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
{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 both the partial correctness Hoare logic and the total correctness Hoare logic: they are embeddable. 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.

A.D. Gordon (Ed.): ESOP 2010, LNCS 6012, pp. 488–506, 2010.
© Springer-Verlag Berlin Heidelberg 2010

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 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 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 standard, 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. 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.

However, the power of our logic goes beyond that of the state-based partial 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 and total correctness Hoare logics. 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`.

[1] We should really say "nonpartial".

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 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 $\mathit{hd}\ \tau$ of a trace τ by case distinction by $\mathit{hd}\ \langle \sigma \rangle = \sigma, \mathit{hd}\ (\sigma :: \tau) = \sigma$. 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

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

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, $\forall \tau.\, (\neg\, \exists \sigma.\, \tau \downarrow \sigma) \rightarrow \tau^{\uparrow}$ is constructively provable, but $\forall \tau.\, (\neg\, \tau^{\uparrow}) \rightarrow \exists \sigma.\, \tau \downarrow \sigma$ 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 $(s, \tau) \overset{*}{\Rightarrow} \tau'$, expressing that running a statement s from the last state (if it exists) of an already accumulated trace τ results in a total trace τ'. Extended evaluation is also defined coinductively, as the coinductive prefix closure of evaluation.

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''

[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).

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

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

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

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

Fig. 1. Big-step semantics

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 $(s_1, \tau) \overset{*}{\Rightarrow} \tau'$, 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).

Proposition 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). This is good for several reasons. First, skip becomes a unit of sequential composition. 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.

Third, we obtain that any while-loop always progresses. For instance, in our semantics we can only derive $(\mathsf{while}\ \mathsf{true}\ \mathsf{do}\ \mathsf{skip}, \sigma) \Rightarrow \sigma :: \sigma :: \sigma :: \ldots$ (up to bisimilarity). Giving up insisting on progress in terms of growing the trace would introduce some semantic anomalies. For instance, we would not like to have $(\mathsf{while}\ \mathsf{true}\ \mathsf{do}\ \mathsf{skip}, \sigma) \Rightarrow \langle \sigma \rangle$, because an intuitively clearly infinite run would then be recorded in a finite trace, with the consequence that $(\mathsf{while}\ \mathsf{true}\ \mathsf{do}\ \mathsf{skip}; x := 17, \sigma) \Rightarrow \sigma :: \langle \sigma[x \mapsto 17] \rangle$ etc. 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. *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.

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]

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

[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.

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

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

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

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

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

$$\begin{array}{c} \langle \sigma \rangle \models Q \\ \hline\hline \langle \sigma \rangle \models_{\langle \sigma \rangle} Q \end{array} \qquad \begin{array}{c} \sigma :: \tau \models Q \\ \hline\hline \sigma :: \tau \models_{\langle \sigma \rangle} Q \end{array} \qquad \begin{array}{c} \tau' \models_{\tau} Q \\ \hline\hline \sigma :: \tau' \models_{\sigma :: \tau} Q \end{array}$$

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

Fig. 2. Semantics of assertions

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. *For any U, $\langle U \rangle$, $U[x \mapsto e]$, $\langle U \rangle^2$ 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.*

A number of logical consequences and equivalences hold about these connectives. 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: *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. If every trace satisfying P is infinite, i.e., if $P \models \mathit{infinite}$, then $\mathit{Last}\ P \Leftrightarrow \mathsf{false}$.

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 $P \models \mathit{infinite}$, 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 $P \models \mathit{infinite}$.

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

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

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

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

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

Fig. 3. Inference rules of Hoare logic

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

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 4 (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.
- $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$: 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., [17]) 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 $sp(s, U)$ is a postcondition according to the logic (i.e., $\{U\}\ s\ \{sp(s, U)\}$ 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 2. *For any s, U, τ, τ', if $\tau \models sp(s, U)$ and $\tau \approx \tau'$ then $\tau' \models sp(s, U)$.*

Lemma 3. *For any s, U, U', if $U \models U'$ then $sp(s, U) \models sp(s, U')$.*

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

Lemma 4. *For any s, U, τ, if $\tau \models sp(s, U)$ then $hd\ \tau \models U$.*

The following lemma is central for the next two important lemmata, stating that *Last* P and $Inv(e, s, U)$ are adequate.

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

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

Fig. 4. Strongest postcondition

Lemma 5. *For any τ, 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 6. *For any P, $P \Leftrightarrow P \mathbin{**} \langle \mathit{Last}\ P\rangle$.*

Proof. Suppose we are given $\tau \models P$. By the definition of $\mathit{Last}\ P$, we have for any σ, $\tau \downarrow \sigma$ implies $\sigma \models \mathit{Last}\ P$. We then deduce $\tau \models_\tau \langle \mathit{Last}\ P\rangle$ by Lemma 5, thus conclude $\tau \models P \mathbin{**} \langle \mathit{Last}\ P\rangle$.

