A HOARE LOGIC FOR THE COINDUCTIVE TRACE-BASED BIG-STEP SEMANTICS OF WHILE

KEIKO NAKATA AND TARMO UUSTALU

Institute of Cybernetics at Tallinn University of Technology, Akadeemia tee 21, EE-12618 Tallinn, Estonia
e-mail address: {keiko,tarmo}@cs.ioc.ee

Abstract. In search for a foundational framework for reasoning about observable behavior of programs that may not terminate, we have previously devised a trace-based big-step semantics for While. In this semantics, both traces and evaluation (relating initial states of program runs to traces they produce) are defined coinductively. On terminating runs, it agrees with the standard inductive state-based semantics. Here we present a Hoare logic counterpart of our coinductive trace-based semantics and prove it sound and complete. Our logic subsumes the standard partial-correctness state-based Hoare logic as well as the total-correctness variation: they are embeddable. In the converse direction, projections can be constructed. Since we work with a constructive underlying logic, the range of expressible program properties has a rich structure; in particular, we can distinguish between termination and nondivergence, e.g., unbounded total search fails to be terminating but is nonetheless nondivergent. Our metatheory is entirely constructive as well, and we have formalized it in Coq.

1. Introduction

Standard big-step semantics and (partial correctness) Hoare logics do not support reasoning about nonterminating runs of programs. Essentially, they ignore them. But of course nonterminating runs are important. Not only need we often program a partially recursive function whose domain of definedness we cannot decide or is undecidable, e.g., an interpreter, but we also have to program functions that are inherently partially recursive. In programming with interactive input/output, for example, diverging runs are often what we really want.

In search for a foundational framework for reasoning about possibly nonterminating programs and intrigued by attempts in this direction in the literature, we have previously devised a big-step semantics for While based on traces [14]. In this semantics, traces are possibly infinite sequences of states that a program run goes through. They are defined coinductively, as is the evaluation relation, relating initial states of program runs to traces they produce. On terminating runs, this nonstandard semantics agrees with the standard, inductive state-based big-step semantics.

In this paper, we put forward a Hoare logic to match this big-step semantics. In this new trace-based logic, program runs are reasoned about in terms of predicates on states and traces. More precisely, our Hoare triple $\{U\}\ s\ \{P\}$ is given by a statement s, a state

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

predicate U (a condition on the initial state of a run of s) and a trace predicate P (a condition on the trace produced by the run). The interesting question is the choice of the language of assertions, i.e., the language in which we want to express these predicates. We would like to identify a suite of connectives for the assertion language with whom we achieve a sound and complete Hoare logic for a constructive underlying logic. We adopt a solution that is reminiscent of interval temporal logic [13, 7] (with a chop-connective). The logic we propose is Spartan in terms of convenience of expression, but should well qualify as a foundational formalism into which more specialized and more applied logics can be translated.

The While language is total (as soon as we accept that traces of program runs can be infinite) and deterministic. This allows our logic to conservatively extend both the standard, state-based partial-correctness Hoare logic as well as the state-based total-correctness Hoare logic. On the level of derivability alone this can be proved semantically by going through the soundness and completeness results. But we go one step further: we show that derivations in these two state-based logics are directly transformable into derivations in our logic, yielding embeddings on the level of derivations, not just mere derivability. The transformations are relatively straightforward and do not require invention of new invariants or variants, demonstrating that our logic incurs no undue proof burden in comparison to the standard Hoare logics. In the converse direction, we can project derivations in our trace-based logic into derivations in the state-based logics.

However, the power of our logic goes beyond that of the state-based partial-correctness and total-correctness Hoare logics. The assertion language has access to traces. As suggested by its similarity to the assertion language of interval temporal logic, this allows us to specify liveness properties of diverging runs. We will demonstrate this extra expressiveness of our logic by a series of examples. Also, interpreted into a constructive underlying logic, our assertion language becomes quite discerning. In particular we can distinguish between termination and nondivergence, e.g., unbounded total[1] search fails to be terminating, but is nonetheless nondivergent.

We do not discuss this in the paper, but our logic can be adjusted to deal with exceptions and nondeterminism.

The paper is organized as follows. In Section 2, we present our trace-based big-step semantics. In Section 3, we proceed to the question of a corresponding Hoare logic. We explain our design considerations and then present our Hoare logic and the soundness and completeness proofs. In Section 4, we show the embeddings of the state-based partial-correctness and total-correctness Hoare logics into our logic and the projections back. In Section 5, we consider examples. In Section 6, we discuss the related work, to conclude in Section 7. We have formalized the development fully constructively in Coq version 8.1pl3 using the Ssreflect syntax extension library. The Coq development is available at `http://cs.ioc.ee/~keiko/abyss.tgz`.

2. Big-step semantics

We start with our big-step semantics. This is defined in terms of states and traces. The notion of a state is standard. A state $\sigma \in \mathit{state}$ is an assignment of integer values to the

[1] We should really say "nonpartial".

variables. Traces $\tau \in \mathit{trace}$ are defined coinductively by the rules[2]

$$\frac{}{\langle\sigma\rangle \in \mathit{trace}} \qquad \frac{\tau \in \mathit{trace}}{\sigma :: \tau \in \mathit{trace}}$$

so a trace is a non-empty colist (possibly infinite sequence) of states. We also define bisimilarity of two traces, $\tau \approx \tau'$, coinductively by

$$\frac{}{\langle\sigma\rangle \approx \langle\sigma\rangle} \qquad \frac{\tau \approx \tau'}{\sigma :: \tau \approx \sigma :: \tau'}$$

Bisimilarity is straightforwardly seen to be an equivalence. We think of bisimilar traces as equal, i.e., type-theoretically we treat traces as a setoid with bisimilarity as the equivalence relation[3]. Accordingly, we have to make sure that all functions and predicates we define on traces are setoid functions and predicates (i.e., insensitive to bisimilarity). We define the initial state $hd\ \tau$ of a trace τ by case distinction by $hd\ \langle\sigma\rangle = \sigma, hd\ (\sigma :: \tau) = \sigma$. The function hd is a setoid function. We also define finiteness of a trace (with a particular final state) and infiniteness of a trace inductively resp. coinductively by

$$\frac{}{\langle\sigma\rangle \downarrow \sigma} \qquad \frac{\tau \downarrow \sigma'}{\sigma :: \tau \downarrow \sigma'} \qquad \frac{\tau^\uparrow}{(\sigma :: \tau)^\uparrow}$$

Finiteness and infiniteness are setoid predicates. It should be noticed that infiniteness is defined positively, not as negation of finiteness. Constructively, it is not the case that $\forall\tau.\,(\exists\sigma.\,\tau \downarrow \sigma) \vee \tau^\uparrow$, which amounts to asserting that finiteness is decidable. In particular, $\forall\tau.\,(\neg\,\exists\sigma.\,\tau \downarrow \sigma) \to \tau^\uparrow$ is constructively provable, but $\forall\tau.\,(\neg\,\tau^\uparrow) \to \exists\sigma.\,\tau \downarrow \sigma$ is not.

Evaluation $(s, \sigma) \Rightarrow \tau$, expressing that running a statement s from a state σ produces a trace τ, is defined coinductively by the rules in Figure 1. The rules for sequence and while implement the necessary sequencing with the help of extended evaluation $(s, \tau) \stackrel{*}{\Rightarrow} \tau'$, also defined coinductively, as the coinductive prefix closure of evaluation: $(s, \tau) \stackrel{*}{\Rightarrow} \tau'$ expresses that running a statement s from the last state (if it exists) of an already accumulated trace τ results in a total trace τ'.

A remarkable feature of the definition of $(s, \tau) \stackrel{*}{\Rightarrow} \tau'$ is that it does not hinge on deciding whether the trace τ is finite or not, which is constructively impossible. A proof of $(s, \tau) \stackrel{*}{\Rightarrow} \tau'$ simply traverses the already accumulated trace τ: if the last element is hit, which is the case when τ is finite, then the statement is run, otherwise the traversal goes on forever.

We look closer at the sequence rule. We want to conclude that $(s_0; s_1, \sigma) \Rightarrow \tau'$ from the premise $(s_0, \sigma) \Rightarrow \tau$. Classically, either the run of s_0 terminates, i.e., $\tau \downarrow \sigma'$ for some σ', or it diverges, i.e., $\tau^\uparrow$. In the first case, we would like to additionally use that τ is a finite prefix of τ' and that $(s_1, \sigma') \Rightarrow \tau''$, where τ'' is the rest of τ'. In the second case, it should be case that $\tau \approx \tau'$. In both cases, the desirable condition is equivalent to $(s_1, \tau) \stackrel{*}{\Rightarrow} \tau'$, which is the second premise of our rule. The use of extended evaluation, defined as the coinductive (rather than inductive) prefix closure of evaluation, allows us to avoid the need to decide whether the run of s_0 terminates or not.

Evaluation is a setoid predicate. Moreover, for While, it is deterministic (up to bisimilarity, as is appropriate for our notion of trace equality).

[2]We mark coinductive definitions by double horizontal rules.

[3]Classically, strong bisimilarity is equality. But we work in an intensional type theory where strong bisimilarity of colists is weaker than equality (just as equality of two functions on all arguments is weaker than equality of these two functions).

$$\frac{}{(x := e, \sigma) \Rightarrow \sigma :: \langle \sigma[x \mapsto [\![e]\!]\sigma] \rangle} \quad \frac{}{(\mathsf{skip}, \sigma) \Rightarrow \langle \sigma \rangle} \quad \frac{(s_0, \sigma) \Rightarrow \tau \quad (s_1, \tau) \overset{*}{\Rightarrow} \tau'}{(s_0; s_1, \sigma) \Rightarrow \tau'}$$

$$\frac{\sigma \models e \quad (s_t, \sigma :: \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau}{(\mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f, \sigma) \Rightarrow \tau} \quad \frac{\sigma \not\models e \quad (s_f, \sigma :: \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau}{(\mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f, \sigma) \Rightarrow \tau}$$

$$\frac{\sigma \models e \quad (s_t, \sigma :: \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau \quad (\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau) \overset{*}{\Rightarrow} \tau'}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \Rightarrow \tau'} \quad \frac{\sigma \not\models e}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \Rightarrow \sigma :: \langle \sigma \rangle}$$

$$\frac{(s, \sigma) \Rightarrow \tau}{(s, \langle \sigma \rangle) \overset{*}{\Rightarrow} \tau} \quad \frac{(s, \tau) \overset{*}{\Rightarrow} \tau'}{(s, \sigma :: \tau) \overset{*}{\Rightarrow} \sigma :: \tau'}$$

Figure 1: Big-step semantics

Proposition 2.1. For any s, σ, τ and τ', if $(s, \sigma) \Rightarrow \tau$ and $(s, \sigma) \Rightarrow \tau'$, then $\tau \approx \tau'$.

In our definition, we have made a choice as regards to what grows the trace of a run. We have decided that assignments and testing of guards of if- and while-statements augment the trace by a state (but skip does not), e.g., we have $(x := 17, \sigma) \Rightarrow \sigma :: \langle \sigma[x \mapsto 17] \rangle$, $(\mathsf{while\ false\ do\ skip}, \sigma) \Rightarrow \sigma :: \langle \sigma \rangle$ and $(\mathsf{while\ true\ do\ skip}, \sigma) \Rightarrow \sigma :: \sigma :: \sigma :: \ldots$.

This is good for several reasons. First, skip becomes a unit of sequential composition, i.e., the semantics does not distinguish s, $\mathsf{skip}; s$ and $s; \mathsf{skip}$. Second, we get a notion of small steps that fully agrees with a very natural coinductive trace-based small-step semantics arising as a straightforward variation of the textbook inductive state-based small-step semantics. The third and most important outcome is that any while-loop always progresses, because testing of the guard is a small step. For instance, in our semantics while true do skip can only derive $(\mathsf{while\ true\ do\ skip}, \sigma) \Rightarrow \sigma :: \sigma :: \sigma :: \ldots$ (up to bisimilarity). As we discuss below, giving up insisting on progress in terms of growing the trace would introduce some semantic anomalies. But it also ensures that evaluation is total—as we should expect. Given that it also is deterministic, we can thus equivalently turn our relational big-step semantics into a functional one: the unique trace for a given statement and initial state is definable by corecursion. (For details, see our previous paper [14].)

The coinductive trace-based semantics agrees with the inductive state-based semantics.

Proposition 2.2. For any s, σ, σ', existence of τ such that $(s, \sigma) \Rightarrow \tau$ and $\tau{\downarrow}\sigma'$ is equivalent to $(s, \sigma) \Rightarrow^{\mathrm{ind}} \sigma'$.

We notice that the inductive state-based semantics is not total constructively—we cannot decide the halting problem.

Discussions on alternative designs. We look at several seemingly not so different but problematic alternatives that we reject, thereby revealing some subtleties in designing coinductive big-step semantics and motivating our design choices.

