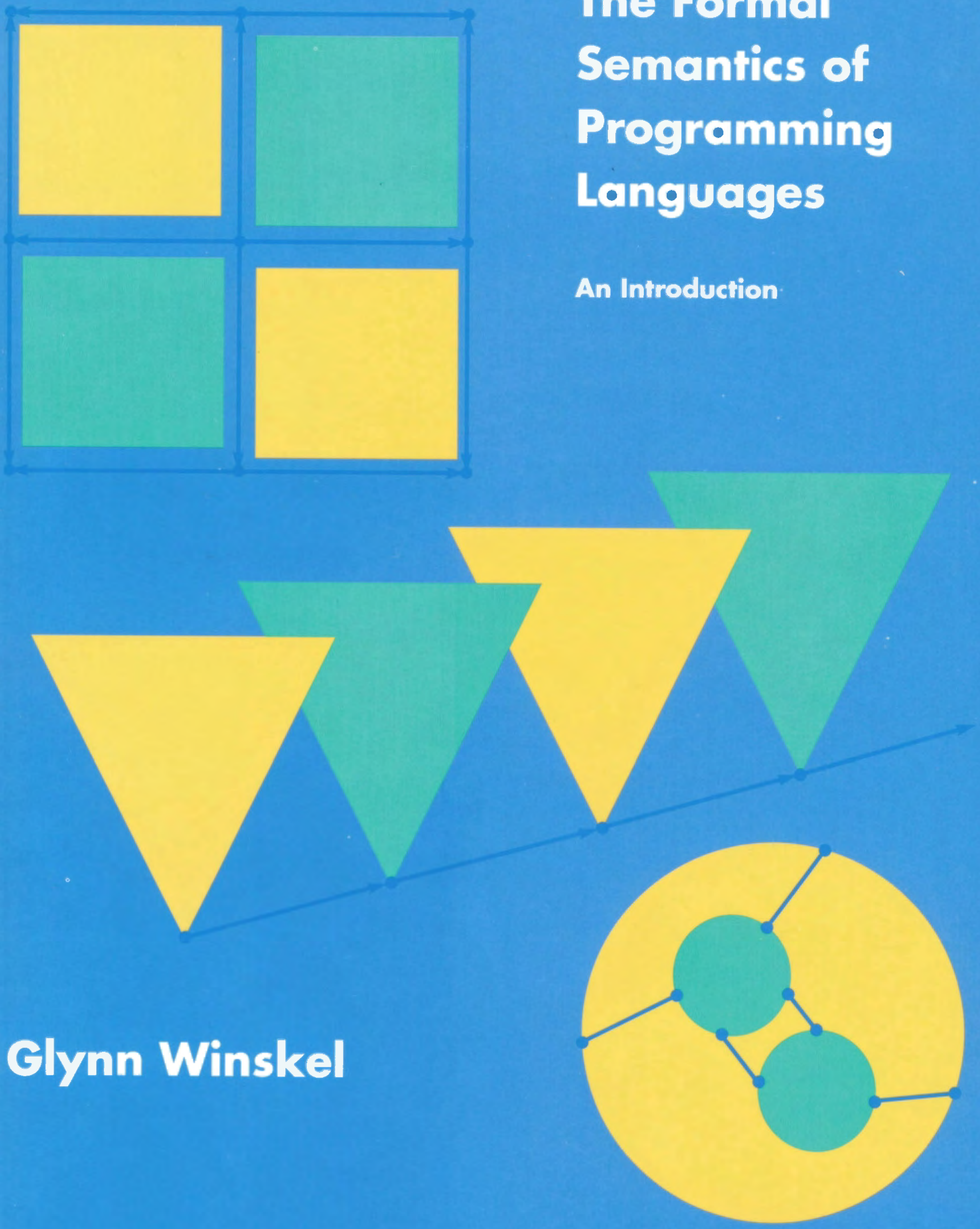
Foundations of Computing Series

The Formal Semantics of Programming Languages

An Introduction



Glynn Winskel

Rules for commands

Atomic commands:

$$\langle \mathbf{skip}, \sigma\rangle \to \sigma$$

$$\frac{\langle a, \sigma\rangle \to m}{\langle X := a, \sigma\rangle \to \sigma[m/X]}$$

Sequencing:

$$\frac{\langle c_0, \sigma\rangle \to \sigma'' \quad \langle c_1, \sigma''\rangle \to \sigma'}{\langle c_0; c_1, \sigma\rangle \to \sigma'}$$

