Induction

Last week

We talked about small-step operational semantics

- compared it to its big-step counterpart
- discussed program equivalence
- talked about derivation sequences vs derivation trees

This week

Induction

- Structural induction
- Induction on derivation trees
- Induction on the length of derivation sequences
- Common pitfalls
- Tips and tricks of the trade

IT UNIVERSITY OF COPENHAGEN

Questions before we start?

IT UNIVERSITY OF COPENHAGEN

Induction on numbers

The most common type of induction is induction on natural numbers

- Prove that a property holds for 0
- Assuming that a property holds for m, prove that it holds for m + 1

Induction on numbers

- Prove that a property holds for 0
- Assuming that a property holds for m, prove that it holds for m + 1

More formally this is written as

$$\frac{P\ 0 \qquad \forall m.\ P\ m \rightarrow P\ (m+1)}{P\ n}$$

Induction on numbers

- Prove that a property holds for 0
- Assuming that a property holds for m, prove that it holds for m + 1

More formally this is written as

$$\frac{P\ 0 \qquad \forall m.\ P\ m \rightarrow P\ (m + 1)}{P\ n}$$

This is referred to as **structural induction**

Structural induction

We can do induction on anything that is inductively defined

Induction on numbers

Inductive definition of natural numbers

$$N \triangleq 0 \mid S\ N$$

Induction principle

$$\frac{P\ 0 \qquad \forall m.\ P\ m \to P\ (S\ m)}{P\ n}$$

Induction on lists

Inductive definition of lists

$$\begin{aligned} \alpha\ \text{list} \triangleq\ & [\,] \\ & \alpha :: \alpha\ \text{list} \end{aligned}$$

Induction principle

$$\frac{P\ [\,] \qquad \forall x\ xs.\ P\ xs \rightarrow P\ (x{::}xs)}{P\ lst}$$

Induction on trees

Inductive definition of binary trees

$$\alpha \text{ tree} \triangleq \text{Leaf}$$
$$\text{Node } (\alpha \text{ tree})\ \alpha\ (\alpha \text{ tree})$$

Induction principle

$$\frac{\begin{array}{l}P\ \text{Leaf} \\ \forall l\ v\ r.\ P\ l \rightarrow P\ r \rightarrow P\ (\text{Node}\ l\ v\ r)\end{array}}{P\ t}$$

Exercise

Write an induction principle for the While language

$$
\begin{aligned}
S \triangleq\ & \text{skip} \\
& a := v \\
& S;\ S \\
& \text{if } b \text{ then } S \text{ else } S \\
& \text{while } b \text{ do } S
\end{aligned}
$$

Determinism of While

The While-language is deterministic

if $\langle S, s \rangle \rightarrow s'$ and $\langle S, s \rangle \rightarrow s''$
then $s' = s''$

Proof by induction on S

Determinism of While

We get stuck in the case for composition

if $\langle S1; S2, s\rangle \to s'$ and $\langle S1; S2, s\rangle \to s''$
then $s' = s''$

Induction hypothesis

if $\langle S1, s\rangle \to s'$ and $\langle S1, s\rangle \to s''$
then $s' = s''$

if $\langle S2, s\rangle \to s'$ and $\langle S2, s\rangle \to s''$
then $s' = s''$

Determinism of While

These are the original states, not the states acquired when doing case analysis on the derivation of S1; S2

Induction hypothesis

if ⟨S1, s⟩ → s′ and ⟨S1, s⟩ → s″
then s′ = s″

if ⟨S2, s⟩ → s′ and ⟨S2, s⟩ → s″
then s′ = s″

Strengthening induction

We need to strengthen the induction hypothesis

We say that a formula is stronger than another one if it proves strictly more things.