Since progress of loops is not required for wellformedness of the definitions of $\Rightarrow$ and $\overset{*}{\Rightarrow}$, one might be tempted to regards guard testing to be instantaneous and modify the rules

for the while-loop to take the form

$$\frac{\sigma \models e \quad (s_t, \sigma) \Rightarrow \tau \quad (\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau) \overset{*}{\Rightarrow} \tau'}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \Rightarrow \tau'} \qquad \frac{\sigma \not\models e}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \Rightarrow \langle\sigma\rangle}$$

This leads to undesirable outcomes. We can derive $(\mathsf{while\ true\ do\ skip}, \sigma) \Rightarrow \langle\sigma\rangle$, which means that the non-terminating while true do skip is considered semantically equivalent to the terminal (immediately terminating) skip. Worse, we can also derive $(\mathsf{while\ true\ do\ skip}; x := 17, \sigma) \Rightarrow \sigma :: \langle\sigma[x \mapsto 17]\rangle$, which is even more inadequate: a sequence can continue to run after the non-termination of the first statement. Yet worse, inspecting the rules closer we discover we are also able to derive $(\mathsf{while\ true\ do\ skip}, \sigma) \Rightarrow \tau$ for any τ. Mathematically, giving up insisting on progress in terms of growing the trace has also the consequence that the relational semantics cannot be turned into a functional one, although While should intuitively be total and deterministic. In a functional semantics, evaluation must be a trace-valued function and in a constructive setting such a function must be productive.

Another option, where assignments and test of guards are properly taken to constitute steps, could be to define $\overset{*}{\Rightarrow}$ by case distinction on the statement by rules such as

$$\frac{\tau \models^* e \quad (s_t, \mathit{duplast}\ \tau) \overset{*}{\Rightarrow} \tau' \quad (\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau') \overset{*}{\Rightarrow} \tau''}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau) \overset{*}{\Rightarrow} \tau''} \qquad \frac{\tau \not\models^* e}{(\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau) \overset{*}{\Rightarrow} \mathit{duplast}\ \tau}$$

Here, *duplast* τ, defined corecursively, traverses τ and duplicates its last state, if it is finite. Similarly, $\tau \models^* e$ and $\tau \not\models^* e$ traverse τ and evaluate e in the last state, if it is finite:

$$\frac{\tau \models^* e}{\sigma :: \tau \models^* e} \quad \frac{\sigma \models e}{\langle\sigma\rangle \models^* e} \qquad \frac{\tau \not\models^* e}{\sigma :: \tau \not\models^* e} \quad \frac{\sigma \not\models e}{\langle\sigma\rangle \not\models^* e}$$

(The rules for skip and sequence are very simple and appealing in this design.) The relation $\Rightarrow$ would then be defined uniformly by the rule

$$\frac{(s, \langle\sigma\rangle) \overset{*}{\Rightarrow} \tau}{(s, \sigma) \Rightarrow \tau}$$

It turns out that we can still derive $(\mathsf{while\ true\ do\ skip}, \sigma) \Rightarrow \tau$ for any τ. We can even derive $(\mathsf{while\ true\ do}\ x := x + 1, \sigma) \Rightarrow \tau$ for any τ.

The third alternative (Leroy and Grall use this technique in [11]) is most close to ours. It introduces, instead of our $\overset{*}{\Rightarrow}$ relation, an auxiliary relation *split*, defined coinductively by

$$\frac{}{\mathit{split}\ \langle\sigma\rangle\ \langle\sigma\rangle\ \sigma\ \langle\sigma\rangle} \qquad \frac{\tau \approx \tau'}{\mathit{split}\ (\sigma :: \tau)\ \langle\sigma\rangle\ \sigma\ (\sigma :: \tau')} \qquad \frac{\mathit{split}\ \tau\ \tau_0\ \sigma'\ \tau_1}{\mathit{split}\ (\sigma :: \tau)\ (\sigma :: \tau_0)\ \sigma'\ \tau_1}$$

so that *split* $\tau'\ \tau_0\ \sigma'\ \tau_1$ expresses that the trace τ' can be split into a concatenation of traces τ_0 and τ_1 glued together at a mid-state σ'. Then the evaluation relation is defined by replacing the uses of $\overset{*}{\Rightarrow}$ with *split*, e.g., the rule for the sequence statement would be:

$$\frac{\mathit{split}\ \tau'\ \tau_0\ \sigma'\ \tau_1 \quad (s_0, \sigma) \Rightarrow \tau_0 \quad (s_1, \sigma') \Rightarrow \tau_1}{(s_0; s_1, \sigma) \Rightarrow \tau'}$$

This third alternative does not cause any outright anomalies for While. But alarmingly s_1 has to be run from some (underdetermined) state within a run of $s_0; s_1$ even if the run of s_0 does not terminate. In a richer language with abnormal terminations, we get a serious problem: no evaluation is derived for (while true do skip); abort although the abort statement should not be reached.

$$\frac{}{\sigma \models \mathsf{true}} \qquad \frac{\neg(\sigma \models U)}{\sigma \models \neg U} \qquad \frac{\sigma \models U \quad \sigma \models V}{\sigma \models U \wedge V} \qquad \dots$$

$$\frac{}{\tau \models \mathsf{true}} \qquad \frac{\neg(\tau \models P)}{\tau \models \neg P} \qquad \frac{\tau \models P \quad \tau \models Q}{\tau \models P \wedge Q} \qquad \dots$$

$$\frac{\sigma \models U}{\langle\sigma\rangle \models \langle U\rangle} \qquad \frac{\tau \models P \quad \tau' \models_\tau Q}{\tau' \models P ** Q} \qquad \frac{\tau \models \langle\mathsf{true}\rangle}{\tau \models P^\dagger} \quad \frac{\tau \models P \quad \tau' \models_\tau P^\dagger}{\tau' \models P^\dagger}$$

$$\frac{\sigma \models U}{\sigma :: (\sigma[x \mapsto e]) \models U[x \mapsto e]} \qquad \frac{\sigma \models U}{\sigma :: \langle\sigma\rangle \models \langle U\rangle^2}$$

$$\frac{\tau \models P \quad \tau \downarrow \sigma}{\sigma \models \mathit{Last}\ P} \qquad \frac{\tau \downarrow \sigma}{\tau \models \mathit{finite}} \qquad \frac{\tau^\uparrow}{\tau \models \mathit{infinite}}$$

$$\frac{hd\ \tau = \sigma \quad \tau \models Q}{\tau \models_{\langle\sigma\rangle} Q}\ [\mathsf{flw\text{-}nil}] \qquad \frac{\tau' \models_\tau Q}{\sigma :: \tau' \models_{\sigma::\tau} Q}\ [\mathsf{flw\text{-}delay}]$$

$$\frac{\forall\sigma\,(\sigma \models U \to \sigma \models V)}{U \models V} \qquad \frac{\forall\tau\,(\tau \models P \to \tau \models Q)}{P \models Q}$$

Figure 2: Semantics of assertions

3. Hoare logic

We now proceed to the Hoare logic and its soundness and completeness proof. As we will base our consequence rule on semantic entailment rather than derivability in some fixed proof system, we sidestep the problem of its unavoidable incompleteness (due to the impossibility of complete axiomatization of any theory containing arithmetic). Regarding the choice of level of expressiveness of the assertion language, we deliberately keep the assertion language open, only making sure we have enough connectives to be able to express the strongest postcondition for any expressible precondition.

3.1. **Assertion language.** Our assertions will be about states and traces, i.e., expressing state and trace predicates. A state predicate U is simply a predicate on states. From a trace predicate P, we require that it is a setoid predicate, i.e., it must be unable to distinguish bisimilar traces.

We introduce a number of connectives for our assertion language. All these connectives yield setoid predicates. The inference rules of the Hoare logic rely on the availability of these connectives. Indeed, it was an intriguing exercise for us to come up with connectives that would be simple but expressive enough practically and at the same time allow us to prove the Hoare logic sound and complete constructively. The semantic definitions of these connectives are given in Figure 2.[4]

[4]We use the symbol $\models$ for visual highlighting of predicates. We are not defining a single satisfaction relation $\models$ for some fixed language of predicates, but a number of individual state/trace predicates and operations on such predicates. Some of them are defined inductively, some coinductively, some definitions are not recursive at all.

The two most primitive state (resp. trace) predicates are true and false, which are respectively true and false for any state (resp. trace). We can also use the standard connectives $\neg, \wedge, \vee$ and quantifiers $\forall, \exists$ to build state and trace predicates. The context disambiguates the overloaded notations for these state and trace predicates.

For a state predicate U, the singleton $\langle U \rangle$ is a trace predicate that is true of singleton traces given by a state satisfying U. In particular $\langle \mathsf{true} \rangle$ is true of any singleton trace.

For a state predicate U, the doubleton $\langle U \rangle^2$ is true of a doubleton trace whose two states are identical and satisfy U.

For a state predicate U, the update $U[x \mapsto e]$ is the strongest postcondition of the statement $x := e$ for the precondition U. It is true of a doubleton trace whose first state σ satisfies U and second state is obtained from the first by modifying the value of x to become $[\![e]\!]\ \sigma$.

For trace predicates P and Q, the chop $P ** Q$ is a trace predicate that is true, roughly speaking, of a trace τ' that has a prefix τ satisfying P, with the rest of τ' satisfying Q. But its definition is carefully crafted, so that Q is not checked, if τ is infinite (in which case necessarily $\tau \approx \tau'$), and this happens without case distinction on whether τ is finite. This effect is achieved with the premise $\tau' \models_\tau Q$. The relation $\tau' \models_\tau Q$ is defined coinductively. It traverses all of τ, making sure that it is a prefix of τ', and, upon possible exhaustion of τ in a finite number of steps, checks Q against the rest of τ'. This way the problem of deciding whether τ is finite is avoided, basically by postponing it, possibly infinitely.

Our chop operator is classically equivalent to the chop operator from interval temporal logic [13, 7] (cf. also the separating conjunction of separating logic). Indeed, classically, $\tau' \models P ** Q$ holds iff

- either, for some finite prefix τ of τ', we have $\tau \models P$ and $\tau'' \models Q$, where τ'' is the rest of τ',
- or τ' is infinite and $\tau' \models P$.

This is how the semantics of chop is defined in interval temporal logic. But it involves upfront decision of whether P will be satisfied by a finite or an infinite prefix of τ'. Our definition is fine-tuned for constructive reasoning.

For a trace predicate P, its iteration $P^\dagger$ is a trace predicate that is true of a trace which is a concatenation of a possibly infinite sequence of traces, each of which satisfies P. It is reminiscent of the Kleene star operator. It is defined by coinduction and takes into account both infiniteness of some single iteration and infinite repetition.

For a trace predicate P, *Last* P is a state predicate that is true of states that can be the last state of a finite trace satisfying P. Note that *Last* P is defined inductively.

Proposition 3.1. For any U, $\langle U \rangle$, $U[x \mapsto e]$ and $\langle U \rangle^2$ are setoid predicates. For any setoid predicates P, Q, $P ** Q$ is a setoid predicate. For any setoid predicate P, $P^\dagger$ is a setoid predicate.

Proof. That $\langle U \rangle$, $U[x \mapsto e]$ and $\langle U \rangle^2$ are setoid predicates follows from the definition. Easy coinduction proves that $P ** Q$ and $P^\dagger$ are setoid predicates when P and Q are. □

Proposition 3.2. For any U and V, if $U \models V$, then $\langle U \rangle \models \langle V \rangle$, $U[x \mapsto e] \models V[x \mapsto e]$ and $\langle U \rangle^2 \models \langle V \rangle^2$. For any setoid predicates P, P' and Q, if $P \models P'$, then $P ** Q \models P' ** Q$ and $Q ** P \models Q ** P'$. For any setoid predicates P and Q, if $P \models Q$, then $P^\dagger \models Q^\dagger$.

Proof. That $\langle U \rangle$, $U[x \mapsto e]$, $\langle U \rangle^2$ are monotone follows from the definition. Easy coinduction proves that $P ** Q$ and $P^\dagger$ are monotone. □

A number of logical consequences and equivalences hold about these connectives, to be proved in Lemma 3.3. We have the trivial equivalence: $\langle \mathsf{true} \rangle ** P \Leftrightarrow P \Leftrightarrow P ** \langle \mathsf{true} \rangle$. The chop operator is associative: $(P ** Q) ** R \Leftrightarrow P ** (Q ** R)$. The iterator operator $P^\dagger$ repeats P either zero times or once followed by further repetitions: $P^\dagger \Leftrightarrow \langle \mathsf{true} \rangle \vee (P ** P^\dagger)$. A trace is infinite if and only if false holds for any last state: $\mathit{infinite} \Leftrightarrow \mathsf{true} ** \langle \mathsf{false} \rangle$. We have $P ** \mathit{Last}\ P \Leftrightarrow P$. We also have $\mathit{Last}\ (P ** Q) \models \mathit{Last}\ Q$, but the converse does not hold. E.g., $\mathit{Last}\ \mathsf{true} \not\models \mathit{Last}\ (\mathsf{false} ** \mathsf{true})$. If every trace satisfying P is infinite, i.e., if $P \models \mathit{infinite}$, then $\mathit{Last}\ P \Leftrightarrow \mathsf{false}$.

