Programming Languages and Computation

Week 8: Reasoning with Derivations

1 Semantic Equivalence

** 1. Suppose that $S_1, S_2, S_3 \in \mathcal{S}$ are statements. Prove that the statement $\{S_1;\ S_2\};\ S_3$ is semantically equivalent to the statement $S_1;\ \{S_2;\ S_3\}$. That is, prove that for, any two states $\sigma, \sigma' \in \mathsf{State}$, $\{S_1;\ S_2\};\ S_3, \sigma \Downarrow \sigma'$ if, and only if, $S_1;\ \{S_2;\ S_3\}, \sigma \Downarrow \sigma'$.

** 2.

(a) Suppose that $e \in \mathcal{A}$ is an arithmetic expression that is semantically equivalent to the arithmetic expression $x \in \mathcal{A}$. Prove that the statement $x \leftarrow e$ is semantically equivalent to the statement skip.

(b) Find an expression $e \in \mathcal{A}$ that is *not* semantically equivalent to x but where the statement $y \leftarrow 0;\ x \leftarrow e$ is semantically equivalent to $y \leftarrow 0$.

** 3.

(a) Prove that the statements $\mathsf{if}\ e\ \mathsf{then}\ S_1;\ \ S_3\ \mathsf{else}\ S_2;\ \ S_3$ and the statement $\{\mathsf{if}\ e\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\};\ S_3$ are semantically equivalent for any Boolean expression $e \in \mathcal{B}$ and statements $S_1, S_2, S_3 \in \mathcal{S}$.

(b) Find an instance where a statement of the form $\mathsf{if}\ e\ \mathsf{then}\ S_1;\ S_2\ \mathsf{else}\ S_1;\ S_3$ and the related statement $S_1;\ \{\mathsf{if}\ e\ \mathsf{then}\ S_2\ \mathsf{else}\ S_3\}$ are not semantically equivalent for some Boolean expression $e \in \mathcal{B}$ and statements $S_1, S_2, S_3 \in \mathcal{S}$.

*** 4. Suppose that $e_1 \in \mathcal{A}$ and $e_2 \in \mathcal{A}$ are arithmetic expressions such that $x \notin \mathsf{FV}(e_2)$ and $y \notin \mathsf{FV}(e_1)$. Prove that the statement $x \leftarrow e_1;\ y \leftarrow e_2$ and statement $y \leftarrow e_2;\ x \leftarrow e_1$ are semantically equivalent. You may use the following result about the denotation of arithmetic expressions:

$$\text{if } \forall x \in \mathsf{FV}(e).\, \sigma(x) = \sigma'(x) \text{ then } [\![e]\!]_{\mathcal{A}}(\sigma) = [\![e]\!]_{\mathcal{A}}(\sigma')$$

** 5. Let us suppose we introduce a new language construct so that statements are defined by the following grammar:

$$S \to \mathsf{skip} \mid x \leftarrow A \mid S_1;\ S_2 \mid \mathsf{if}\ e\ \mathsf{then}\ S\ \mathsf{else} \mid \cdots \mid \mathsf{if}\ e\ \mathsf{then}\ S$$

The operational behaviour of the new construct is given by the inference rules:

$$\frac{S, \sigma \Downarrow \sigma'}{\textsf{if } e \textsf{ then } S, \sigma \Downarrow \sigma'} [\![e]\!]_{\mathcal{B}}(\sigma) = \top \qquad \frac{}{\textsf{if } e \textsf{ then } S, \sigma \Downarrow \sigma} [\![e]\!]_{\mathcal{B}}(\sigma) = \bot$$

Show that the statement if e then S is semantically equivalent to the statement if e then S else skip for any statement $S \in \mathcal{S}$ and Boolean expression $e \in \mathcal{B}$.

2 Proving Termination

** 6. Consider the following While program P:

```
x ← y;
while y + 1 ≤ x * x do
  x ← x - 1;
```

(a) Calculate the final state when executed in the initial states $[y \mapsto 4]$ and $[y \mapsto 5]$.

(b) What function does this program compute?

(c) Prove by induction on $\sigma(x)$ that this program terminates in any state $\sigma \in$ State where $\sigma(y) \geq 0$. That is, prove that, for any state $\sigma \in$ State such that $\sigma(y) \geq 0$, there exists a state $\sigma' \in$ State such that $P, \sigma \Downarrow \sigma'$ where P is the above program.

*** 7. Recall the strong induction principle from the previous sheet:

In order to prove $\forall n \in \mathbb{N}. P(n)$, prove:

1. $P(0)$;
2. And, $P(n+1)$ under the assumption that $P(m)$ holds for all $m \leq n$.

Using the strong induction principle prove the following program terminates:

```
while 1 ≤ x do
  x ← x - 2
```

when executed in any state where $\sigma(x) \geq 0$.

*** 8. Consider the following While program:

```
while 1 ≤ x + y do
  if x ≤ y
    then y ← y - 1
    else x ← x - 1
```

We wish to prove that this program will terminate.

Proof by induction need not be applied to a particular variable, but can generalised to induction over an arbitrary function $f : \mathsf{State} \to \mathbb{N}$ of the state. In such a proof, the base case consider any state where $f(\sigma) = 0$ and the inductive case consider any state where $f(\sigma) = n + 1$ under the assumption that the property holds of $f(\sigma) = n$.

Using this principle, prove that the program terminates when executed in any state $\sigma \in \mathsf{State}$ such that $\sigma(x), \sigma(y) \geq 0$ by induction over $\sigma(x) + \sigma(y)$.

3 Induction over Derivation

** 9. Consider the non-terminating program "while true do skip".

(a) Re-formulate the statement $\forall \sigma \in \mathsf{State}. \nexists \sigma' \in \mathsf{State}. \mathsf{while\ true\ do\ skip}, \sigma \Downarrow \sigma'$ (i.e. the program doesn't terminate in any state) as a statement of the form $\forall (S, \sigma, \sigma') \in \Downarrow . P(S, \sigma, \sigma')$ for some predicate P.

(b) Using the formulation constructed in part 1, prove that the program does not terminate by structural induction. You may omit cases where the $P(S, \sigma, \sigma')$ is trivially false.

** 10. Consider the loop L from Question 6:

```
while y + 1 ≤ x * x do
  x ← x − 1;
```

(a) Prove that, if $L, \sigma \Downarrow \sigma'$, then $\sigma'(x)^2 \leq \sigma(y)$ for any states $\sigma, \sigma' \in \mathsf{State}$ by structural induction over the derivation.

(b) Using this fact, informally argue that if $x \leftarrow y, L, \sigma \Downarrow \sigma'$ then $\sigma'(x)$ is the *largest* integer such that $\sigma'(x)^2 \leq \sigma(y)$.

*** 11. Let us extend the definition of free variables to apply to statements $\mathsf{FV} : \mathcal{S} \to \mathcal{P}(\mathsf{Var})$ as follows:

$$\begin{aligned}
\mathsf{FV}(\mathsf{skip}) &= \emptyset \\
\mathsf{FV}(x \leftarrow e) &= \mathsf{FV}(e) \\
\mathsf{FV}(S_1;\ S_2) &= \mathsf{FV}(S_1) \cup \mathsf{FV}(S_2) \\
\mathsf{FV}(\mathsf{if}\ e\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2) &= \mathsf{FV}(e_1) \cup \mathsf{FV}(S_1) \cup \mathsf{FV}(S_2) \\
\mathsf{FV}(\mathsf{while}\ e\ \mathsf{do}\ S) &= \mathsf{FV}(e) \cup \mathsf{FV}(S)
\end{aligned}$$

Prove the following statement by structural induction over derivations:

$$\text{if } S, \sigma \Downarrow \sigma' \text{ and } x \notin \mathsf{FV}(S) \text{ then } S, \sigma[x \mapsto n] \Downarrow \sigma'[x \mapsto n]$$

You may use the following result about the denotation of arithmetic expressions:

$$\text{if } \forall x \in \mathsf{FV}(e). \sigma(x) = \sigma'(x) \Rightarrow [\![e]\!]_{\mathcal{A}}(\sigma) = [\![e]\!]_{\mathcal{A}}(\sigma')$$

and the equivalent statement about Boolean expressions.