Operational versus Weakest Precondition Semantics for the Probabilistic Guarded Command Language

Friedrich Gretz
RWTH Aachen University
Aachen, Germany
Email: fgretz@cs.rwth-aachen.de

Joost-Pieter Katoen
RWTH Aachen University
Aachen, Germany
Email: katoen@cs.rwth-aachen.de

Annabelle McIver
Macquarie University
Sydney, Australia
Email: annabelle.mciver@mq.edu.au

***Abstract*—This paper proposes a simple operational semantics of pGCL, Dijkstra's guarded command language extended with probabilistic choice, and relates this to pGCL's *wp*-semantics by McIver and Morgan. Parameterised Markov decision processes whose state rewards depend on the post-expectation at hand are used as operational model. We show that the *weakest pre-expectation* of a pGCL-program w.r.t. a post-expectation corresponds to the *expected cumulative reward* to reach a terminal state in the parameterised MDP associated to the program. In a similar way, we show a correspondence between weakest *liberal* pre-expectations and *liberal* expected cumulative rewards.**

I. Introduction

Formal semantics of programming languages has been the subject of intense research in computer science for several decades. Several approaches have been developed for the description of program semantics. Structured operational semantics defines the meaning of a program by means of an abstract machine where states correspond to program configurations (typically consisting of a program counter and a variable valuation) and transitions model the evolution of a program by executing statements. Program executions are then the possible runs of the abstract machine. Denotational semantics maps a program onto a mathematical object that describes for instance its input-output behaviour. Finally, axiomatic semantics provides the program semantics in an indirect manner by describing its properties. A prominent example of the latter are Hoare triples in which annotations, written in predicate logic, are associated to control points of the program.

The semantics of Dijkstra's seminal guarded command language [2] from the seventies is given in terms of weakest preconditions. It is in fact a predicate transformer semantics that is a total function between two predicates on the state of a program. The predicate transformer $E = wp(P, F)$ for program P and postcondition F yields the weakest precondition E on the initial state of P ensuring that the execution of P terminates in a final state satisfying F. There is a direct relation with axiomatic semantics: the Hoare triple $\{E\}P\{F\}$ holds for total correctness if and ony if $E \Rightarrow wp(P, F)$. The weakest *liberal* precondition $wlp(P, F)$ yields the weakest precondition for which P either does not terminate or establishes F. It does not ensure termination and corresponds to Hoare logic in partial correctness. Although providing an operational semantics for the guarded command language is rather straightforward, it lasted until the early nineties until Lukkien [8], [9] provided a formal connection between the predicate transformer semantics and the notion of a computation.

Qualitative annotations in predicate calculus are often insufficient for probabilistic programs as they cannot express quantities such as expectations over program variables. To that end, McIver and Morgan [10] generalised the methods of Dijkstra and Hoare to probabilistic programs by making the annotations real-valued expressions —referred to as expectations— in the program variables. Expectations are the quantitative analogue of predicates. This yields an expectation transformer semantics of the probabilistic guarded command language (pGCL, for short), an extension of Dijkstra's language with a probabilistic choice operator. An expectation transformer is a total function between two expectations on the state of a program. The expectation transformer $wp(P, f)$ for pGCL-program P and post-expectation f over final states yields the least expected value e on P's initial state ensuring that P's execution terminates with a value f. The annotation $\{e\}P\{f\}$ holds for total correctness if and only if $e \leq wp(P, f)$, where $\leq$ is to be interpreted in a point-wise manner. The weakest *liberal* pre-expectation $wlp(P, f)$ yields the least expectation for which P either does not terminate or establishes f. It does not ensure termination and corresponds to partial correctness.

This paper provides a simple operational semantics of pGCL using parametric Markov decision processes (pMDPs), a slight variant of MDPs in which probabilities may be parameterised [3]. Our main contribution in this paper is a formal connection between the *wp*- and *wlp*-semantics of pGCL by McIver and Morgan and the operational semantics. This provides a clean and insightful relationship between the abstract expectation transformer semantics that has been proven useful for formal reasoning about probabilistic programs, and the notion of a computation in terms of the operational model, a pMDP. In order to establish this connection we equip pMDPs with state rewards that depend on the post-expectation at hand. Intuitively speaking, we decorate a terminal state in the operational model of a program with a reward that corresponds to the value of the post-expectation. All other states are assigned reward zero. We then show that the *weakest pre-*

This research has been funded by the DFG Research Training Group 1298 (AlgoSyn), the EU FP7-Project CARP (Correct and Efficient Accelerator Programming), and the Australian Research Council DP1092464.

expectation of a pGCL-program P w.r.t. a post-expectation corresponds to the *expected cumulative reward* to reach a terminal state in the pMDP associated to P. In a similar way, we show that weakest *liberal* pre-expectations correspond to *liberal* expected cumulative rewards. The proofs are by induction on the structure of our probabilistic programs. This paper thus yields a computational view on the expectation transformer semantics of probabilistic programs using first principles of Markov decision processes.

A. Structure of this paper.

The rest of the paper is divided as follows. In Sect. II we introduce the probabilistic programming language pGCL. Parametric Markov decision processes with rewards are introduced in Sect. III. Section IV recaps the denotational semantics of pGCL [10] and introduces operational semantics for this language. Then the main result is established, namely that the two semantics are equivalent. Finally, Sect. V provides an example of reasoning over pGCL programs.

II. Probabilistic Programs

Our input language pGCL [10] is an extension of Dijkstra's guarded command language [2]. Besides a non-deterministic choice operator, denoted $[]$, and a conditional choice, it incorporates a probabilistic choice operator, denoted $[p]$, where p is a real parameter (or constant) whose values lies in the range $[0, 1]$. pGCL is a language to model sequential programs containing randomized assignments. For instance, the assignment $(x := 2 \cdot x \; [0.75] \; x := x{+}1)$ doubles the value of x with probability $\frac{3}{4}$ and increments it by one with the remaining probability $\frac{1}{4}$.

Definition 1. (Syntax of pGCL) Let P, P_1, P_2 be pGCL-programs, p a probability variable, x a program variable, E an expression, and G a Boolean expression. The syntax of a pGCL program P adheres to the following grammar:

$$\mathsf{skip} \mid \mathsf{abort} \mid x := E \mid P_1; P_2 \mid P_1 \,[]\, P_2 \mid P_1 \,[p]\, P_2 \mid$$
$$\mathsf{if}(G)\{P_1\}\,\mathsf{else}\,\{P_2\} \mid \mathsf{while}(G)\{P\}.$$

■

skip stands for the empty statement, abort for abortion, and $x := E$ for an assignment of the value of expression E (over the program variables) to variable x. The sequentially composed program $P_1; P_2$ behaves like P_1 and subsequently like P_2 on the successful termination of P_1. The statement $P_1 \,[]\, P_2$ denotes a non-deterministic choice; it behaves like either P_1 or P_2. The statement $P_1 \,[p]\, P_2$ denotes a probabilistic choice. It behaves like P_1 with probability p and like P_2 with probability $1{-}p$. The remaining two statements are standard: conditional choice and while-loop. Throughout this paper, we assume that pGCL-programs are well-typed. This entails that for assignments of the form $x := E$ we assume that x and E are of the same type. In a similar way, we assume G to denote a Boolean expression and variable p to denote a probability in the real interval $[0, 1]$.