Suppose we are given $\tau_0 \models P$ and $\tau_1 \models_{\tau_0} \langle \mathit{Last}\ P\rangle$. We prove the following condition by coinduction: for any U, τ, τ', $\tau' \models_\tau \langle U\rangle$ implies $\tau \approx \tau'$. Therefore we have $\tau_0 \approx \tau_1$, from which $\tau_1 \models P$ follows. (Recall that P is a setoid predicate.)

Lemma 7. *For any s, e, U, τ, $\mathit{sp}(s, \mathit{Inv}(e,s,U) \wedge e) \Leftrightarrow \mathit{sp}(s, \mathit{Inv}(e,s,U) \wedge e) \mathbin{**} \langle \mathit{Inv}(e,s,U)\rangle$.*

Proof. Suppose we are given $\tau \models \mathit{sp}(s, \mathit{Inv}(e,s,U) \wedge e)$. It suffices to prove $\tau \models_\tau \langle \mathit{Inv}(e,s,U)\rangle$. However, we have $\tau \models \langle \mathit{Inv}(e,s,U) \wedge e\rangle \mathbin{**} \mathit{sp}(s, \mathit{Inv}(e,s,U))$ by Lemma 3 and Lemma 4. By the definition of Inv, we have for any σ, $\tau \downarrow \sigma$ implies $\sigma \models \mathit{Inv}(e,s,U)$. Therefore we conclude $\tau \models_\tau \langle \mathit{Inv}(e,s,U)\rangle$ by Lemma 5. $\mathit{sp}(s, \mathit{Inv}(e,s,U) \wedge e) \mathbin{**} \langle \mathit{Inv}(e,s,U)\rangle \models \mathit{sp}(s, \mathit{Inv}(e,s,U) \wedge e)$ is proved similarly to Lemma 6.

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

Lemma 8. *For any s, U, $\{U\}\ s\ \{\mathit{sp}(s,U)\}$.*

Proof. By induction on s. We show the main case of while: $s = \mathsf{while}\ e\ \mathsf{do}\ s_t$. We are given as induction hypothesis that, for any U_0, $\{U_0\}\ s_t\ \{\mathit{sp}(s_t, U_0)\}$. We have to prove $\{U\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{\langle U\rangle^2 \mathbin{**} (\mathit{sp}(s_t, e \wedge I) \mathbin{**} \langle I\rangle^2)^\dagger \mathbin{**} \langle \neg e\rangle\}$ where $I = \mathit{Inv}(e, s_t, U)$. It is sufficient to prove $\{e \wedge I\}\ s_t\ \{(\mathit{sp}(s_t, e \wedge I) \mathbin{**} \langle I\rangle)\}$, which follows from the induction hypothesis and Lemma 7.

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 4 and the following lemma, stating that any trace which satisfies $sp(s, U)$ is in fact produced by a run of s.

Lemma 9. *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 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 5 (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 1, we have that $sp(s, U) \models P$. By Lemma 8, we have $\{U\}\ s\ \{sp(s, U)\}$. Applying consequence, we get $\{U\}\ s\ \{P\}$.

4 Relation to the Standard Partial 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 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.

We formalize our claim in the next two propositions, whose direct proofs are algorithms for the transformations. Proposition 6 states that, if $\{U\}\ s\ \{V\}$ is a derivable partial correctness formula, then $\{U\}\ s\ \{\mathsf{true} ** \langle V\rangle\}$ is derivable in our logic. The trace predicate $\mathsf{true} ** \langle V\rangle$ indicates that V 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 V 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 7 states that, if $\{U\}\ s\ \{V\}$ is a derivable total correctness judgement, then $\{U\}\ s\ \{\mathit{finite} ** \langle V\rangle\}$ is derivable in our logic. The trace predicate $\mathit{finite} ** \langle V\rangle$ states that the trace τ produced by running s is finite and V holds of the last state of τ; the finiteness of τ guarantees the existence of the last state.

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

Proposition 6. *For any U, s and V, if $\{U\}\ s\ \{V\}$ is derivable in the partial correctness Hoare logic, then $\{U\}\ s\ \{\mathsf{true} ** \langle V\rangle\}$.*

Proof. By induction on the Hoare logic derivation of $\{U\}\ s\ \{V\}$. We show the main case of while: $s \equiv \mathsf{while}\ e\ \mathsf{do}\ s_t$. We are given as the induction hypothesis $\{e \wedge I\}\ s\ \{\mathsf{true} ** \langle I\rangle\}$. We close the case by the derivation

