Lecture 5
IMP Properties
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 $\sigma$ and $\sigma'$, we have

$$\langle \sigma, c \rangle \downarrow \sigma' \iff \langle \sigma, c' \rangle \downarrow \sigma'.$$
Command Equivalence

For example, we can prove that every \textbf{while} command is equivalent to its “unrolling”:

\textbf{Theorem}

For all $b \in \textbf{Bexp}$ and $c \in \textbf{Com}$,

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

\textbf{Proof.}

We show each implication separately...
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: \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.

• Q: What if we replace \textbf{Int} with \textbf{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: \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.

- Q: What if we replace \texttt{Int} with \texttt{Int64}?
  - A: Then we would lose Turing completeness.

- Q: How much space do we need to represent configurations during execution of an IMP program?
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.

Q: What if we replace \texttt{Int} with \texttt{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 \( c \)...

\[ \square \]
Determinism

Theorem

\[ \forall c \in \textbf{Com}, \sigma, \sigma', \sigma'' \in \textbf{Store}. \]
\[ \text{if } \langle \sigma, c \rangle \downarrow \sigma' \text{ and } \langle \sigma, c \rangle \downarrow \sigma'' \text{ then } \sigma' = \sigma''. \]

Proof.

By structural induction on c...

Proof.

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

\[
\begin{align*}
\langle \sigma, 6 \rangle & \Downarrow 6 \\
\langle \sigma, 7 \rangle & \Downarrow 7 \\
\langle \sigma, 6 \times 7 \rangle & \Downarrow 42
\end{align*}
\]

we would write: $\mathcal{D} \vdash \langle \sigma, i := 6 \times 7 \rangle \Downarrow \sigma[i \mapsto 42]$
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 $D'$ is an immediate subderivation of $D$ if $D' \vdash z$ where $z$ is one of the premises used of the final rule of derivation $D$. 
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 inference rule, assume that the property $P$ holds for all immediate subderivations, and show that it holds of the conclusion.
### Large-Step Semantics

- **Skip**
  \[
  \langle \sigma, \text{skip} \rangle \Downarrow \sigma
  \]

- **Assign**
  \[
  \langle \sigma, a \rangle \Downarrow n \\
  \langle \sigma, x := a \rangle \Downarrow \sigma[x \mapsto n]
  \]

- **Seq**
  \[
  \langle \sigma, c_1 \rangle \Downarrow \sigma' \\
  \langle \sigma', c_2 \rangle \Downarrow \sigma'' \\
  \langle \sigma, c_1; c_2 \rangle \Downarrow \sigma''
  \]

- **If-T**
  \[
  \langle \sigma, b \rangle \Downarrow \text{true} \\
  \langle \sigma, c_1 \rangle \Downarrow \sigma' \\
  \langle \sigma, \text{if } b \text{ then } c_1 \text{ else } c_2 \rangle \Downarrow \sigma'
  \]

- **If-F**
  \[
  \langle \sigma, b \rangle \Downarrow \text{false} \\
  \langle \sigma, c_2 \rangle \Downarrow \sigma' \\
  \langle \sigma, \text{if } b \text{ then } c_1 \text{ else } c_2 \rangle \Downarrow \sigma'
  \]

- **While-T**
  \[
  \langle \sigma, b \rangle \Downarrow \text{true} \\
  \langle \sigma, c \rangle \Downarrow \sigma' \\
  \langle \sigma', \text{while } b \text{ do } c \rangle \Downarrow \sigma'' \\
  \langle \sigma, \text{while } b \text{ do } c \rangle \Downarrow \sigma''
  \]

- **While-F**
  \[
  \langle \sigma, b \rangle \Downarrow \text{false} \\
  \langle \sigma, \text{while } b \text{ do } c \rangle \Downarrow \sigma
  \]