y)$ for all $(x, y) \in R$
3. Conclude that there are no infinite sequences x_0, x_1, x_2, etc. such that $(x_i, x_{i+1}) \in R$ for each i), since, if there were, we could construct an infinite descending chain in W.