Listing 1. The duelling cowboys, cf. [10].

```
1 int cowboyDuel(a, b) {
      // 0 < a, b < 1
2   (t := A [] t := B);
       // decide who starts
3   c := 1;
4   while (c = 1) {
5     if (t = A) {
6       (c := 0 [a] t := B);
7     } else {
8       (c := 0 [b] t := A);
9     }
10  }
11  return t; // the survivor
12 }
```

Example 2. (Duelling cowboys [10]) The pGCL program in Lst. 1 models the following situation: There are two cowboys, A and B, who are fighting a classical duel. They take turns, shooting at each other until one of them is hit. If A (resp. B) shoots then he hits B (resp. A) with probability a (resp. b). We assume that either cowboy A or B is allowed to start; the choice of who will start is resolved nondeterministically. Variable t keeps track of the turns, while c determines whether the duel continues or someone is hit. Note that it is a distinctive feature that we do not have to specify exact probabilities and instead allow arbitrary parameters. ■

III. Markov decision processes

This section introduces the basics of MDPs enriched with state rewards. We first recall the definition of an MDP with a countable state space and define elementary notions such as paths and policies. Subsequently, we introduce reward-MDPs in which states are equipped with an integer reward and focus on reachability objectives, in particular (liberal) expected cumulative rewards to reach a set of states. These measures are later shown to closely correspond to weakest pre-condition semantics of pGCL-programs.

A. Preliminaries

Let X be a finite set of real-valued variables, and $V(X)$ denote the set of expressions over X.

Definition 3. (Parametric distribution) A parametric distribution μ is a function that maps states to probabilities. The probabilities are real values in $[0, 1]$ or expressions over X:

$$\mu : S \to V(X) \quad \text{with} \quad \sum_{s \in S} \mu(s) = 1.$$

■

Example 4. (Parametric distribution) Consider $S = \{s_0, s_1, s_2\}$. Then a parametric distribution μ might be: $\mu(s_0) = p$, $\mu(s_1) = 1 - p$ and $\mu(s_2) = 0$ where $p \in [0, 1]$. Just note that p is a symbol and not an explicit number like 0.4. ■

Definition 5. (Markov decision process) An MDP $\mathcal{M}$ is a tuple $(S, S_0, \longrightarrow)$ where S is a countable set of states with

initial state-set $S_0 \subseteq S$ where $S_0 \neq \emptyset$, and $\longrightarrow \subseteq S \times \mathit{Dist}(S)$ is a transition relation from a state to a set of distributions over states. ■

Let $s \to \mu$ denote $(s, \mu) \in \longrightarrow$ and $s \to t$ denote $s \to \mu$ with $\mu(t) = 1$. We define $\mathit{Dist}(s) = \{\, \mu \mid s \to \mu \,\}$ to be the set of enabled distributions in state s. The intuitive operational behavior of an MDP $\mathcal{M}$ is as follows. First, non-deterministically select some initial state $s_0 \in S_0$. In state s with $\mathit{Dist}(s) \neq \emptyset$, non-deterministically select $\mu \in \mathit{Dist}(s)$. The next state t is randomly chosen with probability $\mu(t)$. If $\mathit{Dist}(t) = \emptyset$, exit; otherwise continue as for state s.

Our MDPs are called parametric because the underlying distributions are parametric.

Remark 6. (Finite support) In the context of this paper we are only interested in finitely branching Markov decision processes. This means that every state has finitely many successor states. Therefore $|Dist(s)| < \infty$ for all $s \in S$ and all distributions are assumed to have finite support. ■

A path of MDP $\mathcal{M}$ is a maximal alternating sequence $\pi = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \ldots$ such that $\mu_i(s_{i+1}) > 0$ for all $i \geq 0$. As any path is a maximal sequence, it is either infinite or ends in state s with $\mathit{Dist}(s) = \emptyset$. Reasoning about probabilities on sets of paths of an MDP relies on the resolution of non-determinism. This resolution is performed by a policy[1] that selects one of the enabled distributions in a state. Whereas in general a policy may base its decision in state s on the path fragment from $s_0 \in S_0$ to s, it suffices in the context of this paper to consider positional policies.

Definition 7. (Positional policy) Function $\mathfrak{P} : S \to \mathit{Dist}(S)$ is a positional policy for MDP $\mathcal{M} = (S, S_0, \longrightarrow)$ with $\mathfrak{P}(s) \in \mathit{Dist}(s)$ for all $s \in S$. ■

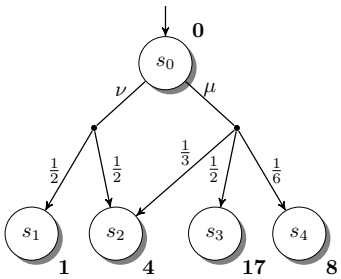
A positional policy thus selects an enabled distribution based on the current state s only. As in the rest of this paper, we only consider positional policies, we call them simply policies. The path fragment leading to s does not play any role. The path $\pi = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \ldots$ is called a $\mathfrak{P}$-path if it is induced by the policy $\mathfrak{P}$, that is, $\mathfrak{P}(s_i) = \mu_i$ for all $i \geq 0$. Let $\mathit{Paths}^{\mathfrak{P}}(s)$ denote the set of $\mathfrak{P}$-paths starting from state s. A policy of an MDP $\mathcal{M}$ induces a Markov chain $\mathcal{M}^{\mathfrak{P}}$ with the same state space as $\mathcal{M}$ and transition probabilities $\mathfrak{P}(s)(t)$ for states s and t. For finite path fragment $\widehat{\pi} = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \ldots \xrightarrow{\mu_{k-1}} s_k$ of a $\mathfrak{P}$-path, let $\mathbf{P}(\widehat{\pi})$ denote the probability of $\widehat{\pi}$ which is defined by $\mu_0(s_1) \times \ldots \times \mu_{k-1}(s_k) = \prod_{i=1}^{k} \mu_{i-1}(s_i)$. Let $\mathit{Pr}^{\mathfrak{P}}(\Pi)$ denote the probability of the set of paths Π under policy $\mathfrak{P}$. This probability measure is defined in the standard way using a cylinder set construction on the induced Markov chain $\mathcal{M}^{\mathfrak{P}}$ [1].

