Nondeterminism and Total Correctness

Handout for “Semantiek” (2IT40)

Jos C.M. Baeten and Bas Luttik

May 10, 2006

In this handout we discuss the semantics of the *Guarded Command Language* (**GCL**). We transform the informal operational intuition associated with the language constructs in Kaldewaij's book [2] into a formal operational semantics. This has two important advantages. First, it can be used to *validate* the rules for total correctness, also presented in [2]; it allows us to formally prove that the rules for total correctness are correct with respect to the operational intuition. Second, when a formula is *not* valid, then the operational semantics gives us the opportunity to formally demonstrate this by exhibiting a *counterexample*, i.e., a computation starting from a state satisfying the precondition that does not terminate in a state satisfying the postcondition.

We begin, however, with introducing, in the context of **IMP**, another kind of operational semantics than the one associated with **IMP** in Winskel's book[1] [3]. It will be argued that this kind of operational semantics, which is called *small-step operational semantics*, is better suited to model nondeterminism if termination is an issue. This small-step operational semantics is briefly mentioned in [3], Section 2.6.

1 Small-step operational semantics

The operational semantics of **IMP** provided in [3] defines a transition relation

$$\rightarrow\ \subseteq (\mathbf{Com} \times \Sigma) \times \Sigma\ .$$

A transition $\langle c, \sigma\rangle \rightarrow \sigma'$ expresses that the full execution of command c in state σ (on an imaginary abstract machine) terminates in state σ'. Since one step corresponds to a full execution sequence, this kind of operational semantics is sometimes called *big-step operational semantics*.

The transition relation induced by a big-step operational semantics formalises the notion of execution in a rather crude way. It merely tells us whether the execution of a command in a state terminates, and what is the state upon termination. It does not tell us anything about the execution itself; e.g., whether it is long or short in terms of the number of executed atomic commands, and how many times the body of some while-loop inside c is executed.

Note that some quantitative information about the full execution represented by the transition $\langle c, \sigma\rangle \rightarrow \sigma'$ is implicitly available. For instance, the length of the execution (measured in terms of the number of atomic commands executed) can be recovered from the derivation of the transition by counting the number of applications of the rules for atomic commands [3, p. 20]. Similarly, the number of applications of the second rule for while-loops [3, p. 20] in the derivation reflects the number of times the body of a while-loop in c is executed.

[1]Chapters 1–6 are reproduced and used as the “dictaat” for the course “Semantiek”.

How does the big-step operational semantics of **IMP** reflect that a command like

$$\text{Loop} \equiv \textbf{while true do skip}$$

does *not* terminate? That c fails to terminate when executed from a state σ is reflected in the transition relation by the absence, for all states σ', of a transition

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

To prove the absence of a transition one needs to establish that there cannot be a derivation of it. Consider as an example Proposition 3.12 in [3] that states that $\langle \text{Loop}, \sigma \rangle \not\to \sigma'$ for all σ', and note that its proof consists of showing that there is no derivation with a transition $\langle \text{Loop}, \sigma \rangle \to \sigma'$ as conclusion.

This particular way of dealing with nontermination makes big-step operational semantics unsuitable for languages with nondeterminism, as we shall argue now. Suppose we add a nondeterministic choice construct **or** to **IMP**. The idea is that the execution of the command c_0 **or** c_1 involves a (nondeterministic) choice between executing c_0 and c_1: either c_0 is executed or c_1 is executed, but not both. Thus it may seem to be adequately formalised with the following operational rules:

$$\frac{\langle c_0, \sigma \rangle \to \sigma'}{\langle c_0 \ \textbf{or} \ c_1, \sigma \rangle \to \sigma'} \qquad \frac{\langle c_1, \sigma \rangle \to \sigma'}{\langle c_0 \ \textbf{or} \ c_1, \sigma \rangle \to \sigma'} \ \ .$$

However, these two rules in combination with the other rules of the big-step operational semantics of **IMP** lead to an equivalence that is arguably undesirable. This is illustrated in the following example.

Example 1.1 Suppose that the syntax of **IMP** is extended with the construct **or** and the big-step operational semantics of **IMP** is extended with the rules above. Then, by the first rule for **or**, for all states σ and σ'

$$\langle c, \sigma \rangle \to \sigma' \text{ implies } \langle c \ \textbf{or} \ \text{Loop}, \sigma \rangle \to \sigma' \ \ .$$

On the other hand, there is no state σ' such that $\langle \text{Loop}, \sigma \rangle \to \sigma'$. (Since we did not add any rules for while-loops, the proof of Proposition 3.12 in [3] still holds for the extension of **IMP** with nondeterminism that we are now considering.) Therefore it holds for all states σ and σ' that

$$\langle c \ \textbf{or} \ \text{Loop}, \sigma \rangle \to \sigma' \text{ implies } \langle c, \sigma \rangle \to \sigma' \ \ .$$

Thus, it follows that in this extension of **IMP** with nondeterministic choice

$$c \sim (c \ \textbf{or} \ \text{Loop}) \ \ .$$

Suppose that the command c in the above example is the following (correct) implementation in **IMP** of Euclid's algorithm to compute the greatest common divisor of two non-negative numbers (see also p. 33 of [3]):

$$\text{Euclid} \equiv \textbf{while} \ \neg(M = N) \ \textbf{do if} \ M \leq N \ \textbf{then} \ N := N - M \ \textbf{else} \ M := M - N \ \ .$$

That the command Euclid terminates if it is executed in a state σ with $\sigma(M), \sigma(N) \geq 1$, is usually considered part of the correctness of the algorithm. (In Winskel's book [3] this fact is proved as Theorem 3.10.) But then, according to this view, the program Euclid **or** Loop is not a correct implementation of Euclid's algorithm, for intuitively it may (nondeterministically) fail to terminate.

The moral of the story is that if termination of programs is an issue, then a big-step operational semantics is not suitable to model nondeterminism. The problem is that, rather than formalising the notion of execution proper, only the *existence* of a terminating execution is formalised. We shall now first discuss an alternative operational semantics for **IMP**, in which a step corresponds to the execution of a single atomic command, rather than to a full terminating execution. This kind of operational semantics is called *small-step* operational semantics. Then, we shall further discuss the extension of **IMP** with the nondeterministic choice construct mentioned above.

1.1 A small-step operational semantics of IMP

Recall that an operational semantics formalises our operational intuition about commands as a transition relation on the configurations of an imaginary abstract machine. Formally, a *configuration* is a pair

$$\langle c, \sigma \rangle$$

consisting of the command c to be executed, and the state σ in which the abstract machine currently resides. If a command terminates by executing its final step, what remains is just the final state. We call such a state standing by itself a *terminal configuration.*

Definition 1.2 ($\to_1$) The set of configurations **Conf** is defined as

$$\mathbf{Conf} = (\mathbf{Com} \times \Sigma) \cup \Sigma \ .$$

The *small-step transition relation* $\to_1$ associated with **IMP** is defined by the rules in Table 1.

Table 1: Small-step operational semantics of **IMP**

$$\langle \mathbf{skip}, \sigma \rangle \to_1 \sigma \qquad \frac{\langle a, \sigma \rangle \to n}{\langle X := a, \sigma \rangle \to_1 \sigma[n/X]}$$

$$\frac{\langle c_0, \sigma \rangle \to_1 \sigma'}{\langle c_0 \,;\, c_1, \sigma \rangle \to_1 \langle c_1, \sigma' \rangle} \qquad \frac{\langle c_0, \sigma \rangle \to_1 \langle c_0', \sigma' \rangle}{\langle c_0 \,;\, c_1, \sigma \rangle \to_1 \langle c_0' \,;\, c_1, \sigma' \rangle}$$

$$\frac{\langle b, \sigma \rangle \to \mathbf{true}}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma \rangle \to_1 \langle c_0, \sigma \rangle} \qquad \frac{\langle b, \sigma \rangle \to \mathbf{false}}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma \rangle \to_1 \langle c_1, \sigma \rangle}$$

$$\frac{\langle b, \sigma \rangle \to \mathbf{true}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \to_1 \langle c \,;\, \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle} \qquad \frac{\langle b, \sigma \rangle \to \mathbf{false}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \to_1 \sigma}$$

A transition $\langle c, \sigma \rangle \to_1 \langle c', \sigma' \rangle$ or $\langle c, \sigma \rangle \to_1 \sigma'$ represents one step in the execution of a command.

For any non-terminal configuration $\langle c, \sigma \rangle$ we write $\langle c, \sigma \rangle \not\to_1$ if there exists no configuration $\langle c', \sigma' \rangle$ or σ' such that $\langle c, \sigma \rangle \to_1 \langle c', \sigma' \rangle$ or $\langle c, \sigma \rangle \to_1 \sigma'$.

An execution is a sequence of transition steps until there is nothing left to execute; it is formally represented by the notion of *computation*, which we shall now define.

Definition 1.3 (i) A *transition sequence* is

- a finite sequence $\langle c_0, \sigma_0 \rangle, \ldots, \langle c_k, \sigma_k \rangle$ ($k \geq 0$) of configurations such that $\langle c_i, \sigma_i \rangle \to_1 \langle c_{i+1}, \sigma_{i+1} \rangle$ for all $0 \leq i < k$,
- a finite sequence $\langle c_0, \sigma_0 \rangle, \ldots, \langle c_k, \sigma_k \rangle, \sigma_{k+1}$ ($k \geq 0$) of configurations such that $\langle c_i, \sigma_i \rangle \to_1 \langle c_{i+1}, \sigma_{i+1} \rangle$ for all $0 \leq i < k$ and $\langle c_k, \sigma_k \rangle \to_1 \sigma_{k+1}$,
- or an infinite sequence $\langle c_0, \sigma_0 \rangle, \ldots, \langle c_k, \sigma_k \rangle, \ldots$ ($k \geq 0$) of configurations such that $\langle c_i, \sigma_i \rangle \to_1 \langle c_{i+1}, \sigma_{i+1} \rangle$ for all $i \geq 0$.