We first observe two results, which are useful for later proofs.

Lemma 3.1. For any U and τ, τ', if $\tau' \models_\tau \langle U \rangle$, then $\tau \approx \tau'$.

Proof. By coinduction and inversion on $\tau' \models_\tau \langle U \rangle$. □

Lemma 3.2. For any τ and U, if for any σ, $\tau \downarrow \sigma$ implies $\sigma \models U$, then $\tau \models_\tau \langle U \rangle$.

Proof. By coinduction with case analysis on τ. □

Lemma 3.3. For any setoid predicates P, Q and R, we have

(1) $\langle \mathsf{true} \rangle ** P \Leftrightarrow P \Leftrightarrow P ** \langle \mathsf{true} \rangle$
(2) $(P ** Q) ** R \Leftrightarrow P ** (Q ** R)$
(3) $P^\dagger \Leftrightarrow \langle \mathsf{true} \rangle \vee (P ** P^\dagger)$
(4) $\mathit{infinite} \Leftrightarrow \mathsf{true} ** \langle \mathsf{false} \rangle$
(5) $P \Leftrightarrow P ** \mathit{Last}\ P$
(6) $\mathit{Last}\ (P ** Q) \models \mathit{Last}\ Q$
(7) $P \models \mathit{infinite}$, then $\mathit{Last}\ P \Leftrightarrow \mathsf{false}$.

Proof. (1) $\langle \mathsf{true} \rangle ** P \Leftrightarrow P$ follows from the definition. $P \models P ** \langle \mathsf{true} \rangle$ holds since we have $\tau \models_\tau \langle \mathsf{true} \rangle$ for any τ, proved by coinduction and case analysis on τ. Suppose $\tau' \models P ** \langle \mathsf{true} \rangle$. There exists τ such that $\tau \models P$ and $\tau' \models_\tau \langle \mathsf{true} \rangle$. By lemma 3.1 $\tau \approx \tau'$ holds, so must $\tau' \models P$ since P is a setoid predicate. This proves $P ** \langle \mathsf{true} \rangle \models P$.

(2) Suppose $\tau'' \models (P ** Q) ** R$. There exist τ and τ' such that $\tau \models P$ and $\tau' \models_\tau Q$ and $\tau'' \models_{\tau'} R$. We prove, for any τ_0, τ_1 and τ_2, $\tau_1 \models_{\tau_0} Q$ and $\tau_2 \models_{\tau_1} R$ imply $\tau_2 \models_{\tau_0} Q ** R$ by coinduction and inversion on $\tau_1 \models_{\tau_0} Q$. This yields $\tau'' \models_\tau Q ** R$ therefore $\tau'' \models P ** (Q ** R)$.

The converse is more subtle. Given $\tau'' \models P ** (Q ** R)$, we have to find a prefix τ' of τ'' that satisfies $P ** Q$ (and $\tau'' \models_{\tau'} R$). We do so by defining a function $\mathit{midp} : (\tau_1 \models_{\tau_0} P ** Q) \to \mathit{trace}$ by corecursion (we take τ_0 and τ_1 to be implicit parameters, inferred from the proof argument).[5]

$$\begin{aligned} &\mathit{midp}\ (\mathsf{flw\text{-}nil}\ \sigma\ \tau_0\ (\_ : \mathit{hd}\ \tau_0 = \sigma)\ (h : \tau_0 \models P ** Q)) \\ &= \mathit{let}\ \mathit{existT}\ \tau_1\ (\_ : \tau_1 \models P \wedge \tau_0 \models_{\tau_1} Q) = h\ \mathit{in}\ \tau_1 \\ &\mathit{midp}\ (\mathsf{flw\text{-}delay}\ \sigma\ \tau_0\ \tau_1\ (h : \tau_1 \models_{\tau_0} P ** Q) = \sigma :: (\mathit{midp}\ h) \end{aligned}$$

We then prove, for any τ_0, τ_1 and $h : \tau_1 \models_{\tau_0} P ** Q$, $\mathit{midp}\ h \models_{\tau_0} P$ and $\tau_1 \models_{\mathit{midp}\ h} Q$ hold by coinduction and inversion on h.

Now assume $\tau'' \models P ** (Q ** R)$. There exists τ such that $\tau \models P$ and $h : \tau'' \models_\tau Q ** R$. We have $\mathit{midp}\ h \models P ** Q$, since $\mathit{midp}\ h \models_\tau Q$. This together with $\tau'' \models_{\mathit{midp}\ h} R$ proves $\tau'' \models (P ** Q) ** R$, as required.

[5] existT is the constructor of Sigma-types in Coq.

(3) Follows from the definition.
(4) We prove an auxiliary condition: for any τ, *infinite* τ iff $\tau \models_\tau \langle \mathsf{false} \rangle$ by coinduction. *infinite* $\models \mathsf{true} ** \langle \mathsf{false} \rangle$ follows from the condition. $\mathsf{true} ** \langle \mathsf{false} \rangle \models$ *infinite* follows from the condition and Lemma 3.1.
(5) Suppose $\tau \models P$. By the definition of *Last* P, we have for any σ, $\tau \downarrow \sigma$ implies $\sigma \models$ *Last* P. We then deduce $\tau \models_\tau \langle$*Last* $P\rangle$ by Lemma 3.2, thus conclude $\tau \models P ** \langle$*Last* $P\rangle$.

    Conversely, suppose that $\tau' \models P * *$*Last* P, i.e., $\tau \models P$ and $\tau' \models_\tau \langle$*Last* $P\rangle$ for some τ. By Lemma 3.1 $\tau \approx \tau'$ holds, so must $\tau' \models P$ since P is a setoid predicate.
(6) Suppose $\sigma \models$ *Last* $(P ** Q)$. There exist τ and τ' such that $\tau' \downarrow \sigma$ and $\tau \models P$ and $\tau' \models_\tau Q$. We have to prove $\sigma \models$ *Last* Q. We do so by proving an auxiliary condition: for any σ_0 and τ_0, if $\tau_0 \downarrow \sigma_0$, then for any τ_1, $\tau_0 \models_{\tau_1} Q$ implies $\sigma_0 \models$ *Last* Q by induction on the derivation of $\tau_0 \downarrow \sigma_0$.
(7) Follows from *infinite* $\wedge$ *finite* $\models \mathsf{false}$.

□

3.2. **Inference rules.** The derivable judgements of the Hoare logic are given by the inductively interpreted inference rules in Figure 3. The proposition $\{U\}\ s\ \{P\}$ states derivability of the judgement. The intent is that $\{U\}\ s\ \{P\}$ should be derivable precisely when running a statement s from a initial state satisfying U is guaranteed to produce a trace satisfying P.

The rules for assignment and skip are self-explanatory.

The rule for sequence is defined in terms of the chop operator. The precondition V for the second statement s_1 is given by those states in which a run of the first statement s_0 may terminate. In particular, if $\{U\}\ s_0\ \{P\}$ and $P \models$ *infinite*, i.e., s_0 is necessarily diverging for the precondition U, then we have $\{U\}\ s_0\ \{P ** \langle \mathsf{false} \rangle\}$. In this case, from the derivability of $\{\mathsf{false}\}\ s_1\ \{Q\}$ for any Q, we get $\{U\}\ s_0; s_1\ \{P ** Q\}$ for any Q. But this makes sense, since $P ** Q \Leftrightarrow P$ as soon as $P \models$ *infinite*.

The rule for if-statement uses the doubleton operator in accordance with the operational semantics where we have chosen that testing the boolean guard grows the trace.

The rule for while-statement is inspired by the corresponding rule of the standard, state-based partial-correctness Hoare logic. It uses a loop invariant I. This is a state predicate that has to be true each time the boolean guard is about to be (re-)tested in a run of the loop. Accordingly, the precondition U should be stronger then I. Also, I must hold each time an iteration of s_t has finished, as enforced by having $P ** \langle I \rangle$ as the postcondition of s_t. The postcondition $\langle U \rangle^2 ** (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle$ of the loop consists of three parts. $\langle U \rangle^2$ accounts for the first test of the guard; $(P ** \langle I \rangle^2)^\dagger$ accounts for iterations of the loop body in alternation with re-tests of the guard (notice that that we are again using the doubleton operator); $\langle \neg e \rangle$ accounts for the state in which the last test of the guard is finished.

We have chosen to introduce a separate rule for instantiating auxiliary variables. Alternatively, we might have stated the consequence rule in a more general form, as suggested by Kleymann [12]; yet the separation facilitates formalization in Coq.

The various logical consequences and equivalences about the connectives suggest also further alternative and equivalent formulations. For instance, we could replace the rule for

$$\frac{}{\{U\}\ x := e\ \{U[x \mapsto e]\}} \quad \frac{}{\{U\}\ \mathsf{skip}\ \{\langle U\rangle\}} \quad \frac{\{U\}\ s_0\ \{P ** \langle V\rangle\} \quad \{V\}\ s_1\ \{Q\}}{\{U\}\ s_0; s_1\ \{P ** Q\}}$$

$$\frac{\{e \wedge U\}\ s_t\ \{P\} \quad \{\neg e \wedge U\}\ s_f\ \{P\}}{\{U\}\ \mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f\ \{\langle U\rangle^2 ** P\}}$$

$$\frac{U \models I \quad \{e \wedge I\}\ s_t\ \{P ** \langle I\rangle\}}{\{U\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle\}}$$

$$\frac{U \models U' \quad \{U'\}\ s\ \{P'\} \quad P' \models P}{\{U\}\ s\ \{P\}} \quad \frac{\forall z.\{U\}\ s\ \{P\}}{\{\exists z.\, U\}\ s\ \{\exists z.\, P\}}$$

Figure 3: Inference rules of Hoare logic

the while-statement by

$$\frac{\{e \wedge I\}\ s_t\ \{P ** \langle I\rangle\}}{\{I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\langle I\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle\}}$$

if we strengthened the consequence rule to

$$\frac{U \models U' \quad \{U'\}\ s\ \{P'\} \quad \langle U\rangle ** P' \models P}{\{U\}\ s\ \{P\}}$$

With our chosen rule for while, this strengthened version of consequence is admissible:

Lemma 3.4. For any U, s and P, $\{U\}\ s\ \{P\}$ then $\{U\}\ s\ \{\langle U\rangle ** P\}$.

Proof. We prove the following more general statement by induction on the derivation of $\{U\}\ s\ \{P\}$: for any U, s and P, $\{U\}\ s\ \{P\}$ then for any V, $\{U \wedge V\}\ s\ \{\langle V\rangle ** P\}$. □

We do not attempt to argue that our formulation is the best choice; yet we found that the present formulation is viable from the points-of-view of both the meta-theory and applicability of the logic.

3.3. **Soundness.** The soundness result states that any derivable Hoare triple is semantically valid.

Proposition 3.3 (Soundness)**.** For any s, U, P, σ, τ, if $\{U\}\ s\ \{P\}$ and $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ and then $\tau \models P$.

Proof. By induction on the derivation of $\{U\}\ s\ \{P\}$. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses that, for any σ, τ, $(s_0, \sigma) \Rightarrow \tau$ and $\sigma \models U$ imply $\tau \models P ** \langle V\rangle$, and that, for any σ, τ, $(s_1, \sigma) \Rightarrow \tau$ and $\sigma \models V$ imply $\tau \models Q$. We have to prove $\tau \models P ** Q$, given $\sigma \models U$ and $(s_0, \sigma) \Rightarrow \tau_0$ and $(s_1, \tau_0) \overset{*}{\Rightarrow} \tau_1$. By the induction hypothesis for s_0, we derive $h_0 : \tau_0 \models P$ and $\tau_0 \models_{\tau_0} \langle V\rangle$. We prove by coinduction an auxiliary lemma: for any τ, τ', $\tau \models_\tau \langle V\rangle$ and $(s_1, \tau) \overset{*}{\Rightarrow} \tau'$ give $\tau' \models_\tau Q$, using the induction hypothesis for s_1. The lemma gives us $h_1 : \tau_1 \models_{\tau_0} Q$. We can now close the case by h_0 and h_1.

$$
\begin{array}{l}
sp(x := e, U) = U[x \mapsto e] \\
sp(\mathsf{skip}, U) = \langle U \rangle \\
sp(s_0; s_1, U) = P ** sp(s_1, Last\ P) \ \textit{where}\ P = sp(s_0, U) \\
sp(\mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f, U) = \langle U \rangle^2 ** (sp(s_t, e \wedge U) \vee sp(s_f, \neg e \wedge U)) \\
sp(\mathsf{while}\ e\ \mathsf{do}\ s, U) = \langle U \rangle^2 ** (sp(s, e \wedge I) ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle \\
\qquad\qquad \textit{where}\ I = Inv(e, s, U)
\end{array}
$$

$$
\frac{\sigma \models U}{\sigma \models Inv(e, s, U)} \qquad \frac{V \models Inv(e, s, U) \quad \sigma \models Last\ (\langle Inv(e, s, U) \wedge e \rangle ** sp(s, V))}{\sigma \models Inv(e, s, U)}
$$

Figure 4: Strongest postcondition

- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: We are given as the induction hypothesis that for any σ and τ, $\sigma \models I \wedge e$ and $(\sigma, s) \Rightarrow \tau$ imply $\tau \models P ** \langle I \rangle$. We have to prove $\tau \models \langle U \rangle^2 ** (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle$, given $U \models I$ and $\sigma \models U$ and $(\mathsf{while}\ e\ \mathsf{do}\ s_t, \sigma) \Rightarrow \tau$. We do so by proving the following conditions by mutual coinduction:
  - for any σ and τ, if $\sigma \models I$ and $(\mathsf{while}\ e\ \mathsf{do}\ s, \sigma) \Rightarrow \sigma :: \tau$, then $\tau \models (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle$
  - for any τ and τ', if $\tau \models_\tau \langle I \rangle$ and $(\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau) \overset{*}{\Rightarrow} \tau'$, then $\tau' \models_\tau \langle I \rangle^2 ** (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle$.

