CS 4110

Programming Languages & Logics

Lecture 5
IMP Properties

Review

Last time we defined the IMP programming language...

$$
\begin{aligned}
a &::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2 \\
b &::= \textbf{true} \mid \textbf{false} \mid a_1 < a_2 \\
c &::= \textbf{skip} \\
&\quad \mid x := a \\
&\quad \mid c_1 ; c_2 \\
&\quad \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\
&\quad \mid \textbf{while } b \textbf{ do } c
\end{aligned}
$$

Large-Step Semantics

Again, three relations, one for each syntactic category:

$$\Downarrow_{\mathbf{Aexp}} \subseteq \mathbf{Store} \times \mathbf{Aexp} \times \mathbf{Int}$$

$$\Downarrow_{\mathbf{Bexp}} \subseteq \mathbf{Store} \times \mathbf{Bexp} \times \mathbf{Bool}$$

$$\Downarrow_{\mathbf{Com}} \subseteq \mathbf{Store} \times \mathbf{Com} \times \mathbf{Store}$$

Large-Step Semantics

$$\frac{}{\langle \sigma, n \rangle \Downarrow n}$$

$$\frac{\sigma(x) = n}{\langle \sigma, x \rangle \Downarrow n}$$

$$\frac{\langle \sigma, e_1 \rangle \Downarrow n_1 \qquad \langle \sigma, e_2 \rangle \Downarrow n_2 \qquad n = n_1 + n_2}{\langle \sigma, e_1 + e_2 \rangle \Downarrow n}$$

$$\frac{\langle \sigma, e_1 \rangle \Downarrow n_1 \qquad \langle \sigma, e_2 \rangle \Downarrow n_2 \qquad n = n_1 \times n_2}{\langle \sigma, e_1 \times e_2 \rangle \Downarrow n}$$

Large-Step Semantics

$$\frac{}{\langle \sigma, \mathbf{true} \rangle \Downarrow \mathbf{true}}$$

$$\frac{}{\langle \sigma, \mathbf{false} \rangle \Downarrow \mathbf{false}}$$

$$\frac{\langle \sigma, a_1 \rangle \Downarrow n_1 \qquad \langle \sigma, a_2 \rangle \Downarrow n_2 \qquad n_1 < n_2}{\langle \sigma, a_1 < a_2 \rangle \Downarrow \mathbf{true}}$$

$$\frac{\langle \sigma, a_1 \rangle \Downarrow n_1 \qquad \langle \sigma, a_2 \rangle \Downarrow n_2 \qquad n_1 \geq n_2}{\langle \sigma, a_1 < a_2 \rangle \Downarrow \mathbf{false}}$$

Large-Step Semantics

$$\textsc{Skip}\ \frac{}{\langle \sigma, \textbf{skip} \rangle \Downarrow \sigma}$$

Large-Step Semantics

$$\textsc{Assgn}\ \frac{\langle \sigma, e \rangle \Downarrow n}{\langle \sigma, x := e \rangle \Downarrow \sigma[x \mapsto n]}$$

Large-Step Semantics

$$\textsc{Seq}\ \frac{\langle \sigma, \mathsf{c}_1 \rangle \Downarrow \sigma' \qquad \langle \sigma', \mathsf{c}_2 \rangle \Downarrow \sigma''}{\langle \sigma, \mathsf{c}_1; \mathsf{c}_2 \rangle \Downarrow \sigma''}$$

Large-Step Semantics

$$\textsc{If-T}\ \frac{\langle \sigma, b \rangle \Downarrow \textbf{true} \qquad \langle \sigma, c_1 \rangle \Downarrow \sigma'}{\langle \sigma, \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \rangle \Downarrow \sigma'}$$

$$\textsc{If-F}\ \frac{\langle \sigma, b \rangle \Downarrow \textbf{false} \qquad \langle \sigma, c_2 \rangle \Downarrow \sigma'}{\langle \sigma, \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \rangle \Downarrow \sigma'}$$