(ii) A *computation* of c starting in σ is a transition sequence that begins with the configuration $\langle c, \sigma \rangle$ and that cannot be extended. That is, a computation of c starting in σ is either a finite transition sequence of the form

$$\langle c, \sigma \rangle = \langle c_0, \sigma_0 \rangle \to_1 \cdots \to_1 \langle c_k, \sigma_k \rangle \not\to_1 \qquad (k \geq 0) \ ,$$

or

$$\langle c, \sigma \rangle = \langle c_0, \sigma_0 \rangle \to_1 \cdots \to_1 \langle c_k, \sigma_k \rangle \to_1 \sigma_{k+1} \qquad (k \geq 0) \ ,$$

or an infinite transition sequence of the form

$$\langle c, \sigma \rangle = \langle c_0, \sigma_0 \rangle \to_1 \cdots \to_1 \langle c_k, \sigma_k \rangle \to_1 \cdots \qquad (k \geq 0) \ .$$

(iii) We say that a command c can *diverge* from a state σ if there is an infinite computation of c starting in σ.

Example 1.4 Consider again the **IMP** command

$$\text{Euclid} \equiv \textbf{while } \neg(M = N) \textbf{ do if } M \leq N \textbf{ then } N := N - M \textbf{ else } M := M - N \ .$$

Here is a finite computation of c in a state σ_0 with $\sigma_0(M) = 9$ and $\sigma_0(N) = 6$:

$$\begin{aligned}
\langle \text{Euclid}, \sigma_0 \rangle &\to_1 \langle (\textbf{if } M \leq N \textbf{ then } N := N - M \textbf{ else } M := M - N) \,; \text{Euclid}, \sigma_0 \rangle \\
&\to_1 \langle M := M - N \,; \text{Euclid}, \sigma_0 \rangle \\
&\to_1 \langle \text{Euclid}, \sigma_0[3/M] \rangle \\
&\to_1 \langle (\textbf{if } M \leq N \textbf{ then } N := N - M \textbf{ else } M := M - N) \,; \text{Euclid}, \sigma_0[3/M] \rangle \\
&\to_1 \langle N := N - M \,; \text{Euclid}, \sigma_0[3/M] \rangle \\
&\to_1 \langle \text{Euclid}, \sigma_0[3/M][3/N] \rangle \\
&\to_1 \sigma_0[3/M][3/N] \ .
\end{aligned}$$

The execution of Euclid in a state σ_1 with $\sigma_1(M) = -1$ and $\sigma_1(N) = 0$ gives rise to an infinite computation of the form:

$$\begin{aligned}
\langle \text{Euclid}, \sigma_1 \rangle &\to_1 \langle (\textbf{if } M \leq N \textbf{ then } N := N - M \textbf{ else } M := M - N) \,; \text{Euclid}, \sigma_1 \rangle \\
&\to_1 \langle N := N - M \,; \text{Euclid}, \sigma_1 \rangle \\
&\to_1 \langle \text{Euclid}, \sigma_1[1/N] \rangle \\
&\to_1 \langle (\textbf{if } M \leq N \textbf{ then } N := N - M \textbf{ else } M := M - N) \,; \text{Euclid}, \sigma_1[1/N] \rangle \\
&\to_1 \langle N := N - M \,; \text{Euclid}, \sigma_1[1/N] \rangle \\
&\to_1 \langle \text{Euclid}, \sigma_1[2/N] \rangle \\
&\to_1 \cdots \\
&\to_1 \langle \text{Euclid}, \sigma_1[3/N] \rangle \\
&\to_1 \cdots \ .
\end{aligned}$$

Thus we see that Euclid can diverge from σ_1.

Exercise 1.1 Consider the following **IMP** command:

```
$c \equiv Z := 0;\ N := Y$
    **while** $\neg(N \leq 0)$ **do**
        $Z := Z + X;$
        $N := N - 1$   .
```

Exhibit the computation of c starting in a state σ with $\sigma(X) = 4$ and $\sigma(Y) = 3$.

Exercise 1.2 Show that the command Loop can diverge from any state σ.

Call a non-terminal configuration $\langle c, \sigma \rangle$ *blocking* if $\langle c, \sigma \rangle \not\to_1$. Later, when we discuss the semantics of the *Guarded Command Language* (**GCL**), blocking configurations will arise. According to the following lemma there are no blocking configurations in the context of **IMP**.

Lemma 1.5 (Absence of Blocking) For every **IMP** command c and for every state σ there exists a configuration $\langle c', \sigma' \rangle$ or σ' such that $\langle c, \sigma \rangle \to_1 \langle c', \sigma' \rangle$ or $\langle c, \sigma \rangle \to_1 \sigma'$.

Proof. The proof is a straightforward induction on the structure of c. □

Since by Lemma 1.5 there are no blocking configurations, every *finite* computation of an **IMP** command c starting in σ is of the form

$$\langle c, \sigma \rangle = \langle c_0, \sigma_0 \rangle \to_1 \cdots \to_1 \langle c_k, \sigma_k \rangle \to_1 \sigma_{k+1} \ .$$

That execution of **IMP** commands is deterministic now takes the following form.

Lemma 1.6 (Determinism) Every command c has from every state σ precisely one computation.

Proof. By structural induction on commands it is easily established that for every configuration $\langle c, \sigma \rangle$ there is at most one command c' and state σ' such that $\langle c, \sigma \rangle \to_1 \langle c', \sigma' \rangle$ or $\langle c, \sigma \rangle \to_1 \sigma'$ holds. Hence, the lemma follows. □

1.2 Extending IMP with nondeterminism

Now consider again the extension of **IMP** with the nondeterministic choice construct **or**, adding the following two rules to the small-step operational semantics of **IMP**:

$$\langle c_0 \textbf{ or } c_1, \sigma \rangle \to_1 \langle c_0, \sigma \rangle \qquad \langle c_0 \textbf{ or } c_1, \sigma \rangle \to_1 \langle c_1, \sigma \rangle \ .$$

Recall that $\to_1^*$ denotes the transitive, reflexive closure of $\to_1$ (cf. p. 10 of [3]). Based on the notion of computation introduced in the previous subsection, we can now redefine the notion of equivalence of **IMP** commands so that it preserves divergence:

Definition 1.7 Two **IMP** commands c_0 and c_1 are equivalent (notation: $c_0 \sim c_1$) if for all states σ and σ':

1. $\langle c_0, \sigma \rangle \to_1^* \sigma'$ iff $\langle c_1, \sigma \rangle \to_1^* \sigma'$; and

2. c_0 can diverge from σ iff c_1 can diverge from σ.

Note that with this new notion of equivalence

$$\text{Euclid} \not\sim (\text{Euclid}\ \mathbf{or}\ \text{Loop})\ ,$$

for in a state σ such that $\sigma(M), \sigma(N) \geq 1$ the left-hand side cannot diverge, whereas the right-hand side can.

Exercise 1.3 Prove the following equivalences:

(a) $c_0\ \mathbf{or}\ c_1 \sim c_1\ \mathbf{or}\ c_0$;

(b) $c_0\ \mathbf{or}\ (c_1\ \mathbf{or}\ c_2) \sim (c_0\ \mathbf{or}\ c_1)\ \mathbf{or}\ c_2$;

(c) $c\ \mathbf{or}\ c \sim c$.

Exercise 1.4 Prove the following equivalences:

(a) $c_0\ ;\ (c_1\ \mathbf{or}\ c_2) \sim (c_0\ ;\ c_1)\ \mathbf{or}\ (c_0\ ;\ c_2)$;

(b) $(c_0\ \mathbf{or}\ c_1)\ ;\ c_2 \sim (c_0\ ;\ c_2)\ \mathbf{or}\ (c_1\ ;\ c_2)$.

2 The Guarded Command Language

Recall Dijkstra's *Guarded Command Language* (**GCL**) as it is, e.g., presented in Kaldewaij's book [2]; it has the following *commands* (called "statements" in [2]):

skip: **skip** is a command; its execution has no effect on the current state;

assignment: $X := a$ is a command; its execution has the effect of storing the value of a in location X;

catenation: if c_0 and c_1 are commands, then $c_0\ ;\ c_1$ is also a command; its operational interpretation according to Kaldewaij [2] is: "first c_0 is executed after which c_1 is executed".

selection: for each $n \geq 0$, if $b_0, \ldots, b_n$ are boolean expressions (guards) and $c_0, \ldots, c_n$ are commands, then $\mathbf{if}\ b_0 \rightarrow c_0\ [\!]\ \cdots\ [\!]\ b_n \rightarrow c_n\ \mathbf{fi}$ is a command; its operational interpretation according to Kaldewaij [2] is:

> "Upon execution of a selection all guards are evaluated. If none of the guards evaluates to true then execution of the selection aborts, otherwise one of the guards that has the value true is chosen *non-deterministically* and the corresponding statement is executed."

repetition: for each $n \geq 0$, if $b_0, \ldots, b_n$ are boolean expressions (guards) and $c_0, \ldots, c_n$ are commands, then $\mathbf{do}\ b_0 \rightarrow c_0\ [\!]\ \cdots\ [\!]\ b_n \rightarrow c_n\ \mathbf{od}$ is a command; its operational interpretation according to Kaldewaij [2] is:

> "Upon execution of a repetition all guards are evaluated. If all guards evaluate to false then skip is executed. Otherwise one of the guards that has the value true is chosen *non-deterministically* and the corresponding statement is executed after which the repetition is executed again."

We denote the set of all **GCL** commands by **Com**; for ease of reference, we summarise the abstract syntax of **GCL** by means of a grammar for $c \in \mathbf{Com}$, using an auxiliary syntactic category of *guarded commands* gc:

$$\begin{array}{rl} c ::= & \mathbf{skip} \mid X := a \mid c_0 \,;\, c_1 \mid \mathbf{if}\ gc\ \mathbf{fi} \mid \mathbf{do}\ gc\ \mathbf{od} \\ gc ::= & b_0 \to c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \to c_n \qquad (n \geq 0) \end{array}$$

where X is a location, a is an arithmetic expression, and b_i is a boolean expression for $i = 0, \ldots, n$.