To compare our operational semantics of pGCL with its *wp*- and *wlp*-semantics, we use rewards (or, dually costs).

Definition 8. (MDP with rewards) An MDP with *rewards* (also called reward-MDP, or shortly RMDP) is a pair $(\mathcal{M}, r)$ with $\mathcal{M}$ an MDP with state space S and $r : S \to \mathbb{N}$ a function assigning a natural reward to each state. ■

Intuitively, the reward $r(s)$ stands for the reward earned on entering state s. The cumulative reward of a finite path fragment $s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \ldots s_k$ is the sum of the rewards in all states that have been visited, i.e., $r(s_0) + \ldots + r(s_k)$ provided $k > 0$, and 0 otherwise.

Example 9. (RMDP, cumulative reward of a path)



Assume a policy $\mathfrak{P}$ with $\mathfrak{P}(s_0) = \mu$. Then $\pi = s_0 \xrightarrow{\mu} s_3$ is a possible path that is taken with probability 0.5 and has cumulative reward $r(\pi) = 17$. ■

B. Reachability objectives

We are interested in reachability events in reward-MDPs. Let $T \subseteq S$ be a set of target states. The event $\Diamond T$ stands for the reachability of some state in T, i.e., $\Diamond T$ is the set of paths in MDP $\mathcal{M}$ that hit some state $s \in T$. Formally $\Diamond T = \{\, \pi \in \mathit{Paths} \mid \exists i \geq 0. \pi[i] \in T \,\}$ where $\pi[i]$ denotes the i-th state visited allong π. We write $\pi \models \Diamond T$ whenever π belongs to $\Diamond T$. It follows by standard arguments that $\Diamond T$ is a measurable event. The cumulative cost for this event is defined as follows.

Definition 10. (Cumulative cost for reachability) Let $\pi = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \ldots$ be a maximal path in reward-MDP $(\mathcal{M}, r)$ and $T \subseteq S$ a set of target states. If $\pi \models \Diamond T$, the *cumulative cost* along π before reaching T is defined by: $r_T(\pi) = r(s_0) + \ldots + r(s_k)$ where $s_i \notin T$ for all $i < k$ and $s_k \in T$. If $\pi \not\models \Diamond T$, then $r_T(\pi) = 0$. ■

Stated in words, the cumulative costs for a path π to reach T is the cumulative cost of the minimal prefix of π satisfying $\Diamond T$. In case π never reaches a state in T, the cumulative cost is defined to be zero. We denote by $\mathit{Paths}(s, \Diamond T)$ the set of paths starting in s that eventually reach T.

Definition 11. (Expected reward for reachability) Let $(\mathcal{M}, r)$ be an RMDP with state space S and $T \subseteq S$ and $s \in S$. The *minimal expected reward* until reaching $T \subseteq S$ from $s \in S$, denoted $\mathit{ExpRew}^{(\mathcal{M},r)}(s \models \Diamond T)$, is defined by:

$$\min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\{\, \pi \in \mathit{Paths}^{\mathfrak{P}}(s, \Diamond T) \mid r_T(\pi) = c \,\} \ .$$

The minimal *liberal* expected reward until reaching T from s,

[1] Also called scheduler, strategy or adversary.

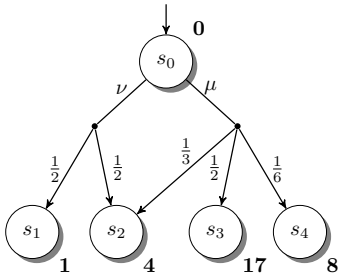
denoted $\mathit{LExpRew}^{(\mathcal{M},r)}(s \models \Diamond T)$, is defined by:

$$\min_{\mathfrak{P}} \left\{ \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\{\, \pi \in \mathit{Paths}^{\mathfrak{P}}(s, \Diamond T) \mid r_T(\pi) = c \,\} + \mathit{Pr}^{\mathfrak{P}}(s \not\models \Diamond T) \right\} .$$

We leave away the superscript when the underlying model is clear from context. ∎

The expected reward in s to reach some state in T is the expected cumulative cost over all paths (reaching T) induced under a demonic policy. The motivation to consider a demonic and not an angelic policy becomes clear further on in this paper, and has a direct relation with the notion of *weakest* pre-expectation. Note that in case T is not reachable from s under a demonic policy, $\mathit{ExpRew}(s \models \Diamond T) = 0$. $\mathit{LExpRew}(s \models \Diamond T)$ is the expected reward to reach T or never reach it from s. In case there is no policy under which T can be reached from s, we have that $\mathit{LExpRew}(s \models \Diamond T) = 1$. Note that *ExpRew* and *LExpRew* coincide if T is reached with probability 1. For finite MDPs without parameters, expected and liberal expected rewards for reachability objectives can be obtained by solving a linear programming problem. A detailed description is outside the scope of this paper; its analogue for Markov chains is fully described in [1, Ch. 10.5].

Example 12. (Expected rewards)



Let $T = \{s_2, s_3\}$. Then $\mathit{ExpRew}(s_0 \models \Diamond T) = \min\{2, \frac{59}{6}\} = 2$. And $\mathit{LExpRew}(s_0 \models \Diamond T) = \min\{2.5, 10\} = 2.5$. ∎

IV. pGCL SEMANTICS

This section describes an expectation transformer semantics of pGCL, as well as an operational semantics using MDPs. The main result of this section is a formal connection between these two semantics.

A. Denotational Semantics

When probabilistic programs are executed they determine a probability distribution over final values of program variables. For instance, on termination of

$$(x := 1 \ [0.75] \ x := 2);$$

the final value of x is 1 with probability $\frac{3}{4}$ or 2 with probability $\frac{1}{4}$. An alternative way to characterise that probabilistic behaviour is to consider the expected values over random variables with respect to that distribution. For example, to determine the probability that x is set to 1, we can compute the expected value of the random variable "x is 1" which is $\frac{3}{4} \cdot 1 + \frac{1}{4} \cdot 0 = \frac{3}{4}$. Similarly, to determine the average value of x, we compute the expected value of the random variable "x" which is $\frac{3}{4} \cdot 1 + \frac{1}{4} \cdot 2 = \frac{5}{4}$.