Large-Step Semantics

$$\textsc{While-F}\ \frac{\langle \sigma, b \rangle \Downarrow \textbf{false}}{\langle \sigma, \textbf{while}\ b\ \textbf{do}\ c \rangle \Downarrow \sigma}$$

$$\textsc{While-T}\ \frac{\langle \sigma, b \rangle \Downarrow \textbf{true} \qquad \langle \sigma, c \rangle \Downarrow \sigma' \qquad \langle \sigma', \textbf{while}\ b\ \textbf{do}\ c \rangle \Downarrow \sigma''}{\langle \sigma, \textbf{while}\ b\ \textbf{do}\ c \rangle \Downarrow \sigma''}$$

Command Equivalence

Intuitively, two commands are equivalent if they produce the same result under any store...

Definition (Equivalence of commands)

Two commands c and c' are equivalent (written $c \sim c'$) if, for any stores σ and σ', we have

$$\langle \sigma, c \rangle \Downarrow \sigma' \iff \langle \sigma, c' \rangle \Downarrow \sigma'.$$

Command Equivalence

For example, we can prove that every **while** command is equivalent to its "unrolling":

Theorem

For all $b \in \mathbf{Bexp}$ *and* $c \in \mathbf{Com}$ *we have*

$$\textbf{\textit{while}}\ b\ \textbf{\textit{do}}\ c \sim \textbf{\textit{if}}\ b\ \textbf{\textit{then}}\ (c; \textbf{\textit{while}}\ b\ \textbf{\textit{do}}\ c)\ \textbf{\textit{else skip}}.$$

Proof.

We show each implication separately... $\square$

IMP Questions

- Q: Can you write a program that doesn't terminate?

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: **while true do skip**

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: `while true do skip`
- Q: Does this mean that IMP is Turing complete?

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: **while true do skip**
- Q: Does this mean that IMP is Turing complete?
- A: Not quite... we also need to check the language is not finite state... but IMP has real mathematical integers.

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: **while true do skip**
- Q: Does this mean that IMP is Turing complete?
- A: Not quite... we also need to check the language is not finite state... but IMP has real mathematical integers.
- Q: What if we replace **Int** with **Int64**?

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: `while true do skip`
- Q: Does this mean that IMP is Turing complete?
- A: Not quite... we also need to check the language is not finite state... but IMP has real mathematical integers.
- Q: What if we replace **Int** with **Int64**?
- A: Then we would lose Turing completeness.

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: **while true do skip**
- Q: Does this mean that IMP is Turing complete?
- A: Not quite... we also need to check the language is not finite state... but IMP has real mathematical integers.
- Q: What if we replace **Int** with **Int64**?
- A: Then we would lose Turing completeness.
- Q: How much space do we need to represent configurations during execution of an IMP program?

IMP Questions

- Q: Can you write a program that doesn't terminate?
- A: **while true do skip**
- Q: Does this mean that IMP is Turing complete?
- A: Not quite... we also need to check the language is not finite state... but IMP has real mathematical integers.
- Q: What if we replace **Int** with **Int64**?
- A: Then we would lose Turing completeness.
- Q: How much space do we need to represent configurations during execution of an IMP program?
- A: Can calculate a fixed bound!

Determinism

Theorem

$\forall c \in \textbf{Com}, \sigma, \sigma' \ \sigma'' \in \textbf{Store}.$
if $\langle \sigma, c \rangle \Downarrow \sigma'$ *and* $\langle \sigma, c \rangle \Downarrow \sigma''$ *then* $\sigma' = \sigma''$.

Determinism

Theorem

$\forall c \in \textbf{Com}, \sigma, \sigma' \; \sigma'' \in \textbf{Store}.$
if $\langle \sigma, c \rangle \Downarrow \sigma'$ *and* $\langle \sigma, c \rangle \Downarrow \sigma''$ *then* $\sigma' = \sigma''$.

Proof.

By structural induction on *c*... □

Determinism

Theorem

$\forall c \in \textbf{Com}, \sigma, \sigma' \ \sigma'' \in \textbf{Store}.$
if $\langle \sigma, c \rangle \Downarrow \sigma'$ *and* $\langle \sigma, c \rangle \Downarrow \sigma''$ *then* $\sigma' = \sigma''$.