Remark 2.1

1. Note that we have redefined the symbol **Com**; it used to denote the set of **IMP** commands, and now it denotes the set of **GCL** commands.
2. For arithmetic and boolean expressions, and also for assertions, which will be needed shortly, we presuppose in this handout the syntax and semantics provided in Winskel's book [3].

The following example illustrates the syntax of **GCL**.

Example 2.2

1. To illustrate the selection construct, here is an example of a command that assigns the minimum of two locations X and Y to a location Z:

   $$\begin{array}{rl} \mathit{MIN} \equiv & \mathbf{if} \\ & \quad X \leq Y \to Z := X \\ & [\!] \\ & \quad Y \leq X \to Z := Y \\ & \mathbf{fi} \end{array}$$

   Note that if $X = Y$ then both guards are true, so according to Kaldewaij's operational interpretation a nondeterministic choice for the execution of $Z{:=}X$ or the execution of $Z := Y$ is made.

2. To illustrate the repetition construct, here is a nice formulation in **GCL** of Euclid's algorithm to compute the greatest common divisor of two numbers:

   $$\begin{array}{rl} \mathit{GCD} \equiv & \mathbf{do} \\ & \quad M > N \to M := M - N \\ & [\!] \\ & \quad N > M \to N := N - M \\ & \mathbf{od} \end{array}$$

   Note that if $M = N$, then neither of the guards is true, so according to Kaldewaij's operational interpretation skip is executed.

Kaldewaij does not formalise the informal operational intuition he assigns to **GCL** commands in his book [2]. But he does provide a proof system by which the correctness of **GCL** commands can be formally established. The basic ingredients of this proof system are formulas of the form

$$\{A\}c\{B\} \ ,$$

with $A, B \in \mathbf{Assn}$ and $c \in \mathbf{Com}$. Kaldewaij assigns the following operational interpretation to them:

> "All executions of c starting in a state satisfying A terminate in a state satisfying B."

This operational interpretation of $\{A\}c\{B\}$ asserts *total correctness*, and it should be contrasted with the interpretation in Winskel's book [3] according to which the formula $\{A\}c\{B\}$ asserts *partial correctness*. The subtle but crucial difference between the two notions is the way termination is dealt with: total correctness *requires* termination, whereas partial correctness *assumes* it. (The reader is advised to consult p. 79 in [3] for a more elaborate discussion of the difference between partial and total correctness!)

Table 2: The proof system TC

(TC1) $\{A\}\mathbf{skip}\{A\}$ (TC2) $\{A[a/X]\}X := a\{A\}$

(TC3) $$\frac{\{A\}c_0\{C\} \quad \{C\}c_1\{B\}}{\{A\}c_0\,;c_1\{B\}}$$

(TC4) $$\frac{\models (A \Rightarrow (b_0 \vee \cdots \vee b_n)) \quad \{A \wedge b_i\}c_i\{B\} \quad (i \in \{0,\ldots,n\})}{\{A\}\mathbf{if}\ b_0 \to c_0 \ [\!] \cdots [\!]\ b_n \to c_n\ \mathbf{fi}\{B\}}$$

(TC5) $$\frac{\models (A \Rightarrow 0 \leq t) \quad \{A \wedge b_i\}c_i\{A\} \quad \{A \wedge b_i \wedge t = z\}c_i\{\neg(z \leq t)\} \quad (i \in \{0,\ldots,n\})}{\{A\}\mathbf{do}\ b_0 \to c_0 \ [\!] \cdots [\!]\ b_n \to c_n\ \mathbf{od}\{A \wedge \neg(b_0 \vee \cdots \vee b_n)\}}$$

where t is an integer expression and z is an integer variable that does not occur in A or t

(TC6) $$\frac{\models (A \Rightarrow A') \quad \{A'\}c\{B'\} \quad \models (B' \Rightarrow B)}{\{A\}c\{B\}}$$

The proof system TC for total correctness consists of the rules listed in Table 2. These rules are very similar to Kaldewaij's; the most notable deviation is that Kaldewaij's system includes additional rules that formalise the reasoning about assertions, whereas TC does not include such rules. Note that the rules of TC make direct reference to the *validity* of assertions.

Definition 2.3 (Derivability) A formula $\{A\}c\{B\}$ is called a *theorem* of TC if it is the conclusion of a derivation within TC (see Table 2). If $\{A\}c\{B\}$ is a theorem, we write

$$\vdash_{\mathsf{TC}} \{A\}c\{B\} \ .$$

Example 2.4 1. The correctness of the command *MIN* of Example 2.2 can be substantiated by showing that $\vdash_{\mathsf{TC}} \{\mathbf{true}\}\mathit{MIN}\{Z = \min(X, Y)\}$.

Note that by (TC2)

$$\{X = \min(X, Y)\}Z := X\{Z = \min(X, Y)\} \ ,$$

and since $\models (\mathbf{true} \wedge X \leq Y \Rightarrow X = \min(X, Y))$ it follows by (TC6) that

$$\{\mathbf{true} \wedge X \leq Y\}Z := X\{Z = \min(X, Y)\} \ .$$

Similarly one obtains a derivation of

$$\{\mathbf{true} \wedge Y \leq X\}Z := Y\{Z = \min(X, Y)\} \ .$$

Since clearly $\models (\textbf{true} \Rightarrow (X \leq Y \vee Y \leq X))$ we can now apply (TC4) and conclude that $\{\textbf{true}\}\mathit{MIN}\{Z = \min(X, Y)\}$ is indeed a theorem of TC.

2. The correctness of the command *GCD* of Example 2.2 can be substantiated by showing that $\vdash_{\mathsf{TC}} \{M = m \wedge N = n \wedge 1 \leq m \wedge 1 \leq n\}\mathit{GCD}\{M = N \wedge N = \gcd(m, n)\}$. We shall not give a complete derivation, but concentrate on the application of (TC5); in particular we consider the rôle of the bound function t. Take as invariant the assertion

   $$A \equiv (\gcd(M, N) = \gcd(m, n) \wedge 1 \leq M \wedge 1 \leq N) \ ,$$

   and as bound function the integer expression $t \equiv M + N$. We want to conclude by an application of (TC5) that $\{A\}\mathit{GCD}\{A \wedge \neg(M > N \vee N > M)\}$, so let us consider the premises of this application.

   That $\models (A \Rightarrow 0 \leq t)$ can be established by means of elementary mathematical reasoning. Furthermore, by means of (TC2) and (TC6) one can derive the premises

   $$\{A \wedge M > N\}M := M - N\{A\} \text{ and}$$
   $$\{A \wedge N > M\}N := N - M\{A\} \ .$$

   We consider the premise $\{A \wedge M > N \wedge t = z\}M := M - N\{\neg(z \leq t)\}$ in more detail. Note that $\{\neg(z \leq t[(M - N)/M])\}M := M - N\{\neg(z \leq t)\}$ according to (TC2). To prove that $\models (A \wedge M > N \wedge t = z \Rightarrow \neg(z \leq t[(M - N)/M]))$, let I be an interpretation and let σ be a state such that $\sigma \models^I A \wedge M > N \wedge t = z$, then $\sigma(M) > \sigma(N) \geq 1$, so $\sigma(t[(M - N)/M]) = \sigma((M - N) + N) = \sigma(M) < \sigma(t) = I(z)$, and hence the validity of the assertion $(A \wedge M > N \wedge t = z \Rightarrow \neg(z \leq t[(M - N)/M]))$ follows. So, by an application of (TC6) the premise

   $$\{A \wedge M > N \wedge t = z\}M := M - N\{\neg(z \leq t)\}$$

   can be derived, and there is a similar derivation for the premise

   $$\{A \wedge N > M \wedge t = z\}N := N - M\{\neg(z \leq t)\} \ .$$

   Now that the premises of the desired application of (TC5) are all derived, we can infer its conclusion:

   $$\{A\}\mathit{GCD}\{A \wedge \neg(M > N \vee N > M)\} \ .$$

   Since

   $$\models (M = m \wedge N = n \wedge 1 \leq m \wedge 1 \leq n \Rightarrow A) \text{ and}$$
   $$\models (A \wedge \neg(M > N \vee N > M) \Rightarrow M = N \wedge N = \gcd(m, n))$$

   it then follows by another application of (TC6) that

   $$\{M = m \wedge N = n \wedge 1 \leq m \wedge 1 \leq n\}\mathit{GCD}\{M = N \wedge N = \gcd(m, n)\}$$

   is indeed a theorem of TC.

Remark 2.5 The reasoning in the above example may at first seem informal, but it isn't really; it conveys all the information needed to construct decent derivations within TC. In Figure 1, the information provided in Example 2.4.1 with respect to the TC-derivation of the formula $\{\textbf{true}\}\mathit{MIN}\{Z = \min(X, Y)\}$ is presented in the

$$\dfrac{\models (A \Rightarrow (b_1 \vee b_2)) \qquad \dfrac{\models (A \wedge b_1 \Rightarrow B[X/Z]) \quad \dfrac{}{\{B[X/Z]\}Z := X\{B\}}\,(\mathsf{TC2})}{\{A \wedge b_1\}Z := X\{B\}}\,(\mathsf{TC6}) \qquad \dfrac{\models (A \wedge b_2 \Rightarrow B[Y/Z]) \quad \dfrac{}{\{B[Y/Z]\}Z := Y\{B\}}\,(\mathsf{TC2})}{\{A \wedge b_2\}Z := Y\{B\}}\,(\mathsf{TC6})}{\{A\}MIN\{B\}}\,(\mathsf{TC4})$$

Used abbreviations: $A \equiv \mathbf{true}$, $B \equiv (Z = \min(X, Y))$, $b_1 \equiv (X \leq Y)$, and $b_2 \equiv (Y \leq X)$.

Figure 1: A formal derivation that proves that the command *MIN* computes the minimum of two numbers.

tree-like manner that we got used to when discussing the operational semantics of **IMP**.