A strong induction principle is applicable to relevant intermediate steps, not just the first one

Determinism of While

The While-language is deterministic

if **for all possible states** s, s′ and s″ we have that ⟨S, s⟩ → s′ and ⟨S, s⟩ → s″ then s′ = s″

Proof by induction on S

Determinism of While

Composition now works

if $\langle S1; S2, s\rangle \rightarrow s'$ and $\langle S1; S2, s\rangle \rightarrow s''$
then $s' = s''$

Induction hypothesis

if $\forall s\ s'\ s''$, $\langle S1, s\rangle \rightarrow s'$ and $\langle S1, s\rangle \rightarrow s''$
then $s' = s''$

if $\forall s\ s'\ s''$, $\langle S2, s\rangle \rightarrow s'$ and $\langle S2, s\rangle \rightarrow s''$
then $s' = s''$

Determinism of While

… but the while command breaks

if ⟨**while** b **do** S, s⟩ → s′ and
⟨**while** b **do** S, s⟩ → s″
then s′ = s″

Induction hypothesis

if ∀s s′ s″, ⟨S, s⟩ → s′ and ⟨S, s⟩ → s″
then s′ = s″

but the while rule requires that we have what we are trying to prove

The while loop

$$[\text{while}^{\text{tt}}_{\text{ns}}]\frac{\langle S, s_1\rangle \to s_2 \quad \langle \textbf{while}\ b\ \textbf{do}\ S, s_2\rangle \to s_3}{\langle \textbf{while}\ b\ \textbf{do}\ S, s_1\rangle \to s_3}\ \mathscr{B}[b]_s = \textbf{tt}$$

The command **while** b **do** S appears both above and bellow the line and structural induction requires that we apply the induction hypothesis on something that is structurally smaller