□

3.4. **Completeness.** The completeness result states that any semantically valid Hoare triple is derivable. Following the standard approach (see, e.g., [18]) we define, for a given statement s and a given precondition U, a trace predicate $sp(s, U)$—the candidate strongest postcondition. Then we prove that $sp(U, s)$ is a postcondition according to the logic (i.e., $\{U\}\ s\ \{sp(U, s)\}$ is derivable) and that $sp(s, U)$ is semantically stronger than any other trace predicate that is a postcondition semantically. Completeness follows.

The trace predicate $sp(s, U)$ is defined by induction on s in Figure 4. The definition is mostly self-explanatory, as it mimics the inference rules of the logic, except that we need the loop-invariant $Inv(e, s, U)$. $Inv(e, s, U)$ characterizes the set of states that running $\mathsf{while}\ e\ \mathsf{do}\ s_t$ from a state satisfying U can reach at the boolean guard in finite steps.

For any s and U, the predicate $sp(s, U)$ is a monotone setoid predicate.

Lemma 3.5. For any s, U, τ, τ', if $\tau \models sp(s, U)$ and $\tau \approx \tau'$ then $\tau' \models sp(s, U)$.

Proof. By induction on the structure of s. □

Lemma 3.6. For any s, U, U', if $U \models U'$ then $sp(s, U) \models sp(s, U')$.

Proof. By induction on the structure of s. □

The following lemma states that any trace which satisfies $sp(s, U)$ has its first state satisfying U.

Lemma 3.7. For any s, U, τ, if $\tau \models sp(s, U)$ then $hd\ \tau \models U$.

Proof. By induction on the structure of s. □

The next lemma states that $Inv(e, s, U)$ is adequate.

Lemma 3.8. For any s, e, U, τ, $sp(s, Inv(e, s, U) \wedge e) \Leftrightarrow sp(s, Inv(e, s, U) \wedge e) ** \langle Inv(e, s, U)\rangle$.

Proof. Suppose $\tau \models sp(s, Inv(e, s, U) \wedge e)$. It suffices to prove $\tau \models_\tau \langle Inv(e, s, U)\rangle$. However, we have $\tau \models \langle Inv(e, s, U) \wedge e\rangle ** sp(s, Inv(e, s, U))$ by Lemma 3.6 and Lemma 3.7. By the definition of Inv, we have for any σ, $\tau \downarrow \sigma$ implies $\sigma \models Inv(e, s, U)$. Therefore we conclude $\tau \models_\tau \langle Inv(e, s, U)\rangle$ by Lemma 3.2. The other direction, $sp(s, Inv(e, s, U) \wedge e) ** \langle Inv(e, s, U)\rangle \models sp(s, Inv(e, s, U) \wedge e)$ holds by Lemma 3.1. □

We are now ready to establish that $sp(s, U)$ is a postcondition according to the Hoare logic.

Lemma 3.9. For any s, U, $\{U\}\ s\ \{sp(s, U)\}$.

Proof. By induction on s. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses that, for any U_0, $\{U_0\}\ s_0\ \{sp(s_0, U_0)\}$ and $\{U_0\}\ s_1\ \{sp(s_1, U_0)\}$. We have to prove $\{U\}\ s_0; s_1\ \{P ** sp(s_1, Last\ P)\}$ where $P = sp(s_0, U)$. By ind. hyp. we have $\{U\}\ s_0\ \{P\}$, thus $\{U\}\ s_0\ \{P ** Last\ P\}$ by (5) of Lemma 3.3 and the consequence rule. We therefore close the case with $\{Last\ P\}\ s_1\ \{sp(s_1, Last\ P)\}$ given by the induction hypothesis.
- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: We are given as the induction hypothesis that, for any U_0, $\{U_0\}\ s_t\ \{sp(s_t, U_0)\}$. We have to prove $\{U\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\langle U\rangle^2 ** (sp(s_t, e \wedge I) ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle\}$ where $I = Inv(e, s_t, U)$. It is sufficient to prove $\{e \wedge I\}\ s_t\ \{(sp(s_t, e \wedge I) ** \langle I\rangle)\}$, which follows from the induction hypothesis and Lemma 3.8.

□

Following the standard route, it should remain to prove the following condition:
for any s, U, P, if for all σ, τ, $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$, then $sp(s, U) \models P$.
This will be an immediate corollary from Lemma 3.7 and the following lemma, stating that any trace which satisfies $sp(s, U)$ is in fact produced by a run of s.

Lemma 3.10. For any s, U, τ, if $\tau \models sp(s, U)$ then $(s, hd\ \tau) \Rightarrow \tau$.

Proof. By induction on s. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses that, for any U', τ', $\tau' \models sp(s_0, U')$ (resp. $\tau' \models sp(s_1, U')$) implies $(s_0, hd\ \tau') \Rightarrow \tau'$ (resp. $(s_1, hd\ \tau') \Rightarrow \tau'$). We have to prove $(s_0; s_1, hd\ \tau) \Rightarrow \tau$, given $\tau \models sp(s_0; s_1, U)$, which unfolds into $\tau_0 \models sp(s_0, U)$ and $\tau \models_{\tau_0} sp(s_1, Last\ (sp(s_0, U)))$. By the induction hypothesis for s_0, we have $(s_0, hd\ \tau_0) \Rightarrow \tau_0$. Using the induction hypothesis for s_1, we prove by coinduction that, for any τ_1, τ_2, $\tau_2 \models_{\tau_1} sp(s_1, Last\ (sp(s_0, U)))$ implies $(s_1, \tau_1) \overset{*}{\Rightarrow} \tau_2$, thereby we close the case.
- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: We are given as the induction hypothesis that, for any U', τ', $\tau' \models sp(s_t, U')$ implies $(s_t, hd\ \tau') \Rightarrow \tau'$. We have to prove $(\mathsf{while}\ e\ \mathsf{do}\ s_t, hd\ \tau) \Rightarrow \tau$, given $\tau \models \langle U\rangle^2 ** (sp(s, e \wedge I) ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle$ where $I = Inv(e, s, U)$. We do so by proving the following two conditions simultaneously by mutual coinduction:
  - for any τ, $\tau \models (sp(s, e \wedge I) ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle$ implies $(\mathsf{while}\ e\ \mathsf{do}\ s_t, hd\ \tau) \Rightarrow hd\ \tau :: \tau$,
  - for any τ and τ', $\tau' \models_\tau \langle I\rangle^2 ** (sp(s, e \wedge I) ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle$ implies $(\mathsf{while}\ e\ \mathsf{do}\ s_t, \tau) \overset{*}{\Rightarrow} \tau'$.

□

Corollary 3.1. For any s, U, P, if for all σ, τ, $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$, then $sp(s, U) \models P$.

Completeness is proved as a corollary of the last two lemmata.

Proposition 3.4 (Completeness)**.** For any s, U, P, if for all σ, τ, $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$, then $\{U\}\ s\ \{P\}$.

Proof. Assume that for all σ, τ, $\sigma \models U$ and $(s, \sigma) \Rightarrow \tau$ imply $\tau \models P$. By Corollary 3.1, we have that $sp(s, U) \models P$. By Lemma 3.9, we have $\{U\}\ s\ \{sp(s, U)\}$. Applying consequence, we get $\{U\}\ s\ \{P\}$. □

4. Relation to the standard partial-correctness and total-correctness Hoare logics

It is easy to see, by going through soundness and completeness results, that our trace-based Hoare logic is a conservative extension of the standard, state-based partial-correctness and total-correctness Hoare logics. But more can be said. The derivations in these two logics are directly transformable into derivations in our logic, preserving their structure, without invention of new invariants or variants. And in the converse direction, derivations in our logic are transformable into derivations into the standard logics in a way that removes from postconditions information about intermediate states. In this direction, the variant for a while-loop is obtained by bounding the length of traces satisfying the trace invariant of the loop.

Concerning total correctness, we use two variations of the while-rule. In the forward transformation, we use a version of the while-rule with a dedicated variant (a natural-valued *function* on states) whereas, in the backward transformation, we work with a version where the invariant (a state predicate) is made dependent on a natural number (i.e., becomes a *relation* between states and naturals; crucially, there is no functionality requirement: in the same state, it can be satisfied by zero or one or several naturals). The two alternative while-rules for total correctness are:

$$\frac{\forall n : nat\ \{e \wedge I \wedge t = n\}\ s_t\ \{I \wedge t < n\}}{\{I \wedge t = m\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{I \wedge \neg e \wedge t \leq m\}}\ \text{while-fun}$$

and

$$\frac{\forall n : nat\ \{e \wedge J\, n\}\ s_t\ \{\exists k.\, k < n \wedge J\, k\}}{\{J\, m\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\exists k.\, k \leq m \wedge J\, k \wedge \neg e\}}\ \text{while-rel}$$

There is a reason for this discrepancy, which reflects our compromise between being constructive approach and striving for purely syntactic translations. We will discuss it after having presented the transformations.

We have tried to fine-tune the inference rules in the different Hoare logics and the transformations between them for smoothness. There is quite some room for variations in them. The transformations are quite sensitive to the exact division of labor in the source and target Hoare logics between the rules for the statement constructors and the consequence rule, but the effects of the possible variations are mostly inessential.

For reference, the inference rules of the state-based logics appear in the Appendix.

4.1. **Embeddings of the standard Hoare logics into the trace-based logic.** We formalize our claim of embeddibility of the standard Hoare logics in the following two propositions, whose direct proofs are algorithms for the transformations.

Proposition 4.1 states that, if $\{U\}\ s\ \{Z\}$ is a derivable partial-correctness judgement, then $\{U\}\ s\ \{\mathsf{true} ** \langle Z\rangle\}$ is derivable in our logic. The trace predicate $\mathsf{true} ** \langle Z\rangle$ indicates that Z holds of any state that is reachable by traversing, in a finite number of steps, the whole trace τ produced by running s. Classically, this amounts to the condition of Z being true of the last state of τ, if τ is finite and hence has one; if τ is infinite, then nothing is required.

Proposition 4.2 states that, if $\{U\}\ s\ \{Z\}$ is a derivable total-correctness judgement, then $\{U\}\ s\ \{\mathit{finite} ** \langle Z\rangle\}$ is derivable in our logic (in fact, it states a little more). The trace predicate $\mathit{finite} ** \langle Z\rangle$ expresses that the trace τ produced by running s is finite and Z holds of the last state of τ; the finiteness of τ guarantees the existence of this last state.

Proposition 4.1. For any U, s and Z, if $\{U\}\ s\ \{Z\}$ is derivable in the partial-correctness Hoare logic, then $\{U\}\ s\ \{\mathsf{true} ** \langle Z\rangle\}$ is derivable in the trace-based Hoare logic.