The purpose of the system TC is only to formalise those parts of the reasoning about **GCL** programs that has to do with the effect of the execution of its commands. Of course, proving the correctness of a **GCL** program also involves a lot of 'logical reasoning', reasoning about the validity of assertions. For instance, it is essential for the TC-derivation in Figure 1 that the assertion $(A \Rightarrow (b_1 \vee b_2))$ is valid; the derivation in Figure 1 just doesn't explain *why* it is valid. Imagine that the validity of the assertion has been demonstrated on a piece of scrap paper. Afterwards the proof on the scrap paper is omitted, because it would only distract from the essence of the proof.

To increase confidence in the correctness of the proof system TC, we are now going to validate it by formalising a correspondence with the operational interpretation. We proceed as follows. First we formalise the operational interpretation assigned to **GCL** commands by giving an operational semantics of **GCL**. This will then allow us to formalise the operational interpretation assigned to total correctness assertions. Thus, a notion of validity for total correctness assertion arises, and it can be shown that TC is sound with respect to this notion of validity. (Appendix A to this handout, which is available from the website [1], contains the full soundness proof. It is considered out of the scope of the course "Semantiek" (2IT40), so it need not be studied.)

2.1 A small-step operational semantics of GCL

Note that the informal operational interpretation given in Kaldewaij's book [2] is conceptually close to a big-step operational semantics, for it is formulated in terms of (complete) executions. However, it was argued in Section 1 that if termination is an issue and the language has nondeterminism, then a small-step operational semantics is a better choice. And it is an issue, since we are now interested in total correctness. So, let us present a small-step operational semantics of **GCL**.

Definition 2.6 ($\rightarrow_1$) The set of configurations **Conf** is defined as

$$\mathbf{Conf} = (\mathbf{Com} \times \Sigma) \cup \Sigma \ .$$

The *small-step transition relation* $\rightarrow_1$ associated with **GCL** is defined by the rules in Table 3.

Conditionals and while-loops of **IMP** can easily be modeled in **GCL**. We reintroduce them as abbreviations of **GCL** commands:

$$\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1 \equiv \mathbf{if}\ b \rightarrow c_0 \ [\!]\ \neg b \rightarrow c_1\ \mathbf{fi}\ \ ;\ \text{and}$$
$$\mathbf{while}\ b\ \mathbf{do}\ c \equiv \mathbf{do}\ b \rightarrow c\ \mathbf{od}\ \ .$$

Table 3: Small-step operational semantics of **GCL**

(O1) $\langle \mathbf{skip}, \sigma\rangle \to_1 \sigma$

(O2) $$\frac{\langle a, \sigma\rangle \to n}{\langle X := a, \sigma\rangle \to_1 \sigma[n/X]}$$

(O3) $$\frac{\langle c_0, \sigma\rangle \to_1 \sigma'}{\langle c_0\,; c_1, \sigma\rangle \to_1 \langle c_1, \sigma'\rangle}$$ (O4) $$\frac{\langle c_0, \sigma\rangle \to_1 \langle c_0', \sigma'\rangle}{\langle c_0\,; c_1, \sigma\rangle \to_1 \langle c_0'\,; c_1, \sigma'\rangle}$$

($\mathsf{O5}_{n,i}$) $$\frac{\langle b_i, \sigma\rangle \to \mathbf{true}}{\langle \mathbf{if}\ b_0 \to c_0 \,[\!]\, \cdots \,[\!]\, b_n \to c_n\ \mathbf{fi}, \sigma\rangle \to_1 \langle c_i, \sigma\rangle}$$

($\mathsf{O6}_{n}$) $$\frac{\langle b_i, \sigma\rangle \to \mathbf{false} \text{ for all } i = 1, \ldots, n}{\langle \mathbf{do}\ b_0 \to c_0 \,[\!]\, \cdots \,[\!]\, b_n \to c_n\ \mathbf{od}, \sigma\rangle \to_1 \sigma}$$

($\mathsf{O7}_{n,i}$) $$\frac{\langle b_i, \sigma\rangle \to \mathbf{true}}{\langle \mathbf{do}\ b_0 \to c_0 \,[\!]\, \cdots \,[\!]\, b_n \to c_n\ \mathbf{od}, \sigma\rangle \to_1 \langle c_i\,; \mathbf{do}\ b_0 \to c_0 \,[\!]\, \cdots \,[\!]\, b_n \to c_n\ \mathbf{od}, \sigma\rangle}$$

The definitions of "blocking configuration", "transition sequence", "computation" and "divergence" provided in Section 1 carry over unchanged to the current setting.

In the previous section we made a distinction between terminal configurations (*i.e.*, configurations just consisting of a state) and blocking configurations (*i.e.*, configurations of the form $\langle c, \sigma\rangle$ with $\langle c, \sigma\rangle \not\to_1$). Blocking configurations did not arise in **IMP** (see Lemma 1.5), but they do arise in **GCL**, e.g. the configuration $\langle \mathbf{if}\ \mathbf{false} \to \mathbf{skip}\ \mathbf{fi}, \sigma\rangle$ is blocking for every state σ. In general, we have that $\langle \mathbf{if}\ b_0 \to c_0 \,[\!]\, \cdots \,[\!]\, b_n \to c_n\ \mathbf{fi}, \sigma\rangle$ is blocking iff $\langle b_i, \sigma\rangle \to \mathbf{false}$ for all $1 \leq i \leq n$. We also want to distinguish computations that end in a terminal configuration from computations that end in a blocking configuration; the latter we shall not consider successful.

Definition 2.7 (Failure) We say that a **GCL** command c can *fail* from a state σ if there is a computation of c starting in σ that ends in a blocking configuration.

Example 2.8

1. If MIN is the command defined in Example 2.2 and $\langle \mathit{MIN}, \sigma\rangle \to_1^* \sigma'$, then $\sigma'(Z) < \sigma'(Y)$ only if $\sigma(X) < \sigma(Y)$, so the command

   $\mathit{MIN}; \mathbf{if}\ Z < Y \to \mathbf{skip}\ \mathbf{fi}$

   can fail from σ if $\sigma(X) \geq \sigma(Y)$.

2. If GCD is the command defined in Example 2.2, σ is a state such that $\sigma(M), \sigma(N) \geq 0$, and $\langle \mathit{GCD}, \sigma\rangle \to_1^* \sigma'$, then $\sigma'(M) = \sigma'(N)$.[2] The configuration

   $\langle \mathbf{if}\ \neg(M = N) \to \mathbf{skip}\ \mathbf{fi}, \sigma'\rangle$

   is blocking if $\sigma'(M) \neq \sigma'(N)$. So, the command

   $\mathit{GCD}\,; \mathbf{if}\ \neg(M = N) \to \mathbf{skip}\ \mathbf{fi}$

[2] Actually, we will not know for sure that this is true until we have validated TC in Section 2.2, so that we can conclude $\sigma'(M) = \sigma'(N)$ from Example 2.4. For now, let us just assume that it holds.

can fail from every state σ such that $\sigma(M), \sigma(N) \geq 0$.

Based on the small-step operational semantics for **GCL** we are now going to define two semantic mappings. The first one, denoted by $\mathcal{O}$, associates with a **GCL** command c and a state σ the set of all states σ' that may result from a successful computation of c starting in σ. The second one, denoted by $\mathcal{O}_{tot}$, adds $\perp$ to this set if c can fail or diverge from σ. We treat $\perp$ as a special kind of state and we put

$$\Sigma_\perp = \Sigma \cup \{\perp\} \ .$$

Definition 2.9 Let c be a **GCL** command.

(i) The *partial correctness semantics* $\mathcal{O}[\![c]\!] : \Sigma \to \mathcal{P}ow(\Sigma)$ of c is defined by

$$\mathcal{O}[\![c]\!]\sigma = \{\sigma' \in \Sigma \mid \langle c, \sigma\rangle \to_1^* \sigma'\} \ .$$

(ii) The *total correctness semantics* $\mathcal{O}_{tot}[\![c]\!] : \Sigma \to \mathcal{P}ow(\Sigma_\perp)$ of c is defined by

$$\mathcal{O}_{tot}[\![c]\!]\sigma = \mathcal{O}[\![c]\!]\sigma \cup \{\perp \mid \text{if } c \text{ can fail or diverge from } \sigma\} \ .$$

Exercise 2.1 Prove the following equivalences:

(a) $\mathcal{O}_{tot}[\![\textbf{if } b_0 \to c_0 \mathbin{[\!]} b_1 \to c_1 \textbf{ fi}]\!] = \mathcal{O}_{tot}[\![\textbf{if } b_1 \to c_1 \mathbin{[\!]} b_0 \to c_0 \textbf{ fi}]\!]$;

(b)* $\mathcal{O}_{tot}[\![\textbf{do } b_0 \to c_0 \mathbin{[\!]} b_1 \to c_1 \textbf{ od}]\!] = \mathcal{O}_{tot}[\![\textbf{do } b_1 \to c_1 \mathbin{[\!]} b_0 \to c_0 \textbf{ od}]\!]$.

Exercise 2.2 Prove the following equivalences:

(a) $\mathcal{O}_{tot}[\![\textbf{do } b \to c \textbf{ od}]\!] = \mathcal{O}_{tot}[\![\textbf{if } b \to (c\,; \textbf{do } b \to c \textbf{ od}) \mathbin{[\!]} \neg b \to \textbf{skip fi}]\!]$;

(b)* $\mathcal{O}_{tot}[\![\textbf{do } b_0 \to c_0 \mathbin{[\!]} b_1 \to c_1 \textbf{ od}]\!] = \mathcal{O}_{tot}[\![\textbf{do } b_0 \vee b_1 \to \textbf{if } b_0 \to c_0 \mathbin{[\!]} b_1 \to c_1 \textbf{ fi od}]\!]$.

2.2 Validation of TC

We can now also formalise the operational interpretation assigned to formulas $\{A\}c\{B\}$. We introduce two notions of validity; the first corresponds to partial correctness and the second corresponds to total correctness; the latter will be used to validate TC.