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

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

Proposition 7. *For any U, s and V, if $\{U\}\ s\ \{V\}$ is derivable in the total correctness Hoare logic, then for any U_0, $\{U \wedge U_0\}\ s\ \{\langle U_0\rangle ** \mathit{finite} ** \langle V\rangle\}$.*

Proof. By induction on the Hoare logic derivation of $\{U\}\ s\ \{V\}$. We show the main case for while: $s \equiv \mathsf{while}\ e\ \mathsf{do}\ s_t$. We are given as the induction hypothesis that for all $n : \mathit{nat}$ and U_0, $\{e \wedge I \wedge t = n \wedge U_0\}\ s_t\ \{\langle U_0\rangle ** \mathit{finite} ** \langle I \wedge t < n\rangle\}$, Therefore we close the case by the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall n.\{e \wedge I \wedge t = n\}\ s_t\ \{\langle t = n\rangle ** \mathit{finite} ** \langle I \wedge t < n\rangle\}}{\{\exists n.e \wedge I \wedge t = n\}\ s_t\ \{\exists n.\langle t = n\rangle ** \mathit{finite} ** \langle I \wedge t < n\rangle\}}}{\{e \wedge I\}\ s_t\ \{(\exists n.\langle t = n\rangle ** \mathit{finite} ** \langle t < n\rangle) ** \langle I\rangle\}}}{\begin{array}{l}\{I \wedge U_0\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\\ \quad \{\langle I \wedge U_0\rangle^2 ** ((\exists n.\langle t = n\rangle ** \mathit{finite} ** \langle t < n\rangle) ** \langle I\rangle^2)^\dagger ** \langle \neg e\rangle\}\end{array}}}{\{I \wedge U_0\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{U_0 ** \mathit{finite} ** \langle I \wedge \neg e\rangle\}}$$

5 Examples

Propositions 6 and 7 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, ¬*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 8 we prove that any trace produced by running s is nondivergent and $B\ x$ holds of the last state.

We define a predicate *cofinally* : *nat* → *trace* → *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 10. *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*.

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

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

$$\dfrac{\dfrac{\{\mathsf{true}\}\ x := 0\ \{\mathsf{true}[x \mapsto 0]\} \qquad \dfrac{\dfrac{\{\mathsf{true}\}\ x := x+1\ \{\mathsf{true}[x \mapsto x+1]\}}{\begin{array}{l}\{x = 0\}\ \mathsf{while\ true\ do}\ x := x+1\\ \quad \{\langle x = 0\rangle^2 ** (\mathsf{true}[x \mapsto x+1] ** \langle \mathsf{true}\rangle^2)^\dagger ** \langle \mathsf{false}\rangle\}\end{array}} \setminus}{\{x = 0\}\ \mathsf{while\ true\ do}\ x := x+1\ \{\mathit{eventually}\ n\}}}{\{\mathsf{true}\}\ x := 0; \mathsf{while\ true\ do}\ x := x+1\ \{\mathsf{true}[x \mapsto 0] ** \mathit{eventually}\ n\}}}{\{\mathsf{true}\}\ x := 0; \mathsf{while\ true\ do}\ x := x+1\ \{\mathit{finite}\ \ ** \langle x = n\rangle ** \mathsf{true}\}}$$

Fig. 6. Derivation of $\{\mathsf{true}\}\ s\ \{\mathit{finite}\ \ ** \langle x = n\rangle ** \mathsf{true}\}$

Proposition 8. $\{\mathsf{true}\}\ \mathit{Search}\ \{(\mathsf{true} ** \langle B\ x\rangle) \wedge \neg \mathit{infinite}\}$.

Proof. The derivation is given in Figure 5, with trivial applications of the consequence rule being omitted.

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 9, we prove that the statement

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

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:

$$\dfrac{\sigma\ x = n}{\langle \sigma \rangle \models \mathit{eventually}\ n} \quad \dfrac{\sigma\ x = n}{\sigma :: \tau \models \mathit{eventually}\ n} \quad \dfrac{\tau \models \mathit{eventually}\ n}{\sigma :: \tau \models \mathit{eventually}\ n}$$

Proposition 9. *For any $n : nat$,* $\{\mathsf{true}\}\ s\ \{\mathit{finite}\ \ {*}{*}\ \langle x = n\rangle\ {*}{*}\ \mathsf{true}\}$ *where* $s \equiv x := 0; \mathsf{while\ true\ do}\ x := x + 1$.

Proof. The derivation is given in Figure 6, with trivial applications of the consequence rule being omitted.

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 $\mathit{up}\ (n : \mathit{nat}) : \mathit{trace} \to \mathit{Prop}$, stating that τ is weakly bisimilar to the ascending sequence of natural numbers starting at n, by projecting the value of x. Formally:

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

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

These definitions are tailored to our example. But a more general weak trace equivalence can be defined similarly. We note that our formulation is not the only one possible nor the most elegant. In particular, with a logic permitting mixing induction and coinduction [5], there is no need to separate the definition

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

Fig. 7. Derivation of $\{\mathsf{true}\}\ s\ \{up\ 0\}$

into an inductive part, $\tau \overset{*}{\leadsto} \tau'$, and a coinductive part, $up\ n$. Yet our formulation is amenable in our underlying logic, Coq.