Proof. By induction on the derivation of $\{U\}\ s\ \{Z\}$. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses $\{U\}\ s_0\ \{\mathsf{true} ** \langle V\rangle\}$ and $\{V\}\ s_1\ \{\mathsf{true} ** \langle Z\rangle\}$. We have to prove $\{U\}\ s_0; s_1\ \{\mathsf{true} ** \langle Z\rangle\}$, which is derived by
$$\dfrac{\dfrac{\begin{array}{cc}\vdots\ \mathrm{IH}_0 & \vdots\ \mathrm{IH}_1\\ \{U\}\ s_0\ \{\mathsf{true} ** \langle V\rangle\} & \{V\}\ s_1\ \{\mathsf{true} ** \langle Z\rangle\}\end{array}}{\{U\}\ s_0; s_1\ \{\mathsf{true} ** \mathsf{true} ** \langle Z\rangle\}}}{\{U\}\ s_0; s_1\ \{\mathsf{true} ** \langle Z\rangle\}}$$
- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: We are given as the induction hypothesis $\{e \wedge I\}\ s_t\ \{\mathsf{true} ** \langle I\rangle\}$. We have to prove $\{e \wedge I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\mathsf{true} ** \langle I \wedge \neg e\rangle\}$, which is derived by
$$\dfrac{\dfrac{\begin{array}{c}\vdots\ \mathrm{IH}_t\\ \{e \wedge I\}\ s\ \{\mathsf{true} ** \langle I\rangle\}\end{array}}{\{I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\langle I\rangle^2 ** (\mathsf{true} ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle\}}}{\{I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\mathsf{true} ** \langle I \wedge \neg e\rangle\}}\ (1)$$
(1) We have
$$\langle I\rangle^2 ** \mathsf{true} ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle \Rightarrow \mathsf{true} ** \langle I\rangle ** \langle \neg e\rangle \Leftrightarrow \mathsf{true} ** \langle I \wedge \neg e\rangle$$

□

For the embedding of total-correctness derivations, we prove a slightly stronger statement to have the induction go through.

Proposition 4.2. For any U, s and Z, if $\{U\}\ s\ \{Z\}$ is derivable in the total-correctness Hoare logic with while-fun, then for any W, $\{U \wedge W\}\ s\ \{\langle W\rangle ** \mathit{finite} ** \langle Z\rangle\}$ is derivable in the trace-based Hoare logic.

Proof. By induction on the derivation of $\{U\}\ s\ \{Z\}$. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses that, for any W_0, $\{U \wedge W_0\}\ s_0\ \{\langle W_0\rangle ** \mathit{finite} ** \langle V\rangle\}$ and, for any W_1, $\{V \wedge W_1\}\ s_1\ \{\langle W_1\rangle ** \mathit{finite} ** \langle Z\rangle\}$. We have to prove that, for any W, $\{U \wedge W\}\ s_0; s_1\ \{\langle W\rangle ** \mathit{finite} ** \langle Z\rangle\}$. This is done by the derivation

$$\dfrac{\dfrac{\vdots\ \mathrm{IH}_0\, W}{\{U \wedge W\}\ s_0\ \{\langle W\rangle ** \mathit{finite} ** \langle V\rangle\}} \qquad \dfrac{\vdots\ \mathrm{IH}_1\, V}{\{V \wedge V\}\ s_1\ \{\langle V\rangle ** \mathit{finite} ** \langle Z\rangle\}}}{\dfrac{\{U \wedge W\}\ s_0; s_1\ \{\langle W\rangle ** \mathit{finite} ** \langle V\rangle ** \mathit{finite} ** \langle Z\rangle\}}{\{U \wedge W\}\ s_0; s_1\ \{\langle W\rangle ** \mathit{finite} ** \langle Z\rangle\}}\ (1)}$$

  (1) We have

$$\mathit{finite} ** \langle V\rangle ** \mathit{finite} \Rightarrow \mathit{finite} ** \mathit{finite} \Leftrightarrow \mathit{finite}$$

- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: We are given as the induction hypothesis that, for all $n : \mathit{nat}$ and W_t, $\{e \wedge I \wedge t = n \wedge W_t\}\ s_t\ \{\langle W_t\rangle ** \mathit{finite} ** \langle I \wedge t < n\rangle\}$. We have to prove that, for any W, $\{I \wedge t = m \wedge W\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{W ** \mathit{finite} ** \langle I \wedge \neg e \wedge t \leq m\rangle\}$. This is accomplished by the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ \forall n.\, \mathrm{IH}_t\, n\, (t = n \wedge t \leq m)}{\forall n.\, \{e \wedge I \wedge t = n \wedge t = n \wedge t \leq m\}\ s_t\ \{\langle t = n \wedge t \leq m\rangle ** \mathit{finite} ** \langle I \wedge t < n\rangle\}}\ (2)}{\forall n.\, \{e \wedge I \wedge t = n \wedge t \leq m\}\ s_t\ \{\langle t = n\rangle ** \mathit{finite} ** \langle I \wedge t < n \wedge t \leq m\rangle\}}}{\{\exists n.\, e \wedge I \wedge t = n \wedge t \leq m\}\ s_t\ \{\exists n.\, \langle t = n\rangle ** \mathit{finite} ** \langle I \wedge t < n \wedge t \leq m\rangle\}}}{\{e \wedge I \wedge t \leq m\}\ s_t\ \{(\exists n.\langle t = n\rangle ** \mathit{finite} ** \langle t < n\rangle) ** \langle I \wedge t \leq m\rangle\}}$$

$$\dfrac{\{I \wedge t = m \wedge W\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\langle I \wedge t = m \wedge W\rangle^2 ** ((\exists n.\langle t = n\rangle ** \mathit{finite} ** \langle t < n\rangle) ** \langle I \wedge t \leq m\rangle^2)^\dagger ** \langle \neg e\rangle\}}{\{I \wedge t = m \wedge W\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{W ** \mathit{finite} ** \langle I \wedge \neg e \wedge t \leq m\rangle\}}\ (1)$$

  (1) The repetition gives rise to a non-empty colist of values of t, which is strictly decreasing and must hence be of finite length. The concatenation of a finite colist of finite traces is finite.
  (2) If $t = n$ and $t \leq m$ in the first state of a trace, then $n \leq m$ (everywhere), so in the last state $t < n$ gives $t < m$, which can be weakened to $t \leq m$.

□

Corollary 4.1. For any U, s and Z, if $\{U\}\ s\ \{Z\}$ is derivable in the total-correctness Hoare logic with while-fun, then $\{U\}\ s\ \{\mathit{finite} ** \langle Z\rangle\}$ is derivable in the trace-based Hoare logic.

Proof. Immediate from Proposition 4.2 by choosing $W = \mathsf{true}$. □

4.2. **Projections of the trace-based logic into standard Hoare logics.** Given that derivations in the standard Hoare logics can be transformed into the trace-based Hoare logic, it is natural to wonder, if it is also possible to translate derivations in the converse direction. Of course, in this direction we would expect some loss (or displacement) of information. Reducing a condition on traces into a condition on those last states that happen to exist or into a condition that also requires their existence must lose or displace the constraints on the intermediate states. So transformations like this are bound to be lossy, but they can still be useful.

We now proceed to demonstrating that meaningful transformations ("projections") from the trace-based logic to the standard logic are indeed possible.

We show that, if $\{U\}\ s\ \{P\}$ in the trace-based Hoare logic, then $\{U\}\ s\ \{\mathit{Last}\ P\}$ in the partial-correctness Hoare logic and $\{U \wedge \lceil P \rceil m\}\ s\ \{\mathit{Last}\ P\}$ in the total-correctness Hoare logic. We will shortly define $\lceil P \rceil m$ mathematically, but intuitively, the total-correctness judgement states that s terminates in a state satisfying $\mathit{Last}\ P$ from any initial state σ such that U holds in σ and any σ-headed trace satisfying P has length at most m. Since m is universally quantified on the top level, we can actually derive $\{U \wedge \exists m.\, \lceil P \rceil m\}\ s\ \{\mathit{Last}\ P\}$, stating that s terminates in a state satisfying $\mathit{Last}\ P$ from any initial state σ such that U holds and the length of σ-headed traces τ satisfying P is bounded.

Proposition 4.3. For any U, s and P, if $\{U\}\ s\ \{P\}$ in the trace-based Hoare logic, then for any W, $\{U \wedge W\}\ s\ \{\mathit{Last}\ (\langle W \rangle ** P)\}$ is derivable in the partial-correctness Hoare logic.

Proof. By induction on the derivation of $\{U\}\ s\ \{P\}$. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses that, for any W_0, $\{U \wedge W_0\}\ s_0\ \{\mathit{Last}\ (\langle W_0 \rangle ** P ** \langle V \rangle)\}$ and, for any W_1, $\{V \wedge W_1\}\ s_1\ \{\mathit{Last}\ (\langle W_1 \rangle ** Q)\}$. We have to show that, for any W, $\{U \wedge W\}\ s_0; s_1\ \{\mathit{Last}\ (\langle W \rangle ** P ** Q)\}$. Let $V' = \mathit{Last}\ (\langle W \rangle ** P ** \langle V \rangle)$. We have the derivation

$$\dfrac{\dfrac{\{U \wedge W\}\ s_0\ \{V'\} \quad \dfrac{\vdots\ \mathrm{IH}_1\, V'}{\{V \wedge V'\}\ s_1\ \{\mathit{Last}\ (\langle V' \rangle ** Q)\}}}{\{V'\}\ s_1\ \{\mathit{Last}\ (\langle V' \rangle ** Q)\}}\ (2)}{\dfrac{\{U \wedge W\}\ s_0; s_1\ \{\mathit{Last}\ (\langle V' \rangle ** Q)\}}{\{U \wedge W\}\ s_0; s_1\ \{\mathit{Last}\ (\langle W \rangle ** P ** Q)\}}\ (1)}$$

  (1) It is easy to see that, for any P and Q, $\mathit{Last}\ (\langle \mathit{Last}\ P \rangle ** Q) \Leftrightarrow \mathit{Last}\ (P ** Q)$ holds. Therefore, noticing that Last is monotone, we have

$$\begin{aligned} & \mathit{Last}\ (\langle V' \rangle ** Q) \\ = \ & \mathit{Last}\ (\langle \mathit{Last}\ (\langle W \rangle ** P ** \langle V \rangle) \rangle ** Q) \\ \Leftrightarrow \ & \mathit{Last}\ (\langle W \rangle ** P ** \langle V \rangle ** Q) \\ \Rightarrow \ & \mathit{Last}\ (\langle W \rangle ** P ** Q) \end{aligned}$$

  (2) By Lemma 3.3 (6), $V' = \mathit{Last}\ (\langle W \rangle ** P ** \langle V \rangle) \models V$, therefore $V' \models V \wedge V'$.
- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: We are given as the induction hypothesis that, for any W_t, $\{I \wedge e \wedge W_t\}\ s_t\ \{\mathit{Last}\ (\langle W_t \rangle ** P ** \langle I \rangle)\}$. We have to prove that, for any W, $\{U \wedge W\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\mathit{Last}\ (\langle W \rangle ** \langle U \rangle^2 ** (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle)\}$, given $U \models I$. Let $I' = \mathit{Last}\ (\langle W \rangle ** \langle U \rangle^2 ** (P ** \langle I \rangle^2)^\dagger)$. We close the case by the derivation

$$\dfrac{\dfrac{\dfrac{\vdots\ \mathrm{IH}_t\, I'}{\{e \wedge I \wedge I'\}\ s_t\ \{\mathit{Last}\ (\langle I' \rangle ** P ** \langle I \rangle)\}}\ (2)}{\{e \wedge I'\}\ s_t\ \{I'\}}}{\{I'\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{I' \wedge \neg e\}}$$
$$\dfrac{}{\{U \wedge W\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\mathit{Last}\ (\langle W \rangle ** \langle U \rangle^2 ** (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle)\}}\ (1)$$

(1) Using (3) of Lemma 3.3, we have

$$\begin{array}{rcl} U \wedge W & & \\ & \Leftrightarrow & \mathit{Last}\ (\langle W\rangle \mathbin{**} \langle U\rangle^2 \mathbin{**} \langle \mathsf{true}\rangle) \\ & \Rightarrow & \mathit{Last}\ (\langle W\rangle \mathbin{**} \langle U\rangle^2 \mathbin{**} (P \mathbin{**} \langle I\rangle^2)^\dagger) \\ & = & I' \end{array}$$

(2) We have

$$\begin{array}{rcl} \mathit{Last}\ (\langle I'\rangle \mathbin{**} P \mathbin{**} \langle I\rangle) & & \\ & = & \mathit{Last}\ (\langle \mathit{Last}\ (\langle W\rangle \mathbin{**} \langle U\rangle^2 \mathbin{**} (P \mathbin{**} \langle I\rangle^2)^\dagger)\rangle \mathbin{**} P \mathbin{**} \langle I\rangle) \\ & \Leftrightarrow & \mathit{Last}\ (\langle W\rangle \mathbin{**} \langle U\rangle^2 \mathbin{**} (P \mathbin{**} \langle I\rangle^2)^\dagger \mathbin{**} P \mathbin{**} \langle I\rangle) \\ & \Leftrightarrow & \mathit{Last}\ (\langle W\rangle \mathbin{**} \langle U\rangle^2 \mathbin{**} (P \mathbin{**} \langle I\rangle^2)^\dagger \mathbin{**} P \mathbin{**} \langle I\rangle^2) \\ & \Rightarrow & \mathit{Last}\ (\langle W\rangle \mathbin{**} \langle U\rangle^2 \mathbin{**} (P \mathbin{**} \langle I\rangle^2)^\dagger) \\ & = & I' \end{array}$$

□

Corollary 4.2. For any U, s and P, if $\{U\}\ s\ \{P\}$ then $\{U\}\ s\ \{\mathit{Last}\ P\}$ is derivable in the partial correctness Hoare logic.