Definition 2.10 (i) We define the *satisfaction relation* $\models^I$ for partial correctness with respect to an interpretation I by

$$\sigma \models^I \{A\}c\{B\} \text{ iff } \left[\sigma \models^I A \Rightarrow \forall \sigma' \in \mathcal{O}[\![c]\!]\sigma.\ \sigma' \models^I B\right] \ .$$

If $\sigma \models^I \{A\}c\{B\}$ for all states σ and for all interpretations I, then we say that $\{A\}c\{B\}$ is *valid in the sense of partial correctness* (notation: $\models \{A\}c\{B\}$).

(ii) We define the *satisfaction relation* $\models^I_{tot}$ for partial correctness with respect to an interpretation I by

$$\sigma \models^I_{tot} \{A\}c\{B\} \text{ iff } \left[\sigma \models^I A \Rightarrow (\perp \notin \mathcal{O}_{tot}[\![c]\!]\sigma \ \&\ \forall \sigma' \in \mathcal{O}_{tot}[\![c]\!]\sigma.\ \sigma' \models^I_{tot} B)\right] \ .$$

If $\sigma \models^I_{tot} \{A\}c\{B\}$ for all states σ and for all interpretations I, then we say that $\{A\}c\{B\}$ is *valid in the sense of total correctness* (notation: $\models_{tot} \{A\}c\{B\}$).

Example 2.11 Consider the command GCD from Example 2.2.

Note that every finite computation of GCD from a state σ that ends in a terminal configuration σ' with $\sigma'(M) = \sigma'(N)$ is of the form

$$\langle GCD, \sigma\rangle \rightarrow_1 \cdots \rightarrow_1 \langle GCD, \sigma'\rangle \rightarrow_1 \sigma' \ ,$$

and the last transition of this computation is justified by an application of ($\mathsf{O6}_n$). This means that $\langle M < N, \sigma'\rangle \rightarrow \mathbf{false}$ and $\langle N < M, \sigma'\rangle \rightarrow \mathbf{false}$. It follows that $\sigma'(M) = \sigma'(N)$, and hence, for every interpretation I, $\sigma' \models^I M = N$. Thus we find that for all states σ and for all interpretations I

$$\sigma \models^I \mathbf{true} \Rightarrow \forall \sigma' \in \mathcal{O}[\![GCD]\!]\sigma.\ \sigma' \models^I M = N \ ,$$

so the formula $\{\mathbf{true}\}GCD\{M = N\}$ is valid in the sense of partial correctness.

On the other hand, if σ_0 is a state such that $\sigma_0(M) = 0$ and $\sigma_0(N) = -1$, then GCD has an infinite computation

$$\langle GCD, \sigma_0\rangle \rightarrow_1 \langle GCD, \sigma_1\rangle \rightarrow_1 \cdots \rightarrow_1 \langle GCD, \sigma_k\rangle \rightarrow_1 \langle GCD, \sigma_{k+1}\rangle \rightarrow_1 \cdots \qquad (k \geq 0) \ ,$$

with $\sigma_{k+1} = \sigma_k[(\sigma_k(M) + 1)/M]$. Hence, $\perp \in \mathcal{O}_{tot}[\![GCD]\!]\sigma$, so

$$\sigma \not\models^I_{tot} \{\mathbf{true}\}GCD\{M = N\} \ ,$$

so the formula $\{\mathbf{true}\}GCD\{M = N\}$ is not valid in the sense of total correctness.

It is important to note that in the above example we have actually *proved* that the formula $\{\mathbf{true}\}GCD\{M = N\}$ is not valid in the sense of total correctness. Our proof consists of exhibiting a *counterexample*, which is in this case a computation that starts in a state satisfying the assertion **true**, but does not terminate in a state satisfying the assertion $M = N$ (in fact, it does not terminate at all).

The operational semantics that we associated with **GCL** has enabled us to formally define when a formula $\{A\}c\{B\}$ is valid in the sense of total correctness. It can now be proved that the proof system TC is indeed sound with respect to this notion of validity.

Theorem 2.12 (Soundness) If $\vdash_{\mathsf{TC}} \{A\}c\{B\}$, then $\models_{tot} \{A\}c\{B\}$.

Proof. The proof of this theorem is outside the scope of the course "Semantiek" (2IT40); it is put in Appendix A, which is available from the webpage [1]. □

So, according to the soundness theorem above, the validity of a formula $\{A\}c\{B\}$ can be formally demonstrated by exhibiting a derivation of it within TC. Recall from Example 2.11 that the invalidity of a formula $\{A\}c\{B\}$ can also be formally demonstrated, by exhibiting a counterexample. We have two types of counterexamples:

1. A counterexample showing the invalidity of a formula $\{A\}c\{B\}$ in the sense of *partial* correctness is a successful computation of c that starts in a state satisfying A and ends in a state not satisfying B.

2. A counterexample showing the invalidity of a formula $\{A\}c\{B\}$ in the sense of *total* correctness is a computation that starts in a state satisfying A and either fails, or diverges, or ends in a state that does not satisfy B.

Exercise 2.3 Which of the following formulas are valid in the sense of total correctness? If the formula is valid in the sense of total correctness, then provide a derivation in TC; otherwise, give a counterexample using the operational semantics. Moreover, for formulas that are invalid in the sense of total correctness, argue (using the operational semantics) whether they are valid or invalid in the sense of partial correctness.

(a) $\{\mathbf{true}\}\mathbf{if}\ X > 0 \rightarrow X := 0 \ [\!] \ X < 0 \rightarrow X := 0\ \mathbf{fi}\{X = 0\}$;

(b) $\{\mathbf{true}\}\mathbf{if}\ X > 0 \rightarrow X := 1 \ [\!] \ X < 0 \rightarrow X := 1\ \mathbf{fi}\{X = 1\}$;

(c)
```
{true}
if X > 0 → X := 0
[] X = 0 → skip
[] X < 0 → X := 0
fi
{X = 0};
```

(d)
```
{true}
if X > 0 → X := 1
[] X = 0 → skip
[] X < 0 → X := 1
fi
{X = 1};
```

(e) $\{\mathbf{true}\}\mathbf{if}\ X > 0\ \mathbf{then}\ X := 0\ \mathbf{else}\ X := 0\{X = 0\}$;

(f) $\{\mathbf{true}\}\mathbf{if}\ X > 0\ \mathbf{then}\ X := 1\ \mathbf{else}\ X := 1\{X = 1\}$.

Exercise 2.4 Do parts (a)–(d) of the previous exercise with repetitions instead of selections.

References

[1] http://www.win.tue.nl/~luttik/Courses/SEM, 2006.

[2] Anne Kaldewaij. *Programming: the derivation of algorithms.* Prentice-Hall, 1990.

[3] Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction.* Foundations of Computing. MIT Press, 1993.

A Proof of Theorem 2.12

The material in this appendix is out of the scope of the course "Semantiek" (2IT40) and need not be studied.

It is our goal to establish the soundness of the proof system TC with respect to validity in the sense of total correctness. That is, we want to establish for all **GCL** commands c and for all assertions A and B:

if $\vdash_{\mathsf{TC}} \{A\}c\{B\}$, then $\models_{tot} \{A\}c\{B\}$.

It suffices to prove that the rules in Table 2 are closed under $\models_{tot}$, for then the soundness of TC follows by the principle of rule induction. Observe that a sufficient condition for the validity of the formula $\{A\}c\{B\}$ in the sense of total correctness is that it is valid in the sense of partial correctness and satisfies the *termination requirement* that c cannot fail or diverge from any state that satisfies the precondition A.

Proposition A.1 For all **GCL** commands c and for all assertions A and B:

$$\models_{tot} \{A\}c\{B\} \text{ iff } \left[\begin{array}{l} \models \{A\}c\{B\}\ \& \\ \forall \sigma \in \Sigma,\ I : \mathbf{Intvar} \to \mathbf{N}.\ \sigma \models^I A \Rightarrow \bot \notin \mathcal{O}_{tot}[\![c]\!]\sigma \end{array}\right] .$$

To show that every theorem $\{A\}c\{B\}$ of TC is valid in the sense of *total* correctness, we shall now first establish that every theorem of TC is valid in the sense of *partial* correctness, and then argue that it also satisfies the *termination requirement.*

To establish that the rules of the proof system TC preserve validity in the sense of partial correctness we closely follow Winskel's soundness proof for the Hoare rules for correctness in **IMP** [3]. We begin with adapting his definition of denotations for **IMP** commands (on p. 60 of [3]) to **GCL** commands, replacing the clauses for the conditionals and while-loops by the following clauses for the selection and repetition constructs of **GCL**:

$$\begin{aligned} &\mathcal{C}[\![\mathbf{if}\ b_0 \to c_0 \ [\!]\ \cdots\ [\!]\ b_n \to c_n\ \mathbf{fi}]\!] \\ &\qquad = \{(\sigma,\sigma') \mid \exists i \in \{1,\ldots,n\}.\ \mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}\ \&\ (\sigma,\sigma') \in \mathcal{C}[\![c_i]\!]\}\ \ ; \text{ and} \end{aligned}$$

$$\begin{aligned} &\mathcal{C}[\![\mathbf{do}\ b_0 \to c_0 \ [\!]\ \cdots\ [\!]\ b_n \to c_n\ \mathbf{od}]\!] = \mathit{fix}(\Gamma)\ , \\ &\qquad \text{where } \Gamma(\varphi) = \{(\sigma,\sigma') \mid \exists i \in \{1,\ldots,n\}.\ \mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}\ \&\ (\sigma,\sigma') \in (\varphi \circ \mathcal{C}[\![c_i]\!])\} \\ &\qquad\qquad\qquad \cup \{(\sigma,\sigma) \mid \forall i \in \{1,\ldots,n\}.\ \mathcal{B}[\![b_i]\!]\sigma = \mathbf{false}\}\ . \end{aligned}$$

With these definitions we get the following correspondence between denotations of **GCL** commands and their operational semantics.