Proof.

By structural induction on *c*... $\square$

Proof.

By induction on the derivation of $\langle \sigma, c \rangle \Downarrow \sigma'$... $\square$

Derivations

Write $\mathcal{D} \Vdash y$ if the conclusion of derivation $\mathcal{D}$ is y.

Derivations

Write $\mathcal{D} \Vdash y$ if the conclusion of derivation $\mathcal{D}$ is y.

Example:

Given the derivation,

$$\dfrac{\dfrac{\dfrac{}{\langle \sigma, 6 \rangle \Downarrow 6} \qquad \dfrac{}{\langle \sigma, 7 \rangle \Downarrow 7}}{\langle \sigma, 6 \times 7 \rangle \Downarrow 42}}{\langle \sigma, \mathsf{i} := 6 \times 7 \rangle \Downarrow \sigma[\mathsf{i} \mapsto 42]}$$

we would write: $\mathcal{D} \Vdash \langle \sigma, \mathsf{i} := 42 \rangle \Downarrow \sigma[\mathsf{i} \mapsto 42]$

Induction on Derivations

Given a set of axioms and inference rules, the set of derivations is itself an inductively defined set!

Induction on Derivations

Given a set of axioms and inference rules, the set of derivations is itself an inductively defined set!

This means we can prove properties by induction on derivations!

Induction on Derivations

Given a set of axioms and inference rules, the set of derivations is itself an inductively defined set!

This means we can prove properties by induction on derivations!

A derivation $\mathcal{D}'$ is an immediate subderivation of $\mathcal{D}$ if $\mathcal{D}' \Vdash z$ where z is one of the premises used of the final rule of derivation $\mathcal{D}$.

Induction on Derivations

Given a set of axioms and inference rules, the set of derivations is itself an inductively defined set!

This means we can prove properties by induction on derivations!

A derivation $\mathcal{D}'$ is an immediate subderivation of $\mathcal{D}$ if $\mathcal{D}' \Vdash z$ where z is one of the premises used of the final rule of derivation $\mathcal{D}$.

In a proof by induction on derivations, for every axiom and inference rule, assume that the property P holds for all immediate subderivations, and show that it holds of the conclusion.

Large-Step Semantics

$$\textsc{Skip}\ \frac{}{\langle \sigma, \mathbf{skip} \rangle \Downarrow \sigma} \qquad \textsc{Assgn}\ \frac{\langle \sigma, a \rangle \Downarrow n}{\langle \sigma, x := a \rangle \Downarrow \sigma[x \mapsto n]}$$

$$\textsc{Seq}\ \frac{\langle \sigma, c_1 \rangle \Downarrow \sigma' \qquad \langle \sigma', c_2 \rangle \Downarrow \sigma''}{\langle \sigma, c_1; c_2 \rangle \Downarrow \sigma''}$$

$$\textsc{If-T}\ \frac{\langle \sigma, b \rangle \Downarrow \mathbf{true} \qquad \langle \sigma, c_1 \rangle \Downarrow \sigma'}{\langle \sigma, \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \rangle \Downarrow \sigma'}$$

$$\textsc{If-F}\ \frac{\langle \sigma, b \rangle \Downarrow \mathbf{false} \qquad \langle \sigma, c_2 \rangle \Downarrow \sigma'}{\langle \sigma, \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \rangle \Downarrow \sigma'}$$

$$\textsc{While-T}\ \frac{\langle \sigma, b \rangle \Downarrow \mathbf{true} \qquad \langle \sigma, c \rangle \Downarrow \sigma' \qquad \langle \sigma', \mathbf{while}\ b\ \mathbf{do}\ c \rangle \Downarrow \sigma''}{\langle \sigma, \mathbf{while}\ b\ \mathbf{do}\ c \rangle \Downarrow \sigma''}$$

$$\textsc{While-F}\ \frac{\langle \sigma, b \rangle \Downarrow \mathbf{false}}{\langle \sigma, \mathbf{while}\ b\ \mathbf{do}\ c \rangle \Downarrow \sigma}$$