More generally, rather than a distribution-centred approach, we take an "expectation transformer" [10] approach. We annotate probabilistic programs with *expectations*, cf. [10]. Expectations are functions which map program states to real values. They are the quantitative analogue to Hoare's predicates for non-probabilistic programs. An expectation transformer is a total function between two expectations on the state of a program. The transformer $\mathit{wp}(P, f)$ for program P and post-expectation f yields the least expected value e on P's initial state ensuring that P's execution terminates with a value f. Annotation $\{e\}\, P\, \{f\}$ holds for total correctness if and only if $e \ \leq \ \mathit{wp}(P, f)$ where $\leq$ is to be interpreted in a point-wise manner. Intuitively, implication between predicates is generalised to pointwise inequality between expectations. For convenience we use square brackets to link boolean truth values to numbers and by convention $[\text{true}] = 1$ and $[\text{false}] = 0$.

Definition 13. (*wp*-semantics of pGCL) Let P and Q be pGCL-programs, f a post-expectation, x a program variable, E an expression, and G a Boolean expression. The *wp*-semantics of a program is defined by structural induction follows:

- $\mathit{wp}(\mathsf{skip}, f) = f$
- $\mathit{wp}(\mathsf{abort}, f) = 0$
- $\mathit{wp}(x := E, f) = f[x := E]$
- $\mathit{wp}(P; Q, f) = \mathit{wp}(P, \mathit{wp}(Q, f))$
- $\mathit{wp}(\mathsf{if}(G)\{P\}\mathsf{else}\{Q\}, f) = [G] \cdot \mathit{wp}(P, f) + [\neg G] \cdot \mathit{wp}(Q, f)$
- $\mathit{wp}(P \,[]\, Q, f) = \min\left(\mathit{wp}(P, f), \mathit{wp}(Q, f)\right)$
- $\mathit{wp}(P \,[p]\, Q, f) = p \cdot \mathit{wp}(P, f) + (1{-}p) \cdot \mathit{wp}(Q, f)$
- $\mathit{wp}(\mathsf{while}(G)\{P\}, f) = \mu X.\, ([G]{\cdot}\mathit{wp}(P, X){+}[\neg G]{\cdot} f)$

Here μ is the least fixed point operator w.r.t. the ordering $\leq$ on expectations. ∎

If program P does not contain a probabilistic choice, then this *wp* is isomorphic to Dijkstra's *wp* [10]. A weakest liberal pre-expectation $\mathit{wlp}(P, f)$ yields the least expectation for which P either does not terminate or establishes f.

Definition 14. (*wlp*-semantics of pGCL) *wlp*-semantics differs from *wp*-semantics only for while and abort:

- $\mathit{wlp}(\mathsf{abort}, f) = 1$
- $\mathit{wlp}(\mathsf{while}(G)\{P\}, f) = \nu X.\, ([G]{\cdot}\mathit{wlp}(P, X){+}[\neg G]{\cdot} f)$

Here ν is the greatest fixed point operator w.r.t. the ordering $\leq$ on expectations. ∎

So the difference between *wp* and *wlp* is lies in the handling non-termination. As for *ExpRew* and *LExpRew* the expectation transformers *wp* and *wlp* coincide for programs that terminate with probability 1.

Example 15. (Application of *wlp*-semantics) Consider again the duelling cowboys example. Assume we are given

the post-expectation:

$$f = [t = A \wedge c = 0] + [t = A \wedge c = 1] \cdot \frac{a}{a+b-ab} + [t = B \wedge c = 1] \cdot \frac{(1-b)a}{a+b-ab} \ .$$

Let us compute the weakest liberal pre-expectation of the loop body from Lst. 1 w.r.t. the post-expectation f. This yields:

$$\begin{aligned}
&\mathit{wlp}(\mathsf{if}(t = A)\{(c := 0\,[a]\,t := B);\}\\
&\qquad \mathsf{else}\{(c := 0\,[b]\,t := A);\}, f)\\
&= [t = A] \cdot \mathit{wlp}((c := 0\,[a]\,t := B), f)\\
&\quad + [t \neq A] \cdot \mathit{wlp}((c := 0\,[b]\,t := A), f)\\
&= [t = A] \cdot (a \cdot \mathit{wlp}((c := 0), f) + (1-a) \cdot \mathit{wlp}(t := B, f))\\
&\quad + [t \neq A] \cdot (b \cdot \mathit{wlp}((c := 0, f) + (1-b) \cdot \mathit{wlp}(t := A, f))\\
&= [t = A \wedge c \neq 1] \cdot a + [t = A \wedge c = 1] \cdot \frac{a}{a+b-ab}\\
&\quad + [t \neq A \wedge c = 0] \cdot (1-b) + [t \neq A \wedge c = 1] \cdot \frac{(1-b)a}{a+b-ab}
\end{aligned}$$

The result of this example will be used in Sect. V. ■

Remark 16. (Expectations are bounded) Reasoning within denotational semantics requires a lower and upper bound on expectations. In [10] expectations are defined to be non-negative with 0 as the least element and 1 as the maximum. We just note that these bounds can be altered or even given up provided that the program at hand has certain properties - the discussion of details is beyond this work. In the following we stick to the original definitions with bounds 0 and 1. ■

B. Operational Semantics

Our aim is to model the stepwise behaviour of a pGCL-program P by an MDP denoted $\mathcal{M}[\![\, P \,]\!]$. This MDP represents the operational interpretation of the program P and intuitively acts as an abstract machine for P. This is done as follows. Let η be a variable valuation of the program variables. That is, η is a mapping from the program variables onto their (possibly infinite) domains. For variable x, $\eta(x)$ denotes the value of x under η. For expression E, let $[\![\, E \,]\!]_\eta$ denote the valuation of E under valuation η. This is defined in the standard way, e.g., for $E = 2 * x{+}y$ with $\eta(x) = 3$ and $\eta(y) = 7$, we have $[\![\, E \,]\!]_\eta = 2 * \eta(x){+}\eta(y) = 13$. We use the distinguished semantic construct exit to denote the successful termination of a program. States in the MDP are of the form $\langle Q, \eta \rangle$ with Q a pGCL-statement or $Q = \mathsf{exit}$ and η a variable valuation. For instance, the execution of the assignment $x := 2 * x{+}y$ under evaluation η with $\eta(x) = 3$ and $\eta(y) = 7$ results in the state $\langle \mathsf{exit}, \eta' \rangle$ where η' is the same as η except that $\eta'(x) = 13$. Initial states of program P are tuples $\langle P, \eta \rangle$ where η maps any variable onto an arbitrary value.

