Lecture 5
IMP Properties
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

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

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

\textbf{Theorem}

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

\texttt{while} $b$ \texttt{do } $c$ \texttt{\sim if } $b$ \texttt{then } $(c; \texttt{while } b \texttt{do } c)$ \texttt{else 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`

- 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.
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?
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!
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 \( \sigma' = \sigma'' \).
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 \( \sigma' = \sigma'' \).

Proof.

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

Theorem

∀c ∈ Com, σ, σ′, σ'' ∈ Store.

if ⟨σ, c⟩ ↓ σ' and ⟨σ, c⟩ ↓ σ'' then σ' = σ''.

Proof.

By structural induction on c...

Proof.

By induction on the derivation of ⟨σ, c⟩ ↓ σ'...
Derivations

Write \( \mathcal{D} \vdash y \) if the conclusion of derivation \( \mathcal{D} \) is \( y \).
Write $\mathcal{D} \vdash y$ if the conclusion of derivation $\mathcal{D}$ is $y$.

**Example:**

Given the derivation,

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

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!
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 $D'$ is an immediate subderivation of $D$ if $D' \vdash z$ where $z$ is one of the premises used of the final rule of derivation $D$.

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

\[
\begin{align*}
\text{Skip} & \quad \langle \sigma, \text{skip} \rangle \downarrow \sigma \\
\text{ASSGN} & \quad \langle \sigma, x := a \rangle \downarrow \sigma[x \mapsto n] \\
\text{SEQ} & \quad \langle \sigma, c_1 \rangle \downarrow \sigma' \quad \langle \sigma', c_2 \rangle \downarrow \sigma'' \\
& \quad \langle \sigma, c_1 ; c_2 \rangle \downarrow \sigma'' \\
\text{IF-T} & \quad \langle \sigma, b \rangle \downarrow \text{true} \quad \langle \sigma, c_1 \rangle \downarrow \sigma' \\
& \quad \langle \sigma, \text{if } b \text{ then } c_1 \text{ else } c_2 \rangle \downarrow \sigma' \\
\text{IF-F} & \quad \langle \sigma, b \rangle \downarrow \text{false} \quad \langle \sigma, c_2 \rangle \downarrow \sigma' \\
& \quad \langle \sigma, \text{if } b \text{ then } c_1 \text{ else } c_2 \rangle \downarrow \sigma' \\
\text{WHILE-T} & \quad \langle \sigma, b \rangle \downarrow \text{true} \quad \langle \sigma, c \rangle \downarrow \sigma' \quad \langle \sigma', \text{while } b \text{ do } c \rangle \downarrow \sigma'' \\
& \quad \langle \sigma, \text{while } b \text{ do } c \rangle \downarrow \sigma'' \\
\text{WHILE-F} & \quad \langle \sigma, b \rangle \downarrow \text{false} \\
& \quad \langle \sigma, \text{while } b \text{ do } c \rangle \downarrow \sigma
\end{align*}
\]