Conditionals:

$$\frac{\langle b, \sigma\rangle \to \mathbf{true} \quad \langle c_0, \sigma\rangle \to \sigma'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{false} \quad \langle c_1, \sigma\rangle \to \sigma'}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \to \sigma'}$$

While-loops:

$$\frac{\langle b, \sigma\rangle \to \mathbf{false}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \to \sigma}$$

$$\frac{\langle b, \sigma\rangle \to \mathbf{true} \quad \langle c, \sigma\rangle \to \sigma'' \quad \langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma''\rangle \to \sigma'}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma\rangle \to \sigma'}$$

Again there is a natural equivalence relation on commands. Define

$$c_0 \sim c_1 \text{ iff } \forall \sigma, \sigma' \in \Sigma.\ \langle c_0, \sigma\rangle \to \sigma' \iff \langle c_1, \sigma\rangle \to \sigma'.$$

Exercise 2.6 Complete Exercise 2.2 of Section 2.2, by coding the rules for the evaluation of boolean expressions and execution of commands in Prolog and/or ML. □

Exercise 2.7 Let $w \equiv$ **while true do skip**. By considering the form of derivations, explain why, for any state σ, there is no state σ' such that $\langle w, \sigma\rangle \to \sigma'$. □

2.5 A simple proof

The operational semantics of the syntactic sets **Aexp**, **Bexp** and **Com** has been given using the same method. By means of rules we have specified the evaluation relations of both types of expressions and the execution relation of commands. All three relations are examples of the general notion of *transition relations*, or *transition systems*, in which the configurations are thought of as some kind of state and the relations as expressing possible transitions, or changes, between states. For instance, we can consider each of

$$\langle 3,\sigma\rangle \to 3, \quad \langle \mathbf{true},\sigma\rangle \to \mathbf{true}, \quad \langle X := 2,\sigma\rangle \to \sigma[2/X].$$

to be transitions.

Because the transition systems for **IMP** are given by rules, we have an elementary, but very useful, proof technique for proving properties of the operational semantics **IMP**.

As an illustration, consider the execution of a while-command $w \equiv \mathbf{while}\ b\ \mathbf{do}\ c$, with $b \in \mathbf{Bexp}, c \in \mathbf{Com}$, in a state σ. We expect that if b evaluates to **true** in σ then w executes as c followed by w again, and otherwise, in the case where b evaluates to **false**, that the execution of w terminates immediately with the state unchanged. This informal explanation of the execution of commands leads us to expect that for all states σ,σ'

$$\langle w,\sigma\rangle \to \sigma' \text{ iff } \langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else}\ \mathbf{skip},\sigma\rangle \to \sigma',$$

i.e., that the following proposition holds.

Proposition 2.8 *Let* $w \equiv \mathbf{while}\ b\ \mathbf{do}\ c$ *with* $b \in \mathbf{Bexp}, c \in \mathbf{Com}$. *Then*

$$w \sim \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else}\ \mathbf{skip}.$$

Proof: We want to show

$$\langle w,\sigma\rangle \to \sigma' \text{ iff } \langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else}\ \mathbf{skip},\sigma\rangle \to \sigma',$$

for all states σ,σ'.

"$\Rightarrow$": Suppose $\langle w,\sigma\rangle \to \sigma'$, for states σ,σ'. Then there must be a derivation of $\langle w,\sigma\rangle \to \sigma'$. Consider the possible forms such a derivation can take. Inspecting the rules for commands we see the final rule of the derivation is either

$$\frac{\langle b,\sigma\rangle \to \mathbf{false}}{\langle w,\sigma\rangle \to \sigma} \tag{1 $\Rightarrow$}$$

or

$$\frac{\langle b,\sigma\rangle \to \mathbf{true} \quad \langle c,\sigma\rangle \to \sigma'' \quad \langle w,\sigma''\rangle \to \sigma'}{\langle w,\sigma\rangle \to \sigma'} \tag{2 $\Rightarrow$}$$