Definition 17. (Operational semantics of pGCL) The operational semantics of pGCL-program P, denoted $\mathcal{M}[\![\, P \,]\!]$, is the MDP $(S, S_0, \to)$ where:

TABLE I
INFERENCE RULES FOR pGCL PROGRAMS

$$\langle \mathsf{skip}, \eta \rangle \to \langle \mathsf{exit}, \eta \rangle \qquad \langle \mathsf{abort}, \eta \rangle \to \langle \mathsf{abort}, \eta \rangle$$

$$\langle x := \mathsf{expr}, \eta \rangle \to \langle \mathsf{exit}, \eta[x := [\![\, \mathsf{expr} \,]\!]_\eta] \rangle$$

$$\frac{\langle P, \eta \rangle \to \mu}{\langle P; Q, \eta \rangle \to \nu} \text{ with } \nu(\langle P'; Q, \eta' \rangle) = \mu(\langle P', \eta' \rangle) \text{ where } \mathsf{exit}; Q = Q.$$

$$\langle P \ [] \ Q, \eta \rangle \to \langle P, \eta \rangle \qquad \langle P \ [] \ Q, \eta \rangle \to \langle Q, \eta \rangle$$

$$\langle P \ [p] \ Q, \eta \rangle \to \mu \text{ with } \mu(\langle P, \eta \rangle) = p \text{ and } \mu(\langle Q, \eta \rangle) = 1{-}p$$

$$\frac{\eta \models G}{\langle \mathsf{if}(G)\{P\}\,\mathsf{else}\,\{Q\}, \eta \rangle \to \langle P, \eta \rangle}$$

$$\frac{\eta \not\models G}{\langle \mathsf{if}(G)\{P\}\,\mathsf{else}\,\{Q\}, \eta \rangle \to \langle Q, \eta \rangle}$$

$$\frac{\eta \models G}{\langle \mathsf{while}(G)\{P\}, \eta \rangle \to \langle P; \mathsf{while}(G)\{P\}, \eta \rangle}$$

$$\frac{\eta \not\models G}{\langle \mathsf{while}(G)\{P\}, \eta \rangle \to \langle \mathsf{exit}, \eta \rangle}$$

- S is the set of pairs $\langle Q, \eta \rangle$ with Q a pGCL-program or $Q = \mathsf{exit}$, and η is a variable valuation of the variables occurring in P,
- $S_0 = \{\, \langle P, \eta \rangle \,\}$ where η maps every variabe in P to an arbitrary value, and
- $\to$ is the smallest relation that is induced by the inference rules in Table I.

■

Example 18. (Operational semantics) Figure 1 depicts the MDP underlying the cowboy example. This MDP is parameterized with parameters a and b but has a finite state space. A slight adaptation of our example program in which we keep track of the number of shots before one of the cowboys dies, yields an MDP with infinitely many states. The support of any distribution in this MDP is finite however. ■

Let $P^{\checkmark}$ denote the set of states in MDP $\mathcal{M}[\![\, P \,]\!]$ of the form $\langle \mathsf{exit}, \eta \rangle$ for arbitrary variable valuation η. Note that states in $P^{\checkmark}$ represent the successful termination of P. If $P^{\checkmark} = \emptyset$, program P diverges under all possible policies.

Definition 19. (Reward-MDP of a pGCL-program) Let P be a pGCL-program and f a post-expectation for P. The reward-MDP associated to P and f is defined as $\mathcal{R}_f[\![\, P \,]\!] =$

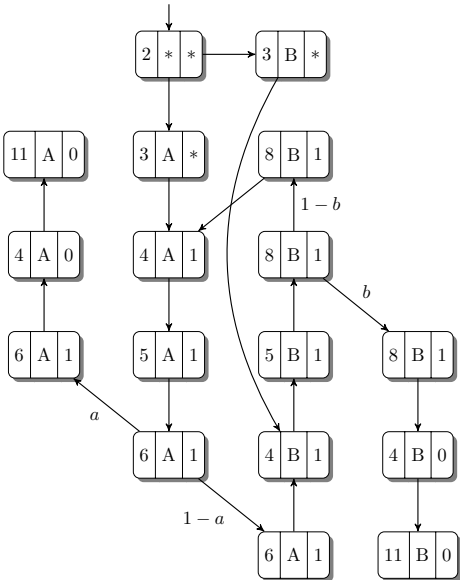


Fig. 1. MDP $\mathcal{M}$ for the duelling cowboys example. Each state is determined by a 3-tuple: (program location, value of t, value of c) where $*$ denotes an arbitrary value.

$(\mathcal{M}[\![\, P \,]\!], r)$ with $\mathcal{M}[\![\, P \,]\!]$ the MDP of P as defined before and reward function r defined by $r(s) = f(\eta)$ if $s = \langle \mathsf{exit}, \eta \rangle \in P^{\checkmark}$ and $r(s) = 0$ otherwise. ∎

Note that we use a special reward structure: only terminal states are assigned a reward which is not necessarily 0. All other states have a zero reward. This property allows us to rewrite the definition of expected rewards as follows.

Lemma 20. (Characterizing expected rewards)

For pGCL program P and variable valuation η, we have:

$$\begin{aligned} &\mathit{ExpRew}^{\mathcal{R}_f[\![\, P \,]\!]}(\langle P, \eta \rangle \models \Diamond P^{\checkmark}) \\ &= \min_{\mathfrak{P}} \sum_{\widehat{\pi} \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s, \Diamond P^{\checkmark})} \mathbf{P}(\widehat{\pi}) \cdot r_{P^{\checkmark}}(\widehat{\pi}) \ , \end{aligned}$$

where $\mathit{Paths}^{\mathfrak{P}}_{\min}(s, \Diamond T)$ is the set containing all finite paths of the form $s_0 \dots s_k$ with $s_0 = s$, $s_k \in T$ and $s_i \notin T$ for all $0 \leq i < k$ that adhere to the policy $\mathfrak{P}$. ∎

Proof: Let $T = P^{\checkmark}$ for pGCL program P. The proof has two ingredients. First, we observe that a path which fails to reach a final state has reward 0 according to the *wp* semantics of abort. Secondly, in a finitely-branching MDP with countably many states there are "only" countably many paths that reach any given set. Consider the definition of expected reward:

$$\min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\{\, \pi \in \mathit{Paths}^{\mathfrak{P}}(s, \Diamond T) \mid r_T(\pi) = c \,\}.$$

Given that $Pr(\pi \models \Diamond T) = \mathbf{P}(\widehat{\pi})$ where prefix $\widehat{\pi}$ of π is minimal and ends in T, the above term equals:

$$\min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathbf{P}\{\widehat{\pi} \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s, T) \mid r_T(\widehat{\pi}) = c\}.$$

As $\mathcal{M}[\![\, P \,]\!]$ is a finitely branching MC, there are countably many $\widehat{\pi}$ for each reward c. This yields:

$$\min_{\mathfrak{P}} \sum_{\pi \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s, T)} \mathbf{P}(\widehat{\pi}) \cdot r_T(\widehat{\pi})$$

∎

We use this fact in our proofs later on.