Proof. Immediate from Proposition 4.3 by instantiating $W = \mathsf{true}$. □

To define the assertion translation for the projection of the trace-based Hoare logic into the total-correctness Hoare logic, we need two new connectives. The inductively defined trace predicate $\mathit{len}\, n$ is true of finite traces with length at most n. Given a trace predicate P, the state predicate $\lceil P \rceil n$ is defined to be true of a state σ if a every trace headed by σ and satisfying P also satisfies $\mathit{len}\, n$.

$$\frac{}{\langle \sigma\rangle \models \mathit{len}\, n} \qquad \frac{\tau \models \mathit{len}\, n}{\sigma :: \tau \models \mathit{len}\, (n+1)}$$

$$\frac{\forall \tau.\, (\mathit{hd}\ \tau = \sigma \wedge \tau \models P \Rightarrow \tau \models \mathit{len}\, n)}{\sigma \models \lceil P \rceil n}$$

Lemma 4.1. For any n, P and Q, if $Q \models P$ then $\lceil P \rceil n \models \lceil Q \rceil n$.

Proposition 4.4. For any U, s and P, if $\{U\}\ s\ \{P\}$ in the trace-based Hoare logic, then for any $m : \mathit{nat}, W$ and R, $\{U \wedge W \wedge \lceil P \mathbin{**} R \rceil m\}\ s\ \{\mathit{Last}\ (\langle W\rangle \mathbin{**} P) \wedge \lceil R \rceil m\}$ is derivable in the total-correctness Hoare logic with while-rel.

Proof. By induction on the derivation of $\{U\}\ s\ \{P\}$. We show the main cases of sequence and while.

- $s = s_0; s_1$: We are given as the induction hypotheses that, for any m_0, W_0 and R_0, $\{U \wedge W_0 \wedge \lceil P \mathbin{**} \langle V\rangle \mathbin{**} R_0 \rceil m_0\}\ s_0\ \{\mathit{Last}\ (\langle W_0\rangle \mathbin{**} P \mathbin{**} \langle V\rangle) \wedge \lceil R_0 \rceil m_0\}$ and, for any m_1, W_1 and R_1, $\{V \wedge W_1 \wedge \lceil Q \mathbin{**} R_1 \rceil m_1\}\ s_1\ \{\mathit{Last}\ (\langle W_1\rangle \mathbin{**} Q) \wedge \lceil R_1 \rceil m_1\}$. We have to prove that, for any m, W and R, $\{U \wedge W \wedge \lceil P \mathbin{**} Q \mathbin{**} R \rceil m\}\ s_0; s_1\ \{\mathit{Last}\ (\langle W\rangle \mathbin{**} P \mathbin{**} Q) \wedge \lceil R \rceil m\}$. Let $V' = \mathit{Last}\ (\langle W\rangle \mathbin{**} P \mathbin{**} \langle V\rangle)$. We close the case by the

derivation

$$\dfrac{\dfrac{\dfrac{\vdots\ \mathrm{IH}_0\ W, Q ** R}{\{U \wedge W \wedge \lceil P ** \langle V\rangle ** Q ** R\rceil m\} s_0}}{\{U \wedge W \wedge \lceil P ** Q ** R\rceil m\} s_0}\ (1) \quad \dfrac{\dfrac{\vdots\ \mathrm{IH}_1\ V', R}{\{V \wedge V' \wedge \lceil Q ** R\rceil m\} s_1}}{\{V' \wedge \lceil Q ** R\rceil m\} s_1}\ (2)}{\dfrac{\{U \wedge W \wedge \lceil P ** Q ** R\rceil m\}\ s_0; s_1\ \{\mathit{Last}\ (\langle W\rangle ** P ** Q) \wedge \lceil R\rceil m\}}{\{U \wedge W \wedge \lceil P ** Q ** R\rceil m\}\ s_0; s_1\ \{\mathit{Last}\ (\langle W\rangle ** P ** Q) \wedge \lceil R\rceil m\}}\ (3)}$$

(1) $\lceil P ** Q ** R\rceil m \models \lceil P ** \langle V\rangle ** Q ** R\rceil m$ follows from Lemma 4.1 and $P ** \langle V\rangle ** Q ** R \models P ** Q ** R$.

(2) $V' \models V \wedge V'$ holds because $V' = \mathit{Last}\ (\langle W\rangle ** P ** \langle V\rangle) \models V$.

(3) We have

$$\begin{array}{ll} & \mathit{Last}\ (\langle V'\rangle ** Q) \\ = & \mathit{Last}(\langle \mathit{Last}\ (\langle W\rangle ** P ** \langle V\rangle)\rangle ** Q) \\ \Leftrightarrow & \mathit{Last}\ (\langle W\rangle ** P ** \langle V\rangle ** Q) \\ \Rightarrow & \mathit{Last}\ (\langle W\rangle ** P ** Q) \end{array}$$

- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$. We are given as the induction hypothesis that, for any m_t, W_t and R_t, $\{e \wedge I \wedge W_t \wedge \lceil P ** \langle I\rangle ** R_t\rceil m_t\}\ s_t\ \{\mathit{Last}\ (\langle W_t\rangle ** P ** \langle I\rangle) \wedge \lceil R_t\rceil m_t\}$. We also have $U \models I$. We have to prove that for any m, W and R, $\{U \wedge W \wedge \lceil \langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil m\}$ $\mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\mathit{Last}\ (\langle W\rangle ** \langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle) \wedge \lceil R\rceil m\}$.

  Let $J_0 = \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger)$, $J_1\, n = \lceil (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil n$ and $Q = \langle I\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R$.

  We have, for any n,

$$\begin{array}{ll} & e \wedge J_0 \wedge J_1\, n \\ = & e \wedge \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger) \wedge \lceil (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil n \\ \Rightarrow & e \wedge \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger) \wedge \lceil P ** \langle I\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil n \\ \Leftrightarrow & e \wedge I \wedge \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger) \wedge \lceil P ** \langle I\rangle ** \langle I\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil n \\ = & e \wedge I \wedge J_0 \wedge \lceil P ** \langle I\rangle ** Q\rceil n \end{array}$$

  and

$$\begin{array}{ll} & \mathit{Last}\ (\langle J_0\rangle ** P ** \langle I\rangle) \wedge \lceil Q\rceil n \\ = & \mathit{Last}\ (\langle \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger)\rangle ** P ** \langle I\rangle) \wedge \lceil Q\rceil n \\ \Rightarrow & \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger) \wedge \lceil Q\rceil n \\ = & J_0 \wedge \lceil \langle I\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil n \qquad (\text{and } n \neq 0) \\ \Rightarrow & J_0 \wedge \lceil (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil (n-1) \\ = & J_0 \wedge J_1\, (n-1) \\ \Rightarrow & \exists k.\, k < n \wedge J_0 \wedge J_1\, k \end{array}$$

  Thus by consequence and the induction hypothesis (with m_t, W_t and R_t respectively instantiated by n, J_0 and Q), we deduce

$$\forall n.\, \{e \wedge J_0 \wedge J_1\, n\}\ s_t\ \{\exists k.\, k < n \wedge J_0 \wedge J_1\, k\}$$

By the rule for the while-statement by taking $J\,n$ to be $J_0 \wedge J_1\,n$, we deduce

$$\{J_0 \wedge J_1\,m\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\exists k.\, k \leq m \wedge J_0 \wedge J_1\,k \wedge \neg e\}$$

However, we have

$$\begin{array}{ll} & U \wedge W \wedge \lceil \langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil m \\ \Rightarrow & \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger) \wedge \lceil (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil m \\ = & J_0 \wedge J_1\,m \end{array}$$

and

$$\begin{array}{ll} & J_0 \wedge J_1\,m \wedge \neg e \\ = & \mathit{Last}\ (\langle U \wedge W\rangle^2 ** (P ** \langle I\rangle^2)^\dagger) \wedge \lceil (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil m \wedge \neg e \\ \Rightarrow & \mathit{Last}\ (\langle W\rangle ** \langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle) \wedge \lceil R\rceil m \end{array}$$

So, by consequence, we have

$$\begin{array}{l} \{U \wedge W \wedge \lceil \langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle ** R\rceil m\}\mathsf{while}\ e\ \mathsf{do}\ s_t \\ \quad \{\mathit{Last}\ (\langle W\rangle ** \langle U\rangle^2 ** (P ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle) \wedge \lceil R\rceil m\} \end{array}$$

as we wanted.

□

Corollary 4.3. For any U, s and P, if $\{U\}\ s\ \{P\}$ in the trace-based Hoare logic, then $\{U \wedge \exists m.\, \lceil P\rceil m\}\ s\ \{\mathit{Last}\ P\}$ is derivable in the total-correctness Hoare logic with while-rel.

Proof. From Proposition 4.3 by taking $W = \mathsf{true}$, $R = \langle \mathsf{true}\rangle$ and then invoking the admissible rule

$$\frac{\forall m.\, \{U\}\ s\ \{Z\}}{\{\exists m.\, U\}\ s\ \{\exists m.\, Z\}}$$

of the total-correctness Hoare logic. □

4.3. **Discussion.** Let us now discuss the transformations from and to the total-correctness Hoare logic. We used two different rules for while. In the forward direction (the embedding), we used while-fun, whereas in the backward direction (the projection), we used while-rel.

One way to complete the cycle would be to show that while-fun is derivable from while-rel (or at least admissible with respect to it). But this is problematic constructively.

We would have to define the I and t of the while-fun rule from the J of the while-rel rule. It is natural to define the invariant I as $I = \exists m.\, J\,m$. The variant t should be defined as

$$t = \begin{cases} \text{least } k \text{ such that } J\,k & \text{if } \exists m.\, J\,m \\ 17 \text{ or any other natural} & \text{if } \neg\exists m.\, J\,m \end{cases}$$

This is a perfectly good classical definition. But constructively, there are two issues with it. First, in the first case, we need to find the least k such that $J\,k$. Luckily, we have a bound for this search, which is given by the witness of $\exists m.\, J\,m$. But in order to search, we must be able to decide where $J\,k$ or $\neg J\,k$ for $k < m$, so J has to be decidable.

Second and worse, in order to choose between the two cases, we must decide whether $\exists m.\, J\,m$ or $\neg\exists m.\, J\,m$. This may well sound like too much too ask. Luckily, ot is unnecessary too. The observation to make is that our rule while-fun is not fine-tuned for constructive reasoning. We only ever need t in contexts where $I = \exists m.\, J\,m$ can be assumed to hold. So a proper rule should not say $I \wedge t = n$, but rather $\Sigma p : I.\, t\,p = n$ etc.: the variant t should

be able to depend on a proof p of the invariant and one is always available when t is used. The better functional-variant based while-rule would be:

$$\frac{\forall n : nat\ \{e \wedge \Sigma p : I.\, t\, p = n\}\ s_t\ \{\Sigma p : I.\, t\, p < n\}}{\{\Sigma p : I.\, t\, p = m\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\Sigma p : I.\, t\, p \leq m \wedge \neg e\}}\ \text{while-fun}'$$

Nonetheless, the first problem remains and cannot be solved for undecidable J.

Have we got other options instead of building a triangle of transformations along the lines we have discussed? In the forward transformation, we could try to embed the total-correctness Hoare logic with while-rel rather than while-fun. This must fail, since, in order to construct a strictly decreasing colist of values of t, we would need to construct minimal witnesses of $\exists m.\, J\, m$, which requires the kind of bounded search with a generally undecidable predicate that we just described.

Alternatively, we could try to reorganize the backward transformation, targeting the total-correctness Hoare logic with while-fun rather than while-rel. That must fail as well, since as the variant t would have to bound the lengths of all traces satisfying $\langle U \rangle^2 ** (P ** \langle I \rangle^2)^\dagger ** \langle \neg e \rangle$. Since there is no general way to enumerate such traces for general U, I, P and they could also be infinitely many, attempts to define t must fail.

Why are we getting these troubles? The reason is that we are working with general Hoare triples in the trace-based Hoare logic, not strongest postcondition triples and we strive for translating assertions purely syntactically. Going more semantic, we could get results of the sort that we really desire. Here is an outline of how.

Instead of working with the supremum of the length of traces satisfying the given postcondition of a given statement (the least upper bound on time-to-completion based on the given postcondition), we can be more semantical and work with the length of the actual trace produced by the statement from a given initial state (the actual time-to-completion, which is the same as the length of the unique trace satisfying the strongest postcondition). As soon as we assume a bound on this value (the bound may be correct or incorrect), it becomes finite and we can compute it by running the statement until the bound; if the bound turns out to be incorrect—the statement does not terminate before reaching the bounding length—it is fine to return 17. Hence the length of the actual trace produced by the loop can be used as the variant t in the while-fun rule.