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

Proof. The derivation is given in Figure 7, with trivial applications of the consequence rule being omitted.

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 [15,16]. 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 [18]. 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), 263–288 (2009)
2. Capretta, V.: General recursion via coinductive types. Logical Methods in Computer Science 1(2), article 1 (2005)
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 1992, Albuquerque, NM, January 1992, pp. 83–94. ACM Press, New York (1992)
4. Cousot, P., Cousot, R.: Bi-inductive structural semantics. Inform. and Comput. 207(2), 258–283 (2009)
5. Danielsson, N.A., Altenkirch, T.: Mixing induction and coinduction. Draft (2009), `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, April 2004. Electron. Notes in Theor. Comput. Sci. vol. 132(1), pp. 73–93. Elsevier, Amsterdam (2004)
7. Halpern, J., Manna, Z., Moszkowski, B.: A hardware semantics based on temporal intervals. In: Díaz, J. (ed.) ICALP 1983. LNCS, vol. 154, pp. 278–291. Springer, Heidelberg (1983)
8. Hasuo, I., Jacobs, B., Sokolova, A.: Generic trace semantics via coinduction. Logical Methods in Computer Science 3(4), article 11 (2007)
9. Hofmann, M., Pavlova, M.: Elimination of ghost variables in program logics. In: Barthe, G., Fournet, C. (eds.) TGC 2007. LNCS, vol. 4912, pp. 1–20. Springer, Heidelberg (2008)
10. Leroy, X.: Coinductive big-step operational semantics. In: Sestoft, P. (ed.) ESOP 2006. LNCS, vol. 3924, pp. 54–68. Springer, Heidelberg (2006)
11. Leroy, X., Grall, H.: Coinductive big-step operational semantics. Inform. and Comput. 207(2), 285–305 (2009)
12. Kleymann, T.: Hoare logic and auxiliary variables. Formal Asp. Comput. 11(5), 541–566 (1999)
13. Moszkowski, B.: A temporal logic for reasoning about hardware. Computer 18(2), 10–19 (1985)
14. Nakata, K., Uustalu, T.: Trace-based coinductive operational semantics for While: big-step and small-step, relational and functional styles. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 375–390. Springer, Heidelberg (2009)
15. Nestra, H.: Fractional semantics. In: Johnson, M., Vene, V. (eds.) AMAST 2006. LNCS, vol. 4019, pp. 278–292. Springer, Heidelberg (2006)
16. Nestra, H.: Transfinite semantics in the form of greatest fixpoint. J. of Logic and Algebr. Program. 78(7), 574–593 (2009)
17. Riis Nielson, H., Nielson, F.: Semantics with Applications: A Formal Introduction. Wiley, Chichester (1992)
18. Rutten, J.: A note on coinduction and weak bisimilarity for While programs. Theor. Inform. and Appl. 33(4-5), 393–400 (1999)

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.

$$\frac{}{\{U\}\ \mathsf{skip}\ \{U\}} \qquad \frac{}{\{U[e/x]\}\ x := e\ \{U\}} \qquad \frac{\{e \wedge U\}\ s_t\ \{V\} \quad \{\neg e \wedge U\}\ s_f\ \{V\}}{\{U\}\ \mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f\ \{V\}}$$

$$\frac{\{e \wedge I\}\ s_t\ \{I\}}{\{I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{I \wedge \neg e\}} \qquad \frac{U \models U' \quad \{U'\}\ s\ \{V'\} \quad V' \models V}{\{U\}\ s\ \{V\}}$$

Fig. 8. Inference rules of partial correctness Hoare logic

$$\frac{}{\{U\}\ \mathsf{skip}\ \{U\}} \qquad \frac{}{\{U[e/x]\}\ x := e\ \{U\}} \qquad \frac{\{e \wedge U\}\ s_t\ \{V\} \quad \{\neg e \wedge U\}\ s_f\ \{V\}}{\{U\}\ \mathsf{if}\ e\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f\ \{V\}}$$

$$\frac{\forall n : nat\ \{e \wedge I \wedge t = n\}\ s_t\ \{I \wedge t < n\}}{\{I\}\ \mathsf{while}\ e\ \mathsf{do}\ s_t\ \{I \wedge \neg e\}} \qquad \frac{U \models U' \quad \{U'\}\ s\ \{V'\} \quad V' \models V}{\{U\}\ s\ \{V\}}$$

Fig. 9. Inference rules of total correctness Hoare logic