Theorem A.2 For all commands c

$$\mathcal{C}[\![c]\!] = \{(\sigma,\sigma') \mid \langle c,\sigma\rangle \to_1^* \sigma'\}\ .$$

Proof. The correspondence can be established by first showing a correspondence between the small-step operational semantics and a big-step operational semantics for **GCL**. Then, the remainder of the proof is a fairly straightforward adaptation of the proof of Theorem 5.7 in [3], respectively changing the arguments about the conditionals and while-loops of **IMP** by arguments about the selection and repetition constructs of **GCL**. The details are not hard and left to the reader. □

Now Winskel's soundness proof can be adapted to the setting of **GCL** so that it proves the following lemma.

Lemma A.3 If $\vdash_{\mathsf{TC}} \{A\}c\{B\}$, then $\models \{A\}c\{B\}$.

Proof. It suffices to show that each rule preserves validity in the sense of partial correctness; then the theorem follows by rule induction. Note that the rules ($\mathsf{TC1}$)–($\mathsf{TC3}$) and ($\mathsf{TC6}$) are, respectively, identical to the rule for **skip**, the rule for assignments, the rule for sequencing and the rule for consequence on p. 89 in [3], and in the proof of Theorem 6.11

in [3] it was already established that these rules are sound with respect to validity in the sense of partial correctness. So we only need to consider the rules (TC4) and (TC5). For the remainder of the proof, let $gc \equiv b_0 \to c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \to c_n$.

(TC4): Assume that the premises of (TC4) are valid in the sense of partial correctness; in particular, assume that $\models \{A \wedge b_i\}c_i\{B\}$ for all $i \in \{0, \dots, n\}$. We need to prove that $\models \{A\}\mathbf{if}\ gc\ \mathbf{fi}\{B\}$. That is, we need to prove for an arbitrary interpretation I and for an arbitrary state σ that

$$\sigma \models^I A \text{ implies } \sigma' \models^I B \text{ for all } \sigma' \in \mathcal{O}[\![\mathbf{if}\ gc\ \mathbf{fi}]\!]\sigma \ . \tag{1}$$

Suppose that $\sigma \models^I A$, and let $\sigma' \in \mathcal{O}[\![\mathbf{if}\ gc\ \mathbf{fi}]\!]\sigma$. Then $\langle \mathbf{if}\ gc\ \mathbf{fi}, \sigma\rangle \to_1^* \sigma'$, and hence, by Theorem A.2, $(\sigma, \sigma') \in \mathcal{C}[\![\mathbf{if}\ gc\ \mathbf{fi}]\!]$. So, according to the definition of $\mathcal{C}[\![\_]\!]$, $\mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}$ and $(\sigma, \sigma') \in \mathcal{C}[\![c_i]\!]$ for some $i \in \{0, \dots, n\}$. From $\mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}$ it follows that $\sigma \models^I b_i$ by Proposition 6.4 in [3], so $\sigma \models^I A \wedge b_i$. From $(\sigma, \sigma') \in \mathcal{C}[\![c_i]\!]$ and $\models \{A \wedge b_i\}c_i\{B\}$ it follows that $\sigma' \models^I B$. So, we have now established the implication (1), and thereby $\models \{A\}\mathbf{if}\ gc\ \mathbf{fi}\{B\}$ as was to be shown.

(TC5): Assume that the premises of (TC5) are valid in the sense of partial correctness; in particular assume that $\models \{A \wedge b_i\}c_i\{A\}$ for all $i \in \{0, \dots, n\}$. Let $b \equiv b_0 \vee \cdots \vee b_n$; we need to prove that $\models \{A\}\mathbf{do}\ gc\ \mathbf{od}\{A \wedge \neg b\}$. That is, we need to prove that for an arbitrary interpretation I and for an arbitrary state σ that

$$\sigma \models^I A \text{ implies } \sigma' \models^I A \wedge \neg b \text{ for all } \sigma' \in \mathcal{O}[\![\mathbf{do}\ gc\ \mathbf{od}]\!]\sigma \ . \tag{2}$$

Suppose that $\sigma \models^I A$, and let $\sigma' \in \mathcal{O}[\![\mathbf{do}\ gc\ \mathbf{od}]\!]\sigma$. Then $\langle \mathbf{do}\ gc\ \mathbf{od}, \sigma\rangle \to_1^* \sigma'$, and hence, by Theorem A.2, $(\sigma, \sigma') \in \mathcal{C}[\![\mathbf{do}\ gc\ \mathbf{od}]\!]$. Recall that

$$\mathcal{C}[\![\mathbf{do}\ gc\ \mathbf{od}]\!] = \bigcup_{n \in \omega} \Gamma^n(\emptyset)$$

where

$$\Gamma^0(\emptyset) = \emptyset, \text{ and}$$
$$\begin{aligned}\Gamma^{n+1}(\emptyset) = \{(\sigma, \sigma') \mid \exists i \in \{1, \dots, n\}.\ \mathcal{B}[\![b_i]\!]\sigma = \mathbf{true} \ \&\ (\sigma, \sigma') \in (\Gamma^n(\emptyset) \circ \mathcal{C}[\![c_i]\!])\}\\ \cup \{(\sigma, \sigma) \mid \forall i \in \{1, \dots, n\}.\ \mathcal{B}[\![b_i]\!]\sigma = \mathbf{false}\} \ .\end{aligned}$$

Note that to establish (2) it remains to prove that

$$(\sigma, \sigma') \in \Gamma^n(\emptyset) \ \&\ \sigma \models^I A \text{ implies } \sigma' \models^I A \wedge \neg b \tag{3}$$

for all $n \in \omega$; we proceed by induction on n.

If $n = 0$, then $\Gamma^n(\emptyset) = \emptyset$, so the implication (3) is vacuously true.

For the inductive step, assume that the implication (3) holds for n (induction hypothesis) and suppose that $(\sigma, \sigma') \in \Gamma^n(\emptyset)$ and $\sigma \models^I A$; there are two cases:

1. If $\mathcal{B}[\![b_i]\!]\sigma = \mathbf{false}$ for all $i \in \{0, \dots, n\}$, then $\sigma' = \sigma$, so $\sigma' \models^I A$. Furthermore, $\mathcal{B}[\![b]\!]\sigma = \mathbf{false}$, so $\sigma \models^I \neg b$ by Proposition 6.4 in [3]. It follows that $\sigma' \models^I A \wedge \neg b$.

2. Suppose that $\mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}$ and $(\sigma, \sigma') \in \Gamma^n(\emptyset) \circ \mathcal{C}[\![c_i]\!]$ for some $i \in \{0, \dots, n\}$. Then from $\mathcal{B}[\![b_i]\!]\sigma = \mathbf{true}$ it follows that $\sigma \models^I b_i$ by Proposition 6.4 in [3], so $\sigma \models^I A \wedge b_i$. Furthermore, since $(\sigma, \sigma') \in \Gamma^n(\emptyset) \circ \mathcal{C}[\![c_i]\!]$, there exists a state σ'' such that $(\sigma, \sigma'') \in \mathcal{C}[\![c_i]\!]$ and $(\sigma'', \sigma') \in \Gamma^n(\emptyset)$. Hence, by the assumption that $\models \{A \wedge b_i\}c_i\{A\}$ it follows that $\sigma'' \models^I A$, and then, by the induction hypothesis, $\sigma' \models^I A \wedge \neg b$.

In both cases we have established implication (3), and thereby $\models \{A\}\mathbf{do}\ gc\ \mathbf{od}\{A \wedge \neg b\}$. $\square$

It remains to prove that every theorem of TC satisfies the termination requirement. The idea is again to use rule induction, showing that the rules of TC preserve the termination requirement. For ($\mathsf{TC1}$) and ($\mathsf{TC2}$) this is clear: the commands **skip** and $X := a$ cannot fail or diverge from any state. That ($\mathsf{TC6}$) preserves the termination requirement is also clear, for if c cannot fail or diverge from any state that satisfies A' and $\models A \Rightarrow A'$, then clearly c cannot fail or diverge from any state that satisfies A.

To see that ($\mathsf{TC3}$) preserves the termination requirement, suppose that $\{A\}c_0\{C\}$ and $\{C\}c_1\{B\}$ are theorems of TC that both satisfy the termination requirement. Then c_0 cannot fail or diverge from any state σ that satisfies A, and c_1 cannot fail or diverge from state σ' that satisfies C. Moreover, by Lemma A.3, $\models \{A\}c_0\{C\}$, so if $\sigma \models^I A$ and $\langle c, \sigma\rangle \rightarrow_1^* \sigma'$, then $\sigma \models^I C$. To show that $\{A\}c_0;c_1\{B\}$ satisfies the termination requirement we first establish that $\langle c_0\,;c_1,\sigma\rangle \rightarrow_1^* \sigma'$ iff $\exists\sigma''.\ \langle c_0,\sigma\rangle \rightarrow_1^* \sigma''$ & $\langle c_1,\sigma''\rangle \rightarrow_1^* \sigma'$ (recall that $\rightarrow_1^*$ denotes the transitive, reflexive closure of $\rightarrow_1$ (cf. p. 10 of [3]), as a corollary to the following lemma.

Lemma A.4 For all commands c_0, c_1, for all states σ, σ', and for all $m \geq 0$:

$$\langle c_0\,;c_1,\sigma\rangle \rightarrow_1^m \sigma' \text{ iff } \exists\sigma'',k,l.\ \langle c_0,\sigma\rangle \rightarrow_1^k \sigma'' \ \&\ \langle c_1,\sigma''\rangle \rightarrow_1^l \sigma' \ \&\ m = k + l\ .$$

Proof. By induction. □

The following corollary is a straightforward consequence.

Corollary A.5 For all commands c_0, c_1 and for all states σ, σ':

$$\langle c_0\,;c_1,\sigma\rangle \rightarrow_1^* \sigma' \text{ iff } \exists\sigma''.\ \langle c_0,\sigma\rangle \rightarrow_1^* \sigma'' \ \&\ \langle c_1,\sigma''\rangle \rightarrow_1^* \sigma'\ .$$

That $\{A\}c_0\,;c_1\{B\}$ satisfies the termination requirement now follows from the following lemma.