5. Examples

Propositions 4.1 and 4.2 show that our trace-based logic is expressive enough to perform the same analyses that the state-based partial or total correctness Hoare logics can perform. However, the expressiveness of our logic goes beyond that of the partial and the total correctness Hoare logics. In this section, we demonstrate this by a series of examples. We adopt the usual notational convention that any occurrence of a variable in a state predicate represents the value of the variable in the state, e.g., a state predicate $x + y = 7$ abbreviates $\lambda\sigma.\, \sigma\, x + \sigma\, y = 7$.

5.1. **Unbounded total search.** Since we work in a constructive underlying logic, we can distinguish between termination of a run, *finite*, and nondivergence, $\neg$*infinite*. For instance, any unbounded nonpartial search fails to be terminating but is nonetheless nondivergent.

This example is inspired by Markov's principle: $\neg\forall n.\, \neg B\, n \to \exists x. B\, n$ for any decidable predicate B on natural numbers, i.e., a predicate satisfying $\forall n.\, B\, n \vee \neg B\, n$. Markov's

principle is a classical tautology, but is not valid constructively. This implies we cannot constructively prove a statement s that searches a natural number n satisfying B by successively checking whether $B\ 0, B\ 1, B\ 2, \ldots$ to be terminating. In other words, we cannot constructively derive a total correctness judgement for s. The assumption $\neg\forall n.\ \neg B\, n$ only guarantees that B is not false everywhere, therefore the search cannot diverge; indeed, we can constructively prove that s is nondivergent in our logic.

We assume given a decidable predicate B on natural numbers and an axiom *B_noncontradictory*: $\neg\forall n.\ \neg B\, n$ stating that B is not false everywhere. Therefore running the statement

$$\mathit{Search} \equiv x := 0; \mathsf{while}\ \neg B\, x\ \mathsf{do}\ x := x + 1$$

cannot diverge: this would contradict *B_noncontradictory*. In Proposition 5.1 we prove that any trace produced by running s is nondivergent and $B\ x$ holds of the last state.

We define a predicate $\mathit{cofinally} : \mathit{nat} \to \mathit{trace} \to \mathit{Prop}$ coinductively as follows:

$$\frac{\sigma\ x = n \quad B\ n}{\sigma :: \langle\sigma\rangle \models \mathit{cofinally}\ n} \qquad \frac{\sigma\ x = n \quad \neg B\ n \quad \tau \models \mathit{cofinally}\ (n+1)}{\sigma :: \sigma :: \tau \models \mathit{cofinally}\ n}$$

cofinally is a setoid predicate.

A crucial observation is that, in the presence of *B_noncontradictory*, *cofinally* 0 is stronger than nondivergent:

Lemma 5.1. *cofinally* $0 \models \neg$*infinite*.

Proof. It is sufficient to prove that, for any τ, $\tau \models$ *cofinally* 0 and $\tau \models$ *infinite* are contradictory. Suppose there is a trace τ such that $\tau \models$ *cofinally* 0 and $\tau \models$ *infinite*. Then by induction on n we can show that, for any n there is a trace τ' such that $\tau' \models$ *cofinally* n and $\tau' \models$ *infinite*. But whenever the latter condition holds for some τ' and n, then $\neg B\, n$. Hence we also have $\forall n.\ \neg B\, n$. But this contradicts *B_noncontradictory*. □

It is straightforward to prove, if τ satisfies *cofinally* 0 then its last state (if exists) satisfies B at x.

Lemma 5.2. *cofinally* $0 \models \mathsf{true} ** \langle B\ x\rangle$.

Proof. We prove the following condition by coinduction, from which the lemma follows: for any n, τ, if $\tau \models$ *cofinally* n then $\tau \models_\tau \langle B\ x\rangle$. □

Proposition 5.1. $\{\mathsf{true}\}$ *Search* $\{(\mathsf{true} ** \langle B\ x\rangle) \wedge \neg$*infinite*$\}$.

Proof. The derivation is given in Figure 5.

(1) We use Lemmata 5.1 and 5.2.

(2) x is incremented by one for every iteration until B holds at x. $\langle x = 0\rangle^2 ** ((\neg B\, x)[x \mapsto x+1] ** \langle\mathsf{true}\rangle^2)^\dagger ** \langle B\ x\rangle \models$ *cofinally* 0 follows from the definition of *cofinally*. (It is proved by coinduction.)

(3) We take true as the loop invariant.

□

$$\dfrac{\dfrac{\dfrac{\{\neg B\,x\}\ x := x+1\ \{(\neg B\,x)[x \mapsto x+1]\}}{\{x=0\}\ \mathsf{while}\ \neg B\,x\ \mathsf{do}\ x := x+1\ \{\langle x=0\rangle^2 ** ((\neg B\,x)[x \mapsto x+1] ** \langle \mathsf{true}\rangle^2)^\dagger ** \langle B\ x\rangle\}}\ (3)}{\{\mathsf{true}\}\ x := 0\ \{\mathsf{true}[x \mapsto 0]\}\quad \backslash\ \{x=0\}\ \mathsf{while}\ \neg B\,x\ \mathsf{do}\ x := x+1\ \{\mathit{cofinally}\ 0\}}\ (2)}{\dfrac{\{\mathsf{true}\}\ x := 0; \mathsf{while}\ \neg B\,x\ \mathsf{do}\ x := x+1\ \{\mathsf{true}[x \mapsto 0] ** \mathit{cofinally}\ 0\}}{\{\mathsf{true}\}\ x := 0; \mathsf{while}\ \neg B\,x\ \mathsf{do}\ x := x+1\ \{(\mathsf{true} ** \langle B\ x\rangle) \wedge \neg \mathit{infinite}\}}\ (1)}$$

Figure 5: Derivation of $\{\mathsf{true}\}$ *Search* $\{(\mathsf{true} ** \langle B\ x\rangle) \wedge \neg \mathit{infinite}\}$

5.2. **Liveness.** As the similarity of our assertion language to the interval temporal logic suggests, we can specify and prove liveness properties. In Proposition 5.2, we prove that the statement

$$x := 0; \mathsf{while}\ \mathsf{true}\ \mathsf{do}\ x := x+1$$

eventually sets the value of x to n for any $n : \mathit{nat}$ at some point.

The example is simple but sufficient to demonstrate core techniques used to prove liveness properties of more practical examples. For instance, imagine that assignment to x involves a system call, with the assigned value as the argument. It is straightforward to enrich traces to record such special events, and we can then apply the same proof technique to prove the statement eventually performs the system call with n as the argument for any n.

We define inductively a predicate $\mathit{eventually} : \mathit{nat} \to \mathit{trace} \to \mathit{Prop}$ stating a state σ in which the value of x is n is eventually reachable by finitely traversing τ:

$$\frac{\sigma\ x = n}{\langle\sigma\rangle \models \mathit{eventually}\ n} \qquad \frac{\sigma\ x = n}{\sigma :: \tau \models \mathit{eventually}\ n} \qquad \frac{\tau \models \mathit{eventually}\ n}{\sigma :: \tau \models \mathit{eventually}\ n}$$

Moreover, if τ satisfies *eventually* n, then it has a finite prefix followed by a state that maps x to n.

Lemma 5.3. For any n, τ, if $\tau \models \mathit{eventually}\ n$ then $\tau \models \mathit{finite} ** \langle x = n\rangle ** \mathsf{true}$.

Proof. By induction on the derivation of *eventually* n. □

Proposition 5.2. For any $n : \mathit{nat}$, $\{\mathsf{true}\}\ s\ \{\mathit{finite}\ ** \langle x = n\rangle ** \mathsf{true}\}$ where $s \equiv x := 0; \mathsf{while}\ \mathsf{true}\ \mathsf{do}\ x := x+1$.

Proof. The derivation is given in Figure 6.

(1) We use Lemma 5.3.

(2) x is incremented by one for every iteration, starting from zero. Each iteration is finite: the length of the trace of each iteration is invariably two. x must eventually become n after a finite number of iteration for any n.

(3) We take true as the invariant.

□

$$\dfrac{\dfrac{\dfrac{\{\mathsf{true}\}\ x := x+1\ \{\mathsf{true}[x \mapsto x+1]\}}{\{x=0\}\ \mathsf{while\ true\ do}\ x := x+1\ \{\langle x=0\rangle^2 \mathbin{**} (\mathsf{true}[x \mapsto x+1] \mathbin{**} \langle\mathsf{true}\rangle^2)^\dagger \mathbin{**} \langle\mathsf{false}\rangle\}}\ (3) \quad \backslash}{\{\mathsf{true}\}\ x := 0\ \{\mathsf{true}[x \mapsto 0]\} \quad \{x=0\}\ \mathsf{while\ true\ do}\ x := x+1\ \{\mathit{eventually}\ n\}}\ (2)}{\{\mathsf{true}\}\ x := 0; \mathsf{while\ true\ do}\ x := x+1\ \{\mathsf{true}[x \mapsto 0] \mathbin{**} \mathit{eventually}\ n\}}}{\{\mathsf{true}\}\ x := 0; \mathsf{while\ true\ do}\ x := x+1\ \{\mathit{finite}\ \mathbin{**} \langle x=n\rangle \mathbin{**} \mathsf{true}\}}\ (1)$$

Figure 6: Derivation of $\{\mathsf{true}\}\ s\ \{\mathit{finite}\ \mathbin{**} \langle x=n\rangle \mathbin{**} \mathsf{true}\}$

5.3. **Weak trace equivalence.** The last example is inspired by a notion of weak trace equivalence: two traces are weakly equivalent if they are bisimilar by identifying a finite number of consecutive identical states with a single state. It is conceivable that (strong) bisimilarity is too strong for some applications and one needs weak bisimilarity. For instance, we may want to prove that the observable behavior, such as the colist i/o events of a potentially diverging run, is bisimilar to a particular colist of i/o events. Then we must be able to collapse a finite number of non-observable internal steps. We definitely should not collapse an infinite number of internal steps, otherwise we would end up concluding that a statement performing an i/o operation after a diverging run, e.g., $\mathsf{while\ true\ do\ skip}; \mathsf{print}$ *"hello"*, is observably equivalent to a statement immediately performing the same i/o operation, e.g., print *"hello"*.

In this subsection, we prove that the trace produced by running the statement

$$\mathsf{while\ true\ do}\ (y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1)$$

is weakly bisimilar to the ascending sequence of natural numbers $0 :: 1 :: 2 :: 3 :: \ldots$, by projecting the value of x. The statement differs from that of the previous subsection in that it "stutters" for a finite but unbounded number of steps, i.e., $\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1$, before the next assignment to x happens.

This exercise is instructive in that we need to formalize weak trace equivalence in our constructive underlying logic. We do so by supplying an inductive predicate $\tau \overset{*}{\rightsquigarrow} \tau'$ stating that τ' is obtained from τ by dropping finitely many elements from the beginning, until the first state with a different value of x is encountered, and a coinductive predicate $up\ (n : nat) : trace \to Prop$, stating that τ is weakly bisimilar to the ascending sequence of natural numbers starting at n, by projecting the value of x. Formally:

$$\dfrac{\sigma\ x = hd\ \tau\ x \quad \tau \overset{*}{\rightsquigarrow} \tau'}{\sigma :: \tau \overset{*}{\rightsquigarrow} \tau'} \qquad \dfrac{\sigma\ x \neq hd\ \tau\ x \quad \tau \approx \tau'}{\sigma :: \tau \overset{*}{\rightsquigarrow} \tau'}$$

$$\dfrac{\sigma\ x = n \quad \sigma :: \tau \overset{*}{\rightsquigarrow} \tau' \quad \tau' \models up\ (n+1)}{\sigma :: \tau \models up\ n}$$

These definitions are tailored to our example. But a more general weak trace equivalence can be defined similarly [15]. We note that our formulation is not the only one possible nor the most elegant. In particular, with a logic supporting nesting of induction into coinduction as primitive [5], there is no need to separate the definition into an inductive part, $\tau \overset{*}{\rightsquigarrow} \tau'$, and a coinductive part, $up\ n$. Yet our formulation is amenable in our underlying logic, Coq.

