Announcements

Office Hours
- Fran: Wednesday at 11-12pm

Homework #1
- Due: Today

Homework #2
- Out: Today
Last time we defined the IMP programming language...

\[
a ::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2
\]

\[
b ::= \text{true} \mid \text{false} \mid a_1 < a_2
\]

\[
c ::= \text{skip} \\
\quad \mid x := a \\
\quad \mid c_1 ; c_2 \\
\quad \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \\
\quad \mid \text{while } b \text{ do } c
\]
Quiz: What does this program do?

```plaintext
x1 := n1; x2 := n2; x3 := n3; tmp := 0;
if x2 <= x1 then
    tmp := x2;
    x2 := x1;
    x1 := tmp
else skip;
if x3 <= x2 then
    tmp := x3;
    x3 := x2;
    x2 := tmp
else skip
```
How about this one?

```plaintext
x1 := n1;  x2 := n2;  x3 := n3;  tmp := 0;  swaps := 1;
while 0 < swaps do
  swaps := 0;
  if x2 <= x1 then
    tmp := x2;
    x2 := x1;
    x1 := tmp;
    swaps := swaps + 1
  else skip;
  if x3 <= x2 then
    tmp := x3;
    x3 := x2;
    x2 := tmp;
    swaps := swaps + 1
  else skip
```

5
Does this program terminate?

```plaintext
x1 := n1; x2 := n2; x3 := n3; tmp := 0; swaps := 1;
while 0 < swaps do
  swaps := 0;
  if x2 <= x1 then
    tmp := x2;
    x2 := x1;
    x1 := tmp;
    swaps := swaps + 1
  else skip;
  if x3 <= x2 then
    tmp := x3;
    x3 := x2;
    x2 := tmp;
    swaps := swaps + 1
  else skip
```

5
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 $\text{Int}$ with $\text{Int64}$?

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!
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!
**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!
IMP Questions

- Q: Can you write a program that doesn’t terminate?
- A: \textbf{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**

\( \forall c \in \text{Com}, \sigma, \sigma', \sigma'' \in \text{Store}. \)

*if \( \langle \sigma, c \rangle \Downarrow \sigma' \) and \( \langle \sigma, c \rangle \Downarrow \sigma'' \) then \( \sigma' = \sigma'' \).*
Theorem

\[ \forall c \in \text{Com}, \sigma, \sigma', \sigma'' \in \text{Store}. \]

if \( \langle \sigma, c \rangle \Downarrow \sigma' \) and \( \langle \sigma, c \rangle \Downarrow \sigma'' \) then \( \sigma' = \sigma'' \).

Proof.

By structural induction on \( c \)...
Theorem

\[ \forall c \in \text{Com}, \sigma, \sigma', \sigma'' \in \text{Store}. \]

if \( \langle \sigma, c \rangle \downarrow \sigma' \) and \( \langle \sigma, c \rangle \downarrow \sigma'' \) then \( \sigma' = \sigma'' \).

Proof.

By structural induction on \( c \)...

Proof.

By induction on the derivation of \( \langle \sigma, c \rangle \downarrow \sigma' \)...

Determinism
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,

\[
\begin{align*}
\langle \sigma, 6 \rangle \downarrow 6 & \quad \langle \sigma, 7 \rangle \downarrow 7 \\
\langle \sigma, 6 \times 7 \rangle \downarrow 42 & \quad \langle \sigma, i := 6 \times 7 \rangle \downarrow \sigma[i \mapsto 42]
\end{align*}
\]

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

\[
\begin{align*}
\text{Skip:} & \quad \langle \sigma, \text{skip} \rangle \Downarrow \sigma \\
\text{Assign:} & \quad \langle \sigma, e \rangle \Downarrow n \\
\text{Seq:} & \quad \langle \sigma, c_1 \rangle \Downarrow \sigma' \quad \langle \sigma', c_2 \rangle \Downarrow \sigma'' \\
\text{If-T:} & \quad \langle \sigma, b \rangle \Downarrow \text{true} \quad \langle \sigma, c_1 \rangle \Downarrow \sigma' \\
\text{If-F:} & \quad \langle \sigma, b \rangle \Downarrow \text{false} \quad \langle \sigma, c_2 \rangle \Downarrow \sigma' \\
\text{While-T:} & \quad \langle \sigma, b \rangle \Downarrow \text{true} \quad \langle \sigma, c \rangle \Downarrow \sigma' \quad \langle \sigma', \text{while } b \text{ do } c \rangle \Downarrow \sigma'' \\
\text{While-F:} & \quad \langle \sigma, b \rangle \Downarrow \text{false}
\end{align*}
\]