Remark 21. (Real valued rewards) Lemma 20 provides a straight-forward way to calculate expected rewards when the rewards are real valued instead of just integer. This is because the summation runs not over the possible cumulative rewards (of which there are uncountably many in the case of real valued rewards) but over the possible paths that reach an *exit* state. In the following we stay with integer rewards as introduced earlier but bear in mind that Theorems 23 and 24 also hold for real valued post-expectations. ∎

Analogously we obtain:

Lemma 22. (Characterizing liberal expected rewards)

For pGCL program P and variable valuation η, we have:

$$\begin{aligned} &\mathit{LExpRew}^{\mathcal{R}_f[\![\, P \,]\!]}(\langle P, \eta \rangle \models \Diamond P^{\checkmark}) \\ &= \min_{\mathfrak{P}} \sum_{\pi \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s, P^{\checkmark})} \mathbf{P}(\widehat{\pi}) \cdot r_{P^{\checkmark}}(\widehat{\pi}) + \mathit{Pr}^{\mathfrak{P}}(\langle P, \eta \rangle \not\models \Diamond P^{\checkmark}) \ . \end{aligned}$$

∎

Proof: Follows immediately from Lemma 20. ∎

C. Main Results

This brings us at a position to present our main results of this paper: a formal relationship between the *wp*-semantics of pGCL-program P and its operational semantics in terms of a reward-MDP, and similary for the *wlp*-semantics. We first consider the *wp*-semantics.

Theorem 23. (Operational vs. *wp*-semantics) For pGCL-program P, variable valuation η, and post-expectation f:

$$\mathit{wp}(P, f)(\eta) \ = \ \mathit{ExpRew}^{\mathcal{R}_f[\![\, P \,]\!]}(\langle P, \eta \rangle \models \Diamond P^{\checkmark}).$$ ∎

Proof: By structural induction over the pGCL program P. For the sake of convenience, let $\mathit{Paths}(P^{\checkmark}, \eta, c)$ denote the set

$$\{\, \pi \in \mathit{Paths}(\langle P, \eta \rangle, \Diamond P^{\checkmark}) \mid r_{P^{\checkmark}}(\pi) = c \,\}.$$

Furthermore we write paths as sequences of states and leave out the distribution in between each pair of states because it is obvious. Induction base:

- For $P = \mathsf{skip}$ we derive:

$$\begin{aligned} &\mathit{ExpRew}^{\mathcal{R}_f[\![\, \mathsf{skip} \,]\!]}(\langle \mathsf{skip}, \eta \rangle \models \Diamond \mathsf{skip}^{\checkmark}) \\ &= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(\mathsf{skip}^{\checkmark}, \eta, c)\right) \\ &= f(\eta) \cdot \mathit{Pr}\{\pi = \langle \mathsf{skip}, \eta \rangle \langle \mathsf{exit}, \eta \rangle \mid r_{\mathsf{skip}^{\checkmark}}(\pi) = f(\eta)\} \\ &= f(\eta) \cdot 1 \\ &= f(\eta) \\ &= \mathit{wp}(\mathsf{skip}, f)(\eta). \end{aligned}$$

- For $P = \mathsf{abort}$ we derive:

$$\begin{aligned}
&\mathit{ExpRew}^{\mathcal{R}_f[\![\,\mathsf{abort}\,]\!]}(\langle \mathsf{abort}, \eta\rangle \models \Diamond \mathsf{abort}^{\surd}) \\
&= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(\mathsf{abort}^{\surd}, \eta, c)\right) \\
&= 0 \\
&= \mathit{wp}(\mathsf{abort}, f)(\eta)
\end{aligned}$$

  as there is no path starting from $\langle \mathsf{abort}, \eta\rangle$ that reaches an exit-state.
- Let P be the assigment $x := E$. For this case, we have:

$$\begin{aligned}
&\mathit{ExpRew}^{\mathcal{R}_f[\![\,x:=E\,]\!]}(\langle x := E, \eta\rangle \models \Diamond x := E^{\surd}) \\
&= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(x := E^{\surd}, \eta, c)\right) \\
&= f(\eta[x/E]) \cdot \mathit{Pr}\{\, \pi = \langle x := E, \eta\rangle\langle \mathsf{exit}, \eta[x/E]\rangle \\
&\qquad\qquad \mid r_{x:=E^{\surd}}(\pi) = f(\eta[x/E])\,\} \\
&= f(\eta[x/E]) \cdot 1 \\
&= f(\eta[x/E]) \\
&= \mathit{wp}(x := E, f)(\eta).
\end{aligned}$$

Induction hypothesis: assume

$$\mathit{wp}(P, f)(\eta) = \mathit{ExpRew}^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\surd}).$$

Induction step:

- Consider the probabilistic choice $P\,[p]\,Q$ (this also covers conditional choice since it can be written as $P\,[G]\,Q$):

$$\begin{aligned}
&\mathit{ExpRew}^{\mathcal{R}_f[\![\,P\,[p]\,Q\,]\!]}(\langle P\,[p]\,Q, \eta\rangle \models \Diamond (P\,[p]\,Q)^{\surd}) \\
&= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}((P\,[p]\,Q)^{\surd}, \eta, c)\right) \\
&= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot p \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(P^{\surd}, \eta, c)\right) \\
&\quad + \sum_{c=0}^{\infty} c \cdot (1-p) \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(Q^{\surd}, \eta, c)\right) \\
&\overset{*}{=} p \cdot \min_{\mathfrak{P}_1} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}_1}\left(\mathit{Paths}^{\mathfrak{P}_1}(P^{\surd}, \eta, c)\right) \\
&\quad + (1-p) \cdot \min_{\mathfrak{P}_2} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}_2}\left(\mathit{Paths}^{\mathfrak{P}_2}(Q^{\surd}, \eta, c)\right) \\
&= p \cdot \mathit{ExpRew}^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\surd}) \\
&\quad + (1-p) \cdot \mathit{ExpRew}^{\mathcal{R}_f[\![\,Q\,]\!]}(\langle Q, \eta\rangle \models \Diamond Q^{\surd}) \\
&\overset{I.H.}{=} p \cdot \mathit{wp}(P, f)(\eta) + (1-p) \cdot \mathit{wp}(Q, f)(\eta) \\
&= \mathit{wp}(P\,[p]\,Q, f)(\eta)
\end{aligned}$$