$$\dfrac{\{y \neq 0\}\ y := y-1\ \{(y \neq 0)[y \mapsto y-1]\}}{\{y \geq 0\}\ \mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1\ \{\langle y \geq 0\rangle^2 ** ((y \neq 0)[y \mapsto y-1] ** \langle \mathsf{true}\rangle^2)^\dagger ** \langle y = 0\rangle\}}\ (4)$$

$$\dfrac{\{y \geq 0\}\ \mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1\ \{\langle y \geq 0\rangle^2 ** ((y \neq 0)[y \mapsto y-1] ** \langle \mathsf{true}\rangle^2)^\dagger ** \langle y = 0\rangle\}}{\{y \geq 0\}\ \mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1\ \{\langle x\rangle^*\}}\ (3)$$

$$\dfrac{\{y \geq 0\}\ \mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1\ \{\langle x\rangle^*\} \qquad \{\mathsf{true}\}\ x := x+1\ \{\mathsf{true}[x \mapsto x+1]\}}{\{y \geq 0\}\ (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1\ \{\langle x\rangle^* ** \mathsf{true}[x \mapsto x+1]\}}$$

$$\dfrac{\{x \geq 0\}\ y := x\ \{(x \geq 0)[y \mapsto x]\} \qquad \{y \geq 0\}\ (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1\ \{\langle x\rangle^* ** \mathsf{true}[x \mapsto x+1]\}}{\{x \geq 0\}\ y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1\ \{\langle x\rangle^* ** \mathsf{true}[x \mapsto x+1]\}}$$

$$\dfrac{\{x \geq 0\}\ y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1\ \{\langle x\rangle^* ** \mathsf{true}[x \mapsto x+1]\}}{\{x = 0\}\ \mathsf{while\ true\ do}\ (y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1)\ \{\langle x = 0\rangle^2 ** (\langle x\rangle^* ** \mathsf{true}[x \mapsto x+1] ** \langle \mathsf{true}\rangle^2)^\dagger ** \langle \mathsf{false}\rangle\}}\ (2)$$

$$\dfrac{\{x = 0\}\ \mathsf{while\ true\ do}\ (y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1)\ \{\langle x = 0\rangle^2 ** (\langle x\rangle^* ** \mathsf{true}[x \mapsto x+1] ** \langle \mathsf{true}\rangle^2)^\dagger ** \langle \mathsf{false}\rangle\}}{\{x = 0\}\ \mathsf{while\ true\ do}\ (y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1)\ \{up\ 0\}}\ (1)$$

Figure 7: Derivation of $\{\mathsf{true}\}\ s\ \{up\ 0\}$

We also use an auxiliary trace predicate $\langle x\rangle^*$ that is true of a finite trace in which the value of x does not change. It is defined inductively as follows:

$$\dfrac{}{\langle\sigma\rangle \models \langle x\rangle^*} \qquad \dfrac{\sigma\ x = hd\,\tau\,x \quad \tau \models \langle x\rangle^*}{\sigma :: \tau \models \langle x\rangle^*}$$

Proposition 5.3. $\{x = 0\}\ s\ \{up\ 0\}$ where $s \equiv \mathsf{while\ true\ do}\ (y := x; (\mathsf{while}\ y \neq 0\ \mathsf{do}\ y := y-1); x := x+1)$.

Proof. The derivation is given in Figure 7.

(1) x is incremented by one for every iteration, starting from zero, and the loop is iterated forever. Each iteration is finite, unbounded though. So x must increase forever.

(2) We take true as the invariant.

(3) y is decremented by one for every iteration, starting from a positive. It must become zero after a finite number of iteration. x does not change.

(4) We take true as the invariant.

□

6. Related work

Coinductive big-step semantics for nontermination have been considered by Leroy and Grall [10, 11] (in the context of the CompCert project, which is a major demonstration of feasibility of certified compilation) and Cousot and Cousot [4]. Leroy and Grall investigate two approaches. The first, based on Cousot and Cousot [3], has different evaluation relations for terminating and diverges runs, one inductive (with finite traces), the other coinductive (with infinite traces). To conclude that any program either terminates or diverges, one needs the law of excluded middle (amounting to decidability of the halting problem), and, as a result, the small-step semantics cannot be proved sound wrt. the big-step semantics constructively. The other approach [1] uses a coinductively defined evaluation relation with possibly infinite traces, where while-loops are not ensured to be progressive in terms of growing traces (an infinite number of consecutive silent small steps may be collapsed).

Some other works on coinductive big-step semantics include Glesner [6] and Nestra [16, 17]. In these it is accepted that a program evaluation can somehow continue after an infinite number of small steps. With Glesner, this seems to have been a curious unintended side-effect of the design, which she was experimenting with just for the interest of it. Nestra developed a nonstandard semantics with transfinite traces on purpose in order to obtain a soundness result for a widely used slicing transformation that is unsound standardly (can turn nonterminating runs into terminating runs).

Our trace-based coinductive big-step semantics [14] was heavily inspired by Capretta's [2] modelling of nontermination in a constructive setting similar to ours. Rather than using coinductive possibly infinite traces, he works with a coinductive notion of a possibly infinitely delayed (final) state. The categorical basis appears in Rutten's work [19]. But Rutten only studied the classical setting (any program terminates or not), where a delayed state collapses to a choice of between a state or a designated token signifying nontermination.

While Hoare logics for big-step semantics based on inductive, finite traces have been considered earlier (to reason about traces of terminating runs), Hoare or VDM-style logics for reasoning about properties of nonterminating runs seem not have been studied before, with one very interesting exception, see below. Neither do we in fact know about dynamic logic or KAT (Kleene algebra with tests) approaches that would have assertions about possibly infinite traces. Rather, nonterminating runs have been typically reasoned about in temporal logics like LTL and CTL* or in interval temporal logic [13, 7]. These are however essentially different in spirit by their "exogeneity": assertions are made about traces in a transition system rather than traces of runs of a particular program. Notably, however, interval temporal logic has connectives similar to ours—in fact they were a source of inspiration for our design.

Hofmann and Pavlova [9] consider a VDM-style logic with finite trace assertions that are applied to all finite prefixes of the trace of a possibly nonterminating run of a program. This logic allows reasoning about safety, but not liveness. We expect that we should be able to embed a logic like this in ours.

7. Conclusions

We have presented a sound and complete Hoare logic for the coinductive trace-based big-step semantics of While. The logic naturally extends both the partial and total correctness Hoare logics. Its design may be exploratory at this stage—in the sense that one might wish to consider alternative choices of primitive connectives. But at any rate we would see our logic as a viable unifying foundational framework facilitating translations from more applied logics.

Acknowledgements. We are grateful to Martin Hofmann, Thierry Coquand and Adam Chlipala for discussions.

We acknowledge the support of the EU FP6 IST integrated project no. 15905 MOBIUS, the Estonian Centre of Excellence in Computer Science, EXCS, financed by the European Regional Development Fund, and the Estonian Science Foundation grant no. 6940.

References

[1] Blazy, S., Leroy, X.: Mechanized semantics for the Clight subset of the C language. J. of Automated Reasoning 43(3) (2009) 263–288

[2] Capretta, V.: General recursion via coinductive types. Logical Methods in Computer Science 1(2) (2005) article 1

[3] Cousot, P., Cousot, R.: Inductive definitions, semantics and abstract interpretation. In Conf. Record of 19th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL '92 (Albuquerque, NM, Jan. 1992). ACM Press (1992) 83–94

[4] Cousot, P., Cousot, R.: Bi-inductive structural semantics. Inform. and Comput. 207(2) (2009) 258–283

[5] Danielsson, N. A., Altenkirch, T.: Mixing induction and coinduction. Draft (2009) Available at `http://www.cs.nott.ac.uk/~nad/publications/`

[6] Glesner, S.: A proof calculus for natural semantics based on greatest fixed point semantics. In Knoop, J., Necula, G. C., Zimmermann, W., eds.: Proc. of 3rd Int. Wksh. on Compiler Optimization Meets Compiler Verification, COCV 2004 (Barcelona, Apr. 2004), Electron. Notes in Theor. Comput. Sci. 132(1). Elsevier (2004) 73–93

[7] Halpern, J., Manna, Z., Moszkowski, B.: A hardware semantics based on temporal intervals. In Díaz, J., ed.: Proc. of 10th Int. Coll. on Automata, Languages and Programming, ICALP '83 (Barcelona, July 1983), Lect. Notes in Comput. Sci. 154. Springer (1983) 278–291

[8] Hasuo, I., Jacobs, B., Sokolova, A.: Generic trace semantics via coinduction. Logical Methods in Computer Science 3(4) (2007) article 11

[9] Hofmann, M., Pavlova, M.: Elimination of ghost variables in program logics. In Barthe, G., Fournet, C., eds.: Revised Selected Papers from 3rd Symp. on Trustworthy Global Computing, TGC 2007 (Sophia-Antipolis, Nov. 2007), Lect. Notes in Comput. Sci. 4912. Springer (2007) 1–20

[10] Leroy, X.: Coinductive big-step operational semantics. In Sestoft, P., ed.: Proc. of 15th Europ. Symp. on Programming, ESOP 2006 (Vienna, March, 2006), Lect. Notes in Comput. Sci. 3924. Springer (2006) 54–68

[11] Leroy, X., Grall, H.: Coinductive big-step operational semantics. Inform. and Comput. 207(2) (2009) 285–305.

[12] Kleymann, T.: Hoare logic and auxiliary variables. Formal Asp. Comput. 11(5) (1999) 541–566

[13] Moszkowski, B.: A temporal logic for reasoning about hardware. Computer 18(2) (1985) 10–19

[14] Nakata, K., Uustalu, T.: Trace-based coinductive operational semantics for While: big-step and small-step, relational and functional styles. In Nipkow, T., Urban, C., eds.: Proc. of 22nd Int. Conf. on Theorem Proving in Higher Order Logics, TPHOLs 2009 (Munich, Aug. 2009), Lect. Notes in Comput. Sci. 5674. Springer (2009) 375–390

[15] Nakata, K., Uustalu, T.: Resumptions, weak bisimilarity and big-step Semantics for While with interactive I/O: an exercise in mixed induction-coinduction. In Aceto, L., Sobocinski, P., eds.: Proc. of 7th Wksh. on Structural Operational Semantics, SOS 2010 (Paris, Aug. 2010), Electron. Proc. in Theor. Comput. Sci., to appear

[16] Nestra, H.: Fractional semantics. In Johnson, M., Vene, V., eds.: Proc. of 11th Int. Conf. on Algebraic Methodology and Software Technology, AMAST 2006 (Kuressaare, July 2006), Lect. Notes in Comput. Sci. 4019. Springer (2006) 278–292

[17] Nestra, H.: Transfinite semantics in the form of greatest fixpoint. J. of Logic and Algebr. Program. 78(7) (2009) 574–593

[18] Riis Nielson, H., Nielson, F.: Semantics with Applications: A Formal Introduction. Wiley (1992)

[19] Rutten, J.: A note on coinduction and weak bisimilarity for While programs. Theor. Inform. and Appl. 33(4–5) (1999) 393–400

Appendix A. State-based partial correctness and total correctness Hoare logics

The figures below give the rules of the standard, state-based partial correctness and total correctness logics in the form used in Section 4.

$$\frac{}{\{U[e/x]\}\ x := e\ \{U\}} \qquad \frac{}{\{U\}\ \mathsf{skip}\ \{U\}} \qquad \frac{\{U\}\ s_0\ \{V\} \quad \{V\}\ s_1\ \{Z\}}{\{U\}\ s_0; s_1\ \{Z\}}$$

$$\frac{\{e \wedge U\}\ s_t\ \{Z\} \quad \{\neg e \wedge U\}\ s_f\ \{Z\}}{\{U\}\ \mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f\ \{Z\}} \qquad \frac{\{e \wedge I\}\ s_t\ \{I\}}{\{I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{I \wedge \neg e\}}$$

$$\frac{U \models U' \quad \{U'\}\ s\ \{Z'\} \quad Z' \models Z}{\{U\}\ s\ \{Z\}}$$

Figure 8: Inference rules of partial-correctness Hoare logic

$$\frac{}{\{U[e/x]\}\ x := e\ \{U\}} \qquad \frac{}{\{U\}\ \mathsf{skip}\ \{U\}} \qquad \frac{\{U\}\ s_0\ \{V\} \quad \{V\}\ s_1\ \{Z\}}{\{U\}\ s_0; s_1\ \{Z\}}$$

$$\frac{\{e \wedge U\}\ s_t\ \{Z\} \quad \{\neg e \wedge U\}\ s_f\ \{Z\}}{\{U\}\ \mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f\ \{Z\}}$$

$$\frac{\forall n : nat\ \{e \wedge I \wedge t = n\}\ s_t\ \{I \wedge t < n\}}{\{I \wedge t = m\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{I \wedge t \leq m \wedge \neg e\}}\ \text{while-fun}$$

alt.

$$\frac{\forall n : nat\ \{e \wedge J\,n\}\ s_t\ \{\exists k.\, k < n \wedge J\,k\}}{\{J\,m\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\exists k.\, k \leq m \wedge J\,k \wedge \neg e\}}\ \text{while-rel}$$

$$\frac{U \models U' \quad \{U'\}\ s\ \{Z'\} \quad Z' \models Z}{\{U\}\ s\ \{Z\}}$$

Figure 9: Inference rules of total-correctness Hoare logic