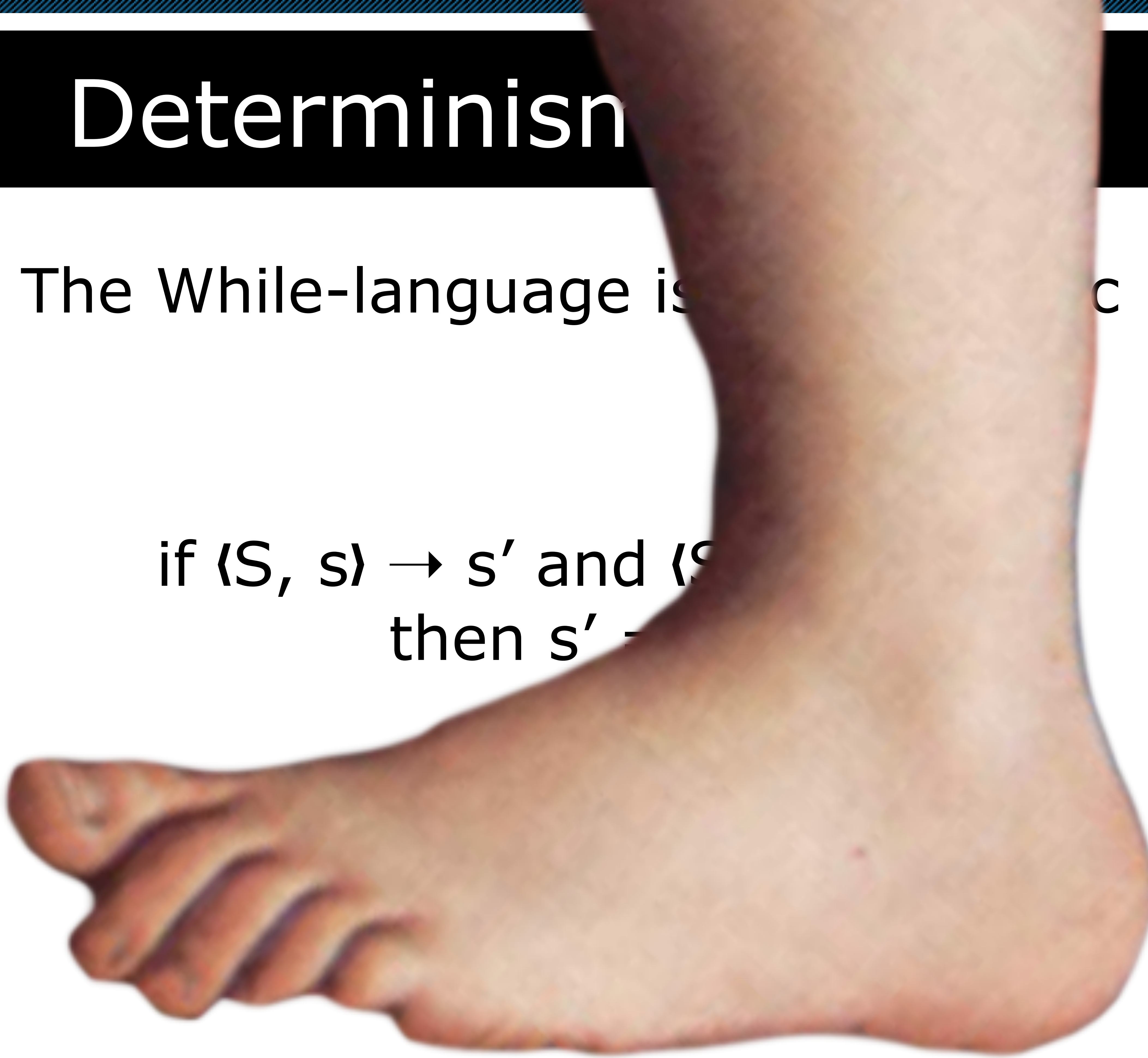
Determinism

The While-language is

c

if ⟨S, s⟩ → s′ and ⟨S

then s′



Induction on trees

Our inference rules are inductively defined. We can do induction on that

Induction on trees

Our inference rules are inductively defined. We can do induction on that

If we want to prove something about the derivation $\langle S, s\rangle \rightarrow s'$, we do induction on that derivation in stead of S.

The book calls this induction on the shape of derivation trees, but it is just the same as structural induction

Intuition

When doing induction on a derivation tree $\langle S, s\rangle \rightarrow s'$, the predicate for our induction principle is a ternary one that takes S, s and s′ as arguments.

Induction principle

We have the following skeleton for an induction principle

$$\langle S, s\rangle \rightarrow s'$$

$$\vdots$$

Inductive cases

$$\overline{\qquad\qquad\qquad\qquad}$$

$$P\ S\ s\ s'$$

Induction principle

We have the following skeleton for an induction principle

$$\begin{array}{c} \langle S, s\rangle \to s' \\ \vdots \\ \hline P\ S\ s\ s' \end{array}$$

We do induction over this transition

Inductive cases

Induction principle

We have the following skeleton for an induction principle

$$\frac{\langle S, s\rangle \to s' \quad \vdots \quad \text{Inductive cases}}{P\ S\ s\ s'}$$

We want to prove this

Determinism of While

The While-language is deterministic

if $\langle S, s\rangle \rightarrow s'$ and $\langle S, s\rangle \rightarrow s''$
then $s' = s''$

We do induction on: $\langle S, s\rangle \rightarrow s'$

Induction predicate:
fun S s s' $\Rightarrow$ if $\langle S, s\rangle \rightarrow s''$ then $s' = s''$

Skip

The skip command does nothing

$$[\text{skip}_{ns}]\quad \langle \textbf{skip}, s\rangle \rightarrow s$$

Skip

The skip command does nothing

$$[\text{skip}_{ns}]\ \langle \textbf{skip}, s\rangle \to s$$

Inductive case:
$\forall s.\ P\ \textbf{skip}\ s\ s$

Assignment

The assignment command updates the state

$$[\text{ass}_{\text{ns}}] \quad \langle x := a, s\rangle \to s[x \mapsto \mathcal{A}[a]_s]$$

Assignment

The assignment command updates the state

$$[\mathrm{ass}_{ns}] \quad \langle x := a, s\rangle \rightarrow s[x \mapsto \mathcal{A}[a]_s]$$

Inductive case:

$$\forall\ x\ a\ s.\ P\ (x := a)\ s\ (s[x \mapsto \mathcal{A}[a]_s])$$

Sequential composition

Sequential composition runs a command from a state provided by the previous command

$$[\text{comp}_{\text{ns}}]\ \frac{\langle S_1, s\rangle \rightarrow s'' \qquad \langle S_2, s''\rangle \rightarrow s'}{\langle S_1; S_2, s\rangle \rightarrow s'}$$