Lemma A.6 Let c_0 and c_1 be commands, and let σ be a state. If c_0 cannot fail or diverge from σ and c_1 cannot fail or diverge from any of the states σ' such that $\langle c_0,\sigma\rangle \rightarrow_1^* \sigma'$, then $c_0\,;c_1$ cannot fail or diverge from σ.

Proof. By Corollary A.5 it holds for all **GCL** commands c_0, c_1, c' and for all states σ, σ' that:

$$\begin{aligned}\langle c_0\,;c_1,\sigma\rangle \rightarrow_1^* \langle c',\sigma'\rangle \text{ iff } \exists c_0'.\ \langle c_0,\sigma\rangle \rightarrow_1^* \langle c_0',\sigma'\rangle \ \&\ c' \equiv c_0'\,;c_1 \text{ or}\\ \exists\sigma''.\ \langle c_0,\sigma\rangle \rightarrow_1^* \sigma'' \ \&\ \langle c_1,\sigma''\rangle \rightarrow_1^* \langle c',\sigma'\rangle\ .\end{aligned} \tag{4}$$

To establish the lemma, we now use (4) to show that if $c_0\,;c_1$ can fail or diverge from σ, then either c_0 can fail or diverge from σ, or c_1 can fail or diverge from some state σ'' such that $\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''$.

First, suppose that $c_0\,;c_1$ can fail from σ, *i.e.* suppose that $\langle c_0\,;c_1,\sigma\rangle \rightarrow_1^* \langle c',\sigma'\rangle$ and $\langle c',\sigma'\rangle$ is blocking. Then by (4) we can distinguish two cases. If $c' \equiv c_0'\,;c_1$ with $\langle c_0,\sigma\rangle \rightarrow_1^* \langle c_0',\sigma'\rangle$, then it is clear that $\langle c_0',\sigma'\rangle$ is blocking, so c_0 can fail from σ. Otherwise, there exists a state σ'' such that $\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''$ and $\langle c_1,\sigma''\rangle \rightarrow_1^* \langle c',\sigma'\rangle$, *i.e.* c_1 can fail from a state σ'' such that $\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''$.

Next, suppose that $c_0\,;c_1$ can diverge from σ, *i.e.* suppose that there is an infinite computation of $c_0\,;c_1$ starting at σ. Again, according to (4) we can distinguish two cases. If in this infinite computation there exists a configuration $\langle c',\sigma'\rangle$ such that $\langle c_0,\sigma\rangle \rightarrow_1^* \sigma''$ and $\langle c_1.\sigma''\rangle \rightarrow_1^* \langle c',\sigma'\rangle$ for some state σ'', then clearly c_1 can diverge from σ''. On the other hand, if no such configuration exists, then the computation is of the form

$$\langle c_0\,;c_1,\sigma\rangle = \langle c_0'\,;c_1,\sigma_0\rangle \rightarrow_1 \langle c_1'\,;c_1,\sigma_1\rangle \rightarrow_1 \cdots \rightarrow_1 \langle c_k'\,;c_1,\sigma_k\rangle \rightarrow_1 \cdots \quad (k \geq 0)\ .$$

Clearly, by omitting the component ';c_1' from every configuration in the above computation an infinite computation of c_0 starting in σ is obtained, so c_0 can diverge from σ. □

To see that (TC4) preserves the termination requirement, suppose that $\{A \wedge b_i\}c_i\{B\}$ ($i \in \{0, \ldots, n\}$) are theorems of TC that satisfy the termination requirement, and suppose that $\models A \Rightarrow (b_0 \vee \cdots \vee b_n)$, Consider a state σ that satisfies A. Then from $\models A \Rightarrow (b_0 \vee \cdots \vee b_n)$ it follows that $\sigma \models^I b_i$ for some $i \in \{0, \ldots, n\}$. Moreover, since $\{A \wedge b_i\}c_i\{B\}$ satisfies the termination requirement, it follows that, for all $i \in \{0, \ldots, n\}$ such that $\sigma \models^I b_i$, c_i cannot fail or diverge from σ. Hence, that $\{A\}\mathbf{if}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{fi}\{B\}$ satisfies the termination requirement now follows from the following lemma.

Lemma A.7 Let $b_0, \ldots, b_n$ be boolean expressions, let $c_0, \ldots, c_n$ be commands, and let σ be a state. If $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$ for some $i \in \{0, \ldots, n\}$, and for all $i \in \{0, \ldots, n\}$ such that $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$ it holds that c_i cannot fail or diverge from σ, then $\mathbf{if}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{fi}$ cannot fail or diverge from σ.

Proof. If $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$ for some $i \in \{0, \ldots, n\}$, then $\langle \mathbf{if}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{fi}, \sigma\rangle$ is not blocking. From the rules in Table 3, every computation of $\langle \mathbf{if}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{fi}, \sigma\rangle$ begins with a transition

$$\langle \mathbf{if}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{fi}, \sigma\rangle \longrightarrow_1 \langle c_i, \sigma\rangle$$

for some $i \in \{0, \ldots, n\}$ with $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$. Hence, since c_i cannot fail or diverge from σ, it follows that $\mathbf{if}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{fi}$ cannot fail or diverge from σ either. □

It remains to argue that (TC5) preserves the termination requirement. First we characterise when a repetition can fail or diverge.

Lemma A.8 Let $b_0, \ldots, b_n$ be boolean expressions, let $c_0, \ldots, c_n$ be commands, and σ be a state. If $\mathbf{do}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{od}$ can fail or diverge from σ, then there exists $i \in \{0, \ldots, n\}$ such that $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$ and

(i) c_i can fail or diverge from σ, or

(ii) there exists a state σ' such that $\langle c_i, \sigma\rangle \longrightarrow_1^* \sigma'$ and $\mathbf{do}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{od}$ can fail or diverge from σ'.

Proof. Note that a computation of $r \equiv \mathbf{do}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{od}$ starting in a state σ can be of two shapes: either it consists of a single transition

$$\langle r, \sigma\rangle \longrightarrow_1 \sigma\ \ ,$$

or it begins with a transition

$$\langle r, \sigma\rangle \longrightarrow_1 \langle c_i\ ;\ r, \sigma\rangle$$

for some $i \in \{0, \ldots, n\}$ such that $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$. It is clear that if r can fail or diverge, then there is a computation of the second shape such that $c_i\ ;\ r$ can fail or diverge from σ. According to Lemma A.6 this means that either c_i can fail or diverge from σ, or there exists a state σ' such that $\langle c_i, \sigma\rangle \longrightarrow_1^* \sigma'$ and r can fail or diverge from σ. □

Now consider the rule (TC5). We want to show that the premises ensure that the repetition $r \equiv \mathbf{do}\ b_0 \rightarrow c_0 \mathbin{[\!]} \cdots \mathbin{[\!]} b_n \rightarrow c_n\ \mathbf{od}$ cannot fail or diverge from any state σ that satisfies A. If r *can* fail or diverge from a state σ, then by the preceding lemma we are in one of two cases:

1. There exists a finite sequence of states $\sigma_0, \ldots, \sigma_m, \sigma_{m+1}$ such that $\sigma = \sigma_0$,

$$\langle r, \sigma_0\rangle \longrightarrow_1^* \langle r, \sigma_1\rangle \longrightarrow_1^* \cdots \longrightarrow_1^* \langle r, \sigma_m\rangle \longrightarrow_1 \langle c_i\ ;\ r, \sigma_{m+1}\rangle\ \ ,$$

   for all $k \in \{0, \ldots, m-1\}$ there exists $l \in \{0, \ldots, n\}$ such that $\langle c_l, \sigma_k\rangle \longrightarrow_1^* \sigma_{k+1}$, and c_i can fail or diverge from σ_{m+1}. To see that the premises of (TC5) exclude this case, note that if σ would satisfy A, then the premise $\{A \wedge b_i\}c_i\{A\}$ would ensure that every σ_k ($0 \leq k \leq m+1$) satisfies A (*i.e.*, A is an invariant of the repetition), and hence, in particular, that σ_{m+1} satisfies A. But then the termination requirement for the premise $\{A \wedge b_i\}c_i\{A\}$ implies that c_i cannot fail or diverge from σ_{m+1}, and thereby this case is excluded.

2. There exists an infinite sequence of states $\sigma_0, \ldots, \sigma_k, \ldots$ such that $\sigma = \sigma_0$,

$$\langle r, \sigma_0 \rangle \to_1^* \cdots \to_1^* \langle r, \sigma_k \rangle \to_1^* \cdots$$

and for all $k \geq 0$ there exists $l \in \{0, \ldots, n\}$ such that $\langle c_l, \sigma_k \rangle \to_1^* \sigma_{k+1}$. To see that the premises of (TC5) exclude this case, note that if σ would satisfy A, then the premise $\{A \wedge b_i\}c_i\{A\}$ would ensure that every σ_k ($k \geq 0$) satisfies A (*i.e.*, A is an invariant of the repetition). Moreover, the premise $\{A \wedge b_i \wedge t = z\}c_i\{\neg(z \leq t)\}$ would ensure that for all $k \geq 0$, the value of t in σ_{k+1} is strictly less than the value of t in σ_k, so with the infinite sequence of states $\sigma_0, \ldots, \sigma_k, \ldots$ corresponds an infinite decreasing sequence of values of t. However, since the premise $\models (A \Rightarrow 0 \leq t)$ ensures that in every state σ_k the value of t is greater or equal 0, such an infinite decreasing sequence cannot exist, and thereby also this case is excluded.

We have now argued that every rule of TC preserves the termination requirement. By the principle of rule induction, the next lemma is an immediate consequence.

Lemma A.9 If $\vdash_{\mathsf{TC}} \{A\}c\{B\}$, then for all states σ and for all interpretations I:

$$\sigma \models^I A \Rightarrow \bot \notin \mathcal{O}_{tot}[\![c]\!]\sigma \ .$$

Proof. We show that the rules of TC preserve the predicate T, for A, c and B defined by

$$T(A, c, B) \Longleftrightarrow \Big[\forall \sigma \in \Sigma,\ I : \mathbf{Intvar} \to \mathbf{N}.\ \sigma \models^I A \Rightarrow \bot \notin \mathcal{O}_{tot}[\![c]\!]\sigma\Big] \ .$$

Then the lemma follows by the principle of rule induction.