  In $*$ we use the fact that the policy for paths starting in $\langle P, \eta\rangle$ is independent of the policy for paths starting in $\langle Q, \eta\rangle$.
- Consider the non-deterministic choice $P\,[]\,Q$:

$$\begin{aligned}
&\mathit{ExpRew}^{\mathcal{R}_f[\![\,P\,[]\,Q\,]\!]}(\langle P\,[]\,Q, \eta\rangle \models \Diamond (P\,[]\,Q)^{\surd}) \\
&= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}((P\,[]\,Q)^{\surd}, \eta, c)\right) \\
&= \min\left\{ \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(P^{\surd}, \eta, c)\right), \right. \\
&\qquad \left. \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}(Q^{\surd}, \eta, c)\right) \right\} \\
&= \min\{\mathit{ExpRew}^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\surd}), \\
&\qquad \mathit{ExpRew}^{\mathcal{R}_f[\![\,Q\,]\!]}(\langle Q, \eta\rangle \models \Diamond Q^{\surd})\} \\
&\overset{I.H.}{=} \min\{\mathit{wp}(P, f), \mathit{wp}(Q, f)\} \\
&= \mathit{wp}(P[]Q, f)(\eta)
\end{aligned}$$

- Consider the sequential composition $P; Q$:

$$\begin{aligned}
&\mathit{ExpRew}^{\mathcal{R}_f[\![\,P;Q\,]\!]}(\langle P; Q, \eta\rangle \models \Diamond (P; Q)^{\surd}) \\
&= \min_{\mathfrak{P}} \sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}((P; Q)^{\surd}, \eta, c)\right) \\
&\overset{La20}{=} \min_{\mathfrak{P}} \sum_{\widehat{\pi} \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s, P^{\surd})} \mathbf{P}(\widehat{\pi}) \cdot r_{P;Q^{\surd}}(\widehat{\pi}) \\
&\overset{*}{=} \min_{\mathfrak{P}} \sum_{\widehat{\pi} \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s, P^{\surd})} \mathbf{P}(\widehat{\pi}) \cdot r^{q}_{P^{\surd}}(\widehat{\pi})
\end{aligned}$$

  where $r^{q}_{P^{\surd}}(\widehat{\pi})$ is the sum of rewards r_q along $\widehat{\pi}$ with

$$r_q(s) = \min_{\mathfrak{P}'} \left(\sum_{\widehat{\pi}' \in \mathit{Paths}^{\mathfrak{P}'}_{\min}(s, Q^{\surd})} \mathbf{P}(\widehat{\pi}') \cdot r_{Q^{\surd}}(\widehat{\pi}') \right)$$

  if $s = \langle \mathsf{exit}, \eta'\rangle \in P^{\surd}$ and $r_q(s) = 0$ otherwise

$$\begin{aligned}
&= \mathit{ExpRew}^{\mathcal{R}_g[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\surd}) \\
&\quad \text{where } g(\eta) = \mathit{ExpRew}^{\mathcal{R}_f[\![\,Q\,]\!]}(\langle Q, \eta\rangle \models \Diamond Q^{\surd}) \\
&\overset{I.H.}{=} \mathit{wp}(P; \mathit{wp}(Q, f))(\eta) \\
&= \mathit{wp}(P; Q, f)(\eta)\ .
\end{aligned}$$

  In $*$ we rewrite each single path into a prefix which corresponds to the execution of P and all possible continuations according to Q. Then we can compute the expected reward r_q of Q and use this as an intermediate result to compute the expected reward of the sequential composition.
- Consider the loop $\mathsf{while}(G)\{P\}$. This case is proven by induction on the number of iterations that a while-loop performs. Let the bounded while-loop for $k > 0$ be:

$$\begin{aligned}
&(\mathsf{while}(G)\{P\})^{k+1} \\
&= \mathsf{if}(G)\{P; (\mathsf{while}(G)\{P\})^{k}\}\ \mathsf{else}\ \{\mathsf{skip}\}\ .
\end{aligned}$$

  where the base case is $(\mathsf{while}(G)\{P\})^{0} = \mathsf{abort}$. We will show for every k that

$$\begin{aligned}
&\mathit{wp}((\mathsf{while}(G)\{P\})^{k}, f)(\eta) \\
&= \mathit{ExpRew}^{\mathcal{R}_f[\![\,(\mathsf{while}(G)\{P\})^{k}\,]\!]}(\eta)\ .
\end{aligned} \tag{1}$$

Observe that

$$\begin{aligned} &wp((\mathsf{while}(G)\{P\})^{k+1}, f)(\eta) \\ \geq\ &wp((\mathsf{while}(G)\{P\})^{k}, f)(\eta)\ . \end{aligned}$$

From the fixpoint theorem 3 in [7] we know that the more iterations the bounded while loop is allowed to perform the closer it approximates the fixpoint given in Def. 13. Formally this means

$$\begin{aligned} &\lim_{k\to\infty} wp((\mathsf{while}(G)\{P\})^{k}, f)(\eta) \\ =\ &wp(\mathsf{while}(G)\{P\}, f)(\eta)\ . \end{aligned} \tag{2}$$

From (1) it follows that for every k, *ExpRew* behaves identically to *wp*. Thus with (2) it follows that

$$wp((\mathsf{while}(G)\{P\}), f)(\eta) = ExpRew^{\mathcal{R}_f[\![\,(\mathsf{while}(G)\{P\})\,]\!]}(\eta).$$

It remains to prove (1). This is done by induction on k.
Base case ($k = 0$):

$$\begin{aligned} &wp((\mathsf{while}(G)\{P\})^{0}, f)(\eta) \\ =\ &wp(\mathsf{abort}, f)(\eta) \\ \overset{*}{=}\ &ExpRew^{\mathcal{R}_f[\![\,\mathsf{abort}\,]\!]}(\eta) \\ =\ &ExpRew^{\mathcal{R}_f[\![\,(\mathsf{while}(G)\{P\})^{0}\,]\!]}(\eta) \end{aligned}$$

($*$) was already shown earlier in the case abort.
Induction hypothesis: equation (1) holds for some unspecified but fixed value of k.
Induction step:

$$\begin{aligned} &wp((\mathsf{while}(G)\{P\})^{k+1}, f)(\eta) \\ =\ &wp(\mathsf{if}(G)\{P;(\mathsf{while}(G)\{P\})^{k}\}\mathsf{else}\{\mathsf{skip}\}, f)(\eta) \\ =\ &[G]\cdot wp(P;(\mathsf{while}(G)\{P\})^{k}) + [\neg G]\cdot wp(\mathsf{skip}, f)(\eta) \\ \overset{*}{=}\ &[G]\cdot ExpRew^{\mathcal{R}_f[\![\,P;(\mathsf{while}(G)\{P\})^{k}\,]\!]}(\eta) \\ &+ [\neg G]\cdot ExpRew^{\mathcal{R}_f[\![\,\mathsf{skip}\,]\!]}(\eta) \\ =\ &ExpRew^{\mathcal{R}_f[\![\,\mathsf{if}(G)\{P;(\mathsf{while}(G)\{P\})^{k}\}\mathsf{else}\{\mathsf{skip}\}\,]\!]}(\eta) \\ =\ &ExpRew^{\mathcal{R}_f[\![\,(\mathsf{while}(G)\{P\})^{k+1}\,]\!]}(\eta) \end{aligned}$$

($*$) follows from the induction hypothesis and the previously shown cases for skip and sequential composition. ∎

Thus, $wp(P, f)$ evaluated at η is the least expected value of f over any of the result distributions of P.