Sequential composition

Sequential composition runs a command from a state provided by the previous command

$$[\text{comp}_{ns}]\ \frac{\langle S_1, s\rangle \to s'' \qquad \langle S_2, s''\rangle \to s'}{\langle S_1; S_2, s\rangle \to s'}$$

Inductive case:

$$\forall s\ s'\ s''\ S_1\ S_2.\ \langle S_1, s\rangle \to s'' \Longrightarrow$$
$$\langle S_2, s''\rangle \to s' \Longrightarrow P\ S_1\ s\ s'' \Longrightarrow$$
$$P\ S_2\ s''\ s' \Longrightarrow P\ (S_1;S_2)\ s\ s'$$

Conditional statements

A conditional statement executes the first branch if the guard is true

$$[\text{if}^{\text{tt}}_{\text{ns}}]\ \frac{\langle S_1, s\rangle \to s'}{\langle \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2, s\rangle \to s'}\ \mathscr{B}[b]_s = \textbf{tt}$$

Conditional statements

A conditional statement executes the first branch if the guard is true

$$[\text{if}^{tt}_{ns}]\ \frac{\langle S_1, s\rangle \to s'}{\langle \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2, s\rangle \to s'}\ \mathscr{B}[b]_s = \textbf{tt}$$

Inductive case:

$$\forall s\ s'\ b\ S_1\ S_2.\ \mathscr{B}[b]_s = \textbf{tt} \Longrightarrow \langle S_1, s\rangle \to s' \Longrightarrow$$
$$P\ S_1\ s\ s' \Longrightarrow$$
$$P\ (\textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2)\ s\ s'$$

Conditional statements

A conditional statement executes the second branch if the guard is false

$$[\mathrm{if}^{\mathrm{ff}}_{\mathrm{ns}}]\ \frac{\langle S_2, s\rangle \rightarrow s'}{\langle \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2, s\rangle \rightarrow s'}\ \mathscr{B}[b]_s = \textbf{ff}$$

Conditional statements

A conditional statement executes the second branch if the guard is false

$$[\text{if}^{\text{ff}}_{\text{ns}}]\ \frac{\langle S_2, s\rangle \to s'}{\langle \textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2, s\rangle \to s'}\ \mathscr{B}[b]_s = \textbf{ff}$$

Inductive case:

$$\forall s\ s'\ b\ S_1\ S_2.\ \mathscr{B}[b]_s = \textbf{ff} \Longrightarrow \langle S_2, s\rangle \to s' \Longrightarrow$$

$$P\ S_2\ s\ s' \Longrightarrow$$

$$P\ (\textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2)\ s\ s'$$

Conditional statements

A loop will keep executing as long as its guard is true

$$[\text{while}^{tt}_{ns}]\ \frac{\langle S, s\rangle \to s'' \quad \langle \textbf{while}\ b\ \textbf{do}\ S, s''\rangle \to s'}{\langle \textbf{while}\ b\ \textbf{do}\ S, s\rangle \to s'}\ \mathscr{B}[b]_s = \textbf{tt}$$

Conditional statements

A loop will keep executing as long as its guard is true

$$[\text{while}^{tt}_{ns}]\ \frac{\langle S, s\rangle \to s'' \quad \langle \textbf{while}\ b\ \textbf{do}\ S, s''\rangle \to s'}{\langle \textbf{while}\ b\ \textbf{do}\ S, s\rangle \to s'}\ \mathscr{B}[b]_s = \textbf{tt}$$

Inductive case:

$$\forall s\ s'\ s''\ b\ S.\ \langle S, s\rangle \to s'' \Rightarrow \langle \textbf{while}\ b\ \textbf{do}\ S, s''\rangle \to s'$$
$$P\ S\ s\ s'' \Rightarrow P\ (\textbf{while}\ b\ \textbf{do}\ S)\ s''\ s' \Rightarrow$$
$$\mathscr{B}[b]_s = \textbf{tt} \Rightarrow P\ (\textbf{while}\ b\ \textbf{do}\ S)\ s\ s'$$