(TC1): Let σ be any state. From the rules in Table 3 it is clear that $\langle \mathbf{skip}, \sigma \rangle \to_1 \sigma$ is the only computation of **skip** starting in σ, so **skip** cannot fail or diverge from σ. Hence, $T(A, \mathbf{skip}, A)$.

(TC2): Let σ be any state. From the rules in Table 3 it is clear that $\langle X := a, \sigma \rangle \to_1 \sigma[n/X]$ with $\langle a, \sigma \rangle \to n$ is the only computation of $X := a$ starting in σ, so it is clear that $X := a$ cannot fail or diverge from σ. Hence, $T(A, X := a, A)$.

(TC3): Assume $\vdash_{\mathsf{TC}} \{A\}c_0\{C\}$ and $T(A, c_0, C)$ and $\vdash_{\mathsf{TC}} \{C\}c_1\{B\}$ and $T(C, c_1, B)$. First observe that from these assumptions it follows by Lemma A.3 and Proposition A.1 that $\models_{tot} \{A\}c_0\{C\}$ and $\models_{tot} \{C\}c_1\{B\}$. Now, to prove that $T(A, c_0\ ;\ c_1, B)$ we need to establish for all interpretations I and for all states σ that

$$\sigma \models^I A \Rightarrow \bot \notin \mathcal{O}_{tot}[\![c_0\ ;\ c_1]\!]\sigma \ .$$

Suppose $\sigma \models^I A$. From $\models_{tot} \{A\}c_0\{C\}$ it follows that

$$\bot \notin \mathcal{O}_{tot}[\![c_0]\!]\sigma \ \&\ \forall \sigma' \in \mathcal{O}[\![c_0]\!]\sigma.\ \sigma' \models^I C \ ,$$

and from $\models_{tot} \{C\}c_1\{B\}$ it follows that

$$\bot \notin \mathcal{O}_{tot}[\![c_1]\!]\sigma'' \ \&\ \forall \sigma'' \in \mathcal{O}[\![c_1]\!]\sigma'.\ \sigma'' \models^I B \text{ for all states } \sigma' \in \mathcal{O}[\![c_0]\!]\sigma \ .$$

So, c_0 cannot fail or diverge from σ and c_1 cannot fail or diverge from σ' for all states σ' such that $\langle c_0, \sigma \rangle \to_1^* \sigma'$; hence $c_0\,;c_1$ cannot fail or diverge from σ by Lemma A.6, and thus $\bot \notin \mathcal{O}_{tot}[\![c_0\ ;\ c_1]\!]\sigma$. Thereby we have established that $T(A, c_0\ ;\ c_1, B)$.

(TC4): Assume that $\models (A \Rightarrow (b_0 \vee \cdots \vee b_n))$ and that $\vdash_{\mathsf{TC}} \{A \wedge b_i\}c_i\{B\}$ and $T(A \wedge b_i, c_i, B)$ for all $i \in \{1, \ldots, n\}$. First, observe that from these assumptions it follows by Lemma A.3 and Proposition A.1 that $\models_{tot} \{A \wedge b_i\}c_i\{B\}$ for all $i \in \{1, \ldots, n\}$. Now, to prove that $T(A, \mathbf{if}\ b_0 \to c_0\ [\!]\ \cdots\ [\!]\ b_n \to c_n\ \mathbf{fi}, B)$, we need to establish for all interpretations I and for all states σ that

$$\sigma \models^I A \Rightarrow \bot \notin \mathcal{O}_{tot}[\![\mathbf{if}\ b_0 \to c_0\ [\!]\ \cdots\ [\!]\ b_n \to c_n\ \mathbf{fi}]\!]\sigma \ .$$

Suppose that $\sigma \models^I A$. Then, for all $i \in \{0, \ldots, n\}$, from $\models_{tot} \{A \wedge b_i\}c_i\{B\}$, it follows that $\sigma \models^I b_i$ implies

$$\bot \notin \mathcal{O}_{tot}[\![c_i]\!]\sigma \ \&\ \forall \sigma' \in \mathcal{O}[\![c_i]\!].\ \sigma' \models^I B \ .$$

Note that from $\models (A \Rightarrow (b_0 \vee \cdots \vee b_n))$ it follows, using Exercise 3.5, Lemma 5.4 and Proposition 6.4 in [3], that $\langle b_i, \sigma\rangle \not\rightarrow \mathbf{false}$ for some $i \in \{0, \ldots, n\}$. Similarly it follows for all $i \in \{0, \ldots, n\}$ that $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$ implies $\sigma \models^I b_i$, and hence that c_i cannot fail or diverge from σ. So by Lemma A.7 it follows that that **if** $b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n$ **fi** cannot fail or diverge from σ, and thus

$$\perp \notin \mathcal{O}_{tot}[\![\mathbf{if}\ b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n\ \mathbf{fi}]\!]\sigma \ .$$

Hence $T(A, \mathbf{if}\ b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n\ \mathbf{fi}, B)$.

(TC5): Assume that t is an integer expression, that z is an integer variable that does not occur in t and A such that the premises of (TC5) hold. In particular, assume that $\models (A \Rightarrow 0 \leq t)$, $\vdash_{\mathsf{TC}} \{A \wedge b_i \wedge t = z\} c_i \{\neg(z \leq t)\}$ and $T(A \wedge b_i \wedge t = z, c_i, \neg(z \leq t))$ for all $i \in \{0, \ldots, n\}$. First observe that from these assumptions it follows by Lemma A.3 and Proposition A.1 that $\models_{tot} \{A \wedge b_i \wedge t = z\} c_i \{\neg(z \leq t)\}$. Now, to prove that $T(A, \mathbf{do}\ b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n\ \mathbf{od}, A \wedge \neg(b_0 \vee \cdots \vee b_n))$, we need to establish for all interpretations I and for all states σ that

$$\sigma \models^I A \Rightarrow \perp \notin \mathcal{O}_{tot}[\![\mathbf{do}\ b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n\ \mathbf{od}]\!]\sigma \ . \tag{5}$$

We shall prove that the implication holds for all interpretations I and for all states σ by deriving a contradiction from the contrary assumption. So, assume that there exist pairs of interpretations and states for which the implication (5) does *not* hold. By the assumptions that $\models (A \Rightarrow 0 \leq t)$ and that z does not occur in A or t, we can then choose among them a pair (I, σ) such that $\mathcal{A}v[\![t]\!]I\sigma$ is minimal and $I(z) = \mathcal{A}v[\![t]\!]I\sigma$. Since $\perp \notin \mathcal{O}_{tot}[\![\mathbf{do}\ b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n\ \mathbf{od}]\!]\sigma$, *i.e.* **do** $b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n$ **od** can fail or diverge from σ, there exists according to Lemma A.8 an $i \in \{0, \ldots, n\}$ such that $\langle b_i, \sigma\rangle \rightarrow \mathbf{true}$ and

(i) c_i can fail or diverge from σ, or

(ii) there exists a state σ' such that $\langle c_i, \sigma\rangle \rightarrow_1^* \sigma'$ and **do** $b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n$ **od** can fail or diverge from σ'.

Since $\sigma \models^I A \wedge b_i \wedge t = z$, it follows from the assumption $\models_{tot} \{A \wedge b_i \wedge t = z\} c_i \{\neg(z \leq t)\}$ that c_i cannot fail or diverge from σ, so the first case (i) does not apply. So, it remains to derive a contradiction for the second case (ii); let σ' be such that $\langle c_i, \sigma\rangle \rightarrow_1^* \sigma'$ and **do** $b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n$ **od** can fail or diverge from σ'. Note that the latter means that the pair $(\sigma', I[\mathcal{A}v[\![t]\!]I\sigma'/z])$ satisfies (5). However, from the assumption that $\models_{tot} \{A \wedge b_i \wedge t = z\} c_i \{\neg(z \leq t)\}$ it follows that $\sigma' \models^I \neg(z \leq t)$, and hence $\mathcal{A}v[\![t]\!]I\sigma' < \mathcal{A}v[\![t]\!]I\sigma$, which contradicts the minimality of σ. We have now derived a contradiction from the assumption that there exist an interpretation I and a state σ that do not satisfy the implication (5). It follows that the implication (5) holds for all interpretations I and for all states σ. Hence, $T(A, \mathbf{do}\ b_0 \rightarrow c_0 \; [] \cdots [] \; b_n \rightarrow c_n\ \mathbf{od}, A \wedge \neg(b_0 \vee \cdots \vee b_n))$.

(TC6): Assume $\models (A \Rightarrow A')$, $\vdash_{\mathsf{TC}} \{A'\} c \{B'\}$ and $T(A', c, B')$, and $\models (B' \Rightarrow B)$. To prove that $T(A, c, B)$ we need to establish for all interpretations I and for all states σ that

$$\sigma \models^I A \Rightarrow \perp \notin \mathcal{O}_{tot}[\![c]\!]\sigma \ .$$

Suppose that $\sigma \models^I A$. Then, from $\models (A \Rightarrow A')$, it follows that $\sigma \models^I A'$. Since $T(A', c, B')$, it follows that c cannot fail or diverge from σ, so $\perp \notin \mathcal{O}_{tot}[\![c]\!]\sigma$. Hence $T(A, c, B)$. □

We are now in a position to prove that the system TC is sound for validity in the sense of total correctness.

Theorem 2.12 If $\vdash_{\mathsf{TC}} \{A\} c \{B\}$, then $\models \{A\} c \{B\}$.

Proof. Suppose that $\{A\} c \{B\}$ is a theorem of TC. Then by Lemma A.3 it is valid in the sense of partial correctness, and by Lemma A.9 it satisfies the termination requirement. Hence, by Proposition A.1 it is valid in the sense of total correctness. □