Theorem 24. (Operational vs. *wlp*-semantics) For pGCL-program P, variable valuation η, and post-expectation f:

$$wlp(P, f)(\eta)\ =\ LExpRew^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\checkmark}).$$ ∎

Proof: By structural induction over the pGCL program P (analogously to the proof of Theorem 23). Due to space limitations we skip the base cases which are rather simple.
Induction hypothesis: assume

$$wlp(P, f)(\eta) = LExpRew^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\checkmark}).$$

Induction step:

- Consider the probabilistic choice $P\,[p]\,Q$ (again, this covers conditional choice):

$$\begin{aligned} &LExpRew^{\mathcal{R}_f[\![\,P\,[p]\,Q\,]\!]}(\langle P\,[p]\,Q, \eta\rangle \models \Diamond (P\,[p]\,Q)^{\checkmark}) \\ =\ &\min_{\mathfrak{P}} \Bigg(\sum_{c=0}^{\infty} c\cdot Pr^{\mathfrak{P}}\left(Paths^{\mathfrak{P}}((P\,[p]\,Q)^{\checkmark}, \eta, c)\right) \\ &\quad + Pr^{\mathfrak{P}}(s \models \Diamond (P\,[p]\,Q)^{\checkmark})\Bigg) \\ =\ &\min_{\mathfrak{P}} \Bigg(\sum_{c=0}^{\infty} c\cdot p\cdot Pr^{\mathfrak{P}}\left(Paths^{\mathfrak{P}}(P^{\checkmark}, \eta, c)\right) \\ &\quad + p\cdot Pr^{\mathfrak{P}}(s \models \Diamond P^{\checkmark}) \\ &\quad + \sum_{c=0}^{\infty} c\cdot (1-p)\cdot Pr^{\mathfrak{P}}\left(Paths^{\mathfrak{P}}(Q^{\checkmark}, \eta, c)\right) \\ &\quad + (1-p)\cdot Pr^{\mathfrak{P}}(s \models \Diamond Q^{\checkmark})\Bigg) \\ =\ &p\cdot \min_{\mathfrak{P}_1} \Bigg(\sum_{c=0}^{\infty} c\cdot Pr^{\mathfrak{P}_1}\left(Paths^{\mathfrak{P}_1}(P^{\checkmark}, \eta, c)\right) \\ &\quad + Pr^{\mathfrak{P}_1}(s \models \Diamond P^{\checkmark})\Bigg) \\ &\quad + (1-p)\cdot \min_{\mathfrak{P}_2} \Bigg(\sum_{c=0}^{\infty} c\cdot Pr^{\mathfrak{P}_2}\left(Paths^{\mathfrak{P}_2}(Q^{\checkmark}, \eta, c)\right) \\ &\quad + Pr^{\mathfrak{P}_2}(s \models \Diamond Q^{\checkmark})\Bigg) \\ =\ &p\cdot LExpRew^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\checkmark}) \\ &\quad + (1-p)\cdot LExpRew^{\mathcal{R}_f[\![\,Q\,]\!]}(\langle Q, \eta\rangle \models \Diamond Q^{\checkmark}) \\ \overset{I.H.}{=}\ &p\cdot wlp(P, f)(\eta) + (1-p)\cdot wlp(Q, f)(\eta) \\ =\ &wlp(P\,[p]\,Q, f)(\eta)\ . \end{aligned}$$

- Consider the non-deterministic choice $P[]Q$:

$$\begin{aligned} &LExpRew^{\mathcal{R}_f[\![\,P\,[]\,Q\,]\!]}(\langle P\,[]\,Q, \eta\rangle \models \Diamond (P\,[]\,Q)^{\checkmark}) \\ =\ &\min_{\mathfrak{P}} \Bigg(\sum_{c=0}^{\infty} c\cdot Pr^{\mathfrak{P}}\left(Paths^{\mathfrak{P}}((P\,[]\,Q)^{\checkmark}, \eta, c)\right) \\ &\quad + Pr^{\mathfrak{P}}(s \models \Diamond (P\,[]\,Q)^{\checkmark})\Bigg) \\ =\ &\min\Bigg\{ \min_{\mathfrak{P}} \Bigg(\sum_{c=0}^{\infty} c\cdot Pr^{\mathfrak{P}}\left(Paths^{\mathfrak{P}}(P^{\checkmark}, \eta, c)\right) \\ &\quad + Pr^{\mathfrak{P}}(s \models \Diamond P^{\checkmark})\Bigg), \\ &\quad \min_{\mathfrak{P}} \Bigg(\sum_{c=0}^{\infty} c\cdot Pr^{\mathfrak{P}}\left(Paths^{\mathfrak{P}}(Q^{\checkmark}, \eta, c)\right) \\ &\quad + Pr^{\mathfrak{P}}(s \models \Diamond Q^{\checkmark})\Bigg)\Bigg\} \\ =\ &\min\{LExpRew^{\mathcal{R}_f[\![\,P\,]\!]}(\langle P, \eta\rangle \models \Diamond P^{\checkmark}), \\ &\quad LExpRew^{\mathcal{R}_f[\![\,Q\,]\!]}(\langle Q, \eta\rangle \models \Diamond Q^{\checkmark})\} \\ \overset{I.H.}{=}\ &\min\{wlp(P, f)(\eta), wlp(Q, f)(\eta)\} \\ =\ &wlp(P[]Q, f)(\eta)\ . \end{aligned}$$

- Consider the sequential composition $P;Q$:

$$
\begin{aligned}
&\mathit{LExpRew}^{\mathcal{R}_f[\![P;Q]\!]}(\langle P;Q,\eta\rangle \models \Diamond(P;Q)^{\surd}) \\
&= \min_{\mathfrak{P}} \left(\sum_{c=0}^{\infty} c \cdot \mathit{Pr}^{\mathfrak{P}}\left(\