Conditional statements

A loop with a false guard behaves exactly like skip

$$[\text{while}^{\text{ff}}_{\text{ns}}] \quad \langle \textbf{while}\ b\ \textbf{do}\ S,\ s\rangle \to s \quad \mathscr{B}[b]_s = \textbf{ff}$$

Conditional statements

A loop with a false guard behaves exactly like skip

$$[\text{while}^{\text{ff}}_{\text{ns}}] \quad \langle \textbf{while } b \textbf{ do } S, s\rangle \to s \quad \mathscr{B}[b]_s = \textbf{ff}$$

Inductive case:

$$\forall s\ b\ S,\ \mathscr{B}[b]_s = \textbf{ff} \Longrightarrow P\ (\textbf{while } b \textbf{ do } S)\ s\ s$$

Induction rule for While

$\langle S, s\rangle \rightarrow s'$ $\forall s.\ P\ \textbf{skip}\ s\ s$

$\forall x\ a\ s.\ P\ (x := a)\ s\ (s[x \mapsto \mathcal{A}[a]_s])$

$\forall s\ s'\ s''\ S_1\ S_2.\ \langle S_1, s\rangle \rightarrow s'' \Longrightarrow \langle S_2, s''\rangle \rightarrow s' \Longrightarrow$

$P\ S_1\ s\ s'' \Longrightarrow P\ S_2\ s''\ s' \Longrightarrow P\ (S_1;S_2)\ s\ s'$

$\forall s\ s'\ b\ S_1\ S_2.\ \mathcal{B}[b]_s = \textbf{tt} \Longrightarrow \langle S_1, s\rangle \rightarrow s' \Longrightarrow$

$P\ S_1\ s\ s' \Longrightarrow P\ (\textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2)\ s\ s'$

$\forall s\ s'\ b\ S_1\ S_2.\ \mathcal{B}[b]_s = \textbf{ff} \Longrightarrow \langle S_2, s\rangle \rightarrow s' \Longrightarrow$

$P\ S_2\ s\ s' \Longrightarrow P\ (\textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2)\ s\ s'$

$\forall s\ s'\ s''\ b\ S.\ \langle S, s\rangle \rightarrow s'' \Longrightarrow \langle \textbf{while}\ b\ \textbf{do}\ S, s''\rangle \rightarrow s'$

$P\ S\ s\ s'' \Longrightarrow P\ (\textbf{while}\ b\ \textbf{do}\ S)\ s''\ s' \Longrightarrow$

$\mathcal{B}[b]_s = \textbf{tt} \Longrightarrow P\ (\textbf{while}\ b\ \textbf{do}\ S)\ s\ s'$

$\forall s\ b\ S.\ \mathcal{B}[b]_s = \textbf{ff} \Longrightarrow P\ (\textbf{while}\ b\ \textbf{do}\ S)\ s\ s$

$P\ S\ s\ s'$

Questions?

Determinism of While

The While-language is deterministic

if for all possible states s, s' and s'' we have that $\langle S, s\rangle \rightarrow s'$ and $\langle S, s\rangle \rightarrow s''$ then $s' = s''$

Proof by induction on $\langle S, s\rangle \rightarrow s'$

Small-step induction

Structural induction on the derivation does not work for small-step semantics.

Small-step induction

Structural induction on the derivation does not work for small-step semantics.

$$\langle S, s\rangle \rightarrow^k \gamma$$

Here we need to do induction on the length of the derivation sequence k. In effect this is just standard induction on the natural numbers

Example proof

Prove that if

$\langle S_1; S_2, s\rangle \rightarrow^{k} s''$

then there exists a state s′ and natural numbers k_1 and k_2 such that

$\langle S_1, s\rangle \rightarrow^{k1} s'$

and

$\langle S_2, s'\rangle \rightarrow^{k2} s''$

and

$k = k_1 + k_2$

Next week

- We extend the While-language with more constructs and prove properties about them
- We introduce the concept of Program Logic