In case (1 $\Rightarrow$), the derivation of $\langle w,\sigma\rangle \to \sigma'$ must have the form

$$\frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \mathbf{false}}}{\langle w,\sigma\rangle \to \sigma}$$

which includes a derivation of $\langle b,\sigma\rangle \to \mathbf{false}$. Using this derivation we can build the following derivation of $\langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else\ skip}, \sigma\rangle \to \sigma$:

$$\dfrac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \mathbf{false}} \qquad \dfrac{}{\langle \mathbf{skip},\sigma\rangle \to \sigma}}{\langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else\ skip}, \sigma\rangle \to \sigma}$$

In case (2 $\Rightarrow$), the derivation of $\langle w,\sigma\rangle \to \sigma'$ must take the form

$$\dfrac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \mathbf{true}} \qquad \dfrac{\vdots}{\langle c,\sigma\rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}}{\langle w,\sigma\rangle \to \sigma'}$$

which includes derivations of $\langle b,\sigma\rangle \to \mathbf{true}$, $\langle c,\sigma\rangle \to \sigma''$ and $\langle w,\sigma''\rangle \to \sigma'$. From these we can obtain a derivation of $\langle c; w,\sigma\rangle \to \sigma'$, *viz.*

$$\dfrac{\dfrac{\vdots}{\langle c,\sigma\rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}}{\langle c; w,\sigma\rangle \to \sigma'}$$

We can incorporate this into a derivation:

$$\dfrac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \mathbf{true}} \qquad \dfrac{\dfrac{\vdots}{\langle c,\sigma\rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}}{\langle c; w,\sigma\rangle \to \sigma'}}{\langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else\ skip}, \sigma\rangle \to \sigma'}$$

In either case, (1 $\Rightarrow$) or (2 $\Rightarrow$), we obtain a derivation of

$$\langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else\ skip}, \sigma\rangle \to \sigma'$$

from a derivation of

$$\langle w,\sigma\rangle \to \sigma'.$$

Thus

$$\langle w,\sigma\rangle \to \sigma' \text{ implies } \langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else\ skip}, \sigma\rangle \to \sigma',$$

for any states σ, σ'.
"$\Leftarrow$": We also want to show the converse, that $\langle \mathbf{if}\ b\ \mathbf{then}\ c; w\ \mathbf{else\ skip}, \sigma\rangle \to \sigma'$ implies $\langle w,\sigma\rangle \to \sigma'$, for all states σ, σ'.

Suppose $\langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma\rangle \to \sigma'$, for states σ, σ'. Then there is a derivation with one of two possible forms:

$$\frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \textbf{false}} \qquad \dfrac{}{\langle \textbf{skip},\sigma\rangle \to \sigma}}{\langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma\rangle \to \sigma} \tag{1 $\Leftarrow$}$$

$$\frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \textbf{true}} \qquad \dfrac{\vdots}{\langle c; w,\sigma\rangle \to \sigma'}}{\langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma\rangle \to \sigma'} \tag{2 $\Leftarrow$}$$

where in the first case, we also have $\sigma' = \sigma$, got by noting the fact that

$$\frac{}{\langle \textbf{skip},\sigma\rangle \to \sigma}$$

is the only possible derivation associated with **skip**.

From either derivation, (1 $\Leftarrow$) or (2 $\Leftarrow$), we can construct a derivation of $\langle w,\sigma\rangle \to \sigma'$. The second case, (2 $\Leftarrow$), is the more complicated. Derivation (2 $\Leftarrow$) includes a derivation of $\langle c; w,\sigma\rangle \to \sigma'$ which has to have the form

$$\frac{\dfrac{\vdots}{\langle c,\sigma\rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}}{\langle c; w,\sigma\rangle \to \sigma'}$$

for some state σ''. Using the derivations of $\langle c,\sigma\rangle \to \sigma''$ and $\langle w,\sigma''\rangle \to \sigma'$ with that for $\langle b,\sigma\rangle \to \textbf{true}$, we can produce the derivation

$$\frac{\dfrac{\vdots}{\langle b,\sigma\rangle \to \textbf{true}} \qquad \dfrac{\vdots}{\langle c,\sigma\rangle \to \sigma''} \qquad \dfrac{\vdots}{\langle w,\sigma''\rangle \to \sigma'}}{\langle w,\sigma\rangle \to \sigma'}$$

More directly, from the derivation (1 $\Leftarrow$), we can construct a derivation of $\langle w,\sigma\rangle \to \sigma'$ (How?).

Thus if $\langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma\rangle \to \sigma'$ then $\langle w,\sigma\rangle \to \sigma'$ for any states σ, σ'.

We can now conclude that

$$\langle w,\sigma\rangle \to \sigma' \text{ iff } \langle \textbf{if } b \textbf{ then } c; w \textbf{ else skip}, \sigma\rangle \to \sigma',$$

for all states σ, σ', and hence

$$w \sim \textbf{if } b \textbf{ then } c; w \textbf{ else skip}$$

as required. $\square$

This simple proof of the equivalence of while-command and its conditional unfolding exhibits an important technique: in order to prove a property of an operational semantics it is helpful to consider the various possible forms of derivations. This idea will be used again and again, though never again in such laborious detail. Later we shall meet other techniques, like "rule induction" which, in principle, can supplant the technique used here. The other techniques are more abstract however, and sometimes more confusing to apply. So keep in mind the technique of considering the forms of derivations when reasoning about operational semantics.

2.6 Alternative semantics

The evaluation relations

$$\langle a, \sigma \rangle \to n \text{ and } \langle b, \sigma \rangle \to t$$

specify the evaluation of expressions in rather large steps; given an expression and a state they yield a value directly. It is possible to give rules for evaluation which capture single steps in the evaluation of expressions. We could instead have defined an evaluation relation between pairs of configurations, taking *e.g.*

$$\langle a, \sigma \rangle \to_1 \langle a', \sigma' \rangle$$

to mean one step in the evaluation of a in state σ yields a' in state σ'. This intended meaning is formalised by taking rules such as the following to specify single steps in the left-to-right evaluation of sum.

$$\frac{\langle a_0, \sigma \rangle \to_1 \langle a_0', \sigma \rangle}{\langle a_0 + a_1, \sigma \rangle \to_1 \langle a_0' + a_1, \sigma \rangle}$$

$$\frac{\langle a_1, \sigma \rangle \to_1 \langle a_1', \sigma \rangle}{\langle n + a_1, \sigma \rangle \to_1 \langle n + a_1', \sigma \rangle}$$

$$\langle n + m, \sigma \rangle \to_1 \langle p, \sigma \rangle$$

where p is the sum of m and n.

Note how the rules formalise the intention to evaluate sums in a left-to-right sequential fashion. To spell out the meaning of the first sum rule above, it says: if one step in the evaluation of a_0 in state σ leads to a_0' in state σ then one step in the evaluation of $a_0 + a_1$ in state σ leads to $a_0' + a_1$ in state σ. So to evaluate a sum first evaluate the component expression of the sum and when this leads to a number evaluate the second component of the sum, and finally add the corresponding numerals (and we assume a mechanism to do this is given).

Exercise 2.9 Complete the task, begun above, of writing down the rules for $\rightarrow_1$, one step in the evaluation of integer and boolean expressions. What evaluation strategy have you adopted (left-to-right sequential or $\cdots$) ? □

We have chosen to define full execution of commands in particular states through a relation

$$\langle c, \sigma\rangle \rightarrow \sigma'$$

between command configurations. We could instead have based our explanation of the execution of commands on a relation expressing single steps in the execution. A single step relation between two command configurations

$$\langle c, \sigma\rangle \rightarrow_1 \langle c', \sigma'\rangle$$

means the execution of one instruction in c from state σ leads to the configuration in which it remains to execute c' in state σ'. For example,

$$\langle X := 5; Y := 1, \sigma\rangle \rightarrow_1 \langle Y := 1, \sigma[5/X]\rangle.$$

Of course, as this example makes clear, if we consider continuing the execution, we need some way to represent the fact that the command is empty. A configuration with no command left to execute can be represented by a state standing alone. So continuing the execution above we obtain

$$\langle X := 5; Y := 1, \sigma\rangle \rightarrow_1 \langle Y := 1, \sigma[5/X]\rangle \rightarrow_1 \sigma[5/X][1/Y].$$

We leave the detailed presentation of rules for the definition of this one-step execution relation to an exercise. But note there is some choice in what is regarded as a single step. If

$$\langle b, \sigma\rangle \rightarrow_1 \langle \mathbf{true}, \sigma\rangle$$

do we wish

$$\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \rightarrow_1 \langle c_0, \sigma\rangle$$

or

$$\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle \rightarrow_1 \langle \mathbf{if}\ \mathbf{true}\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma\rangle$$

to be a single step? For the language **IMP** these issues are not critical, but they become so in languages where commands can be executed in parallel; then different choices can effect the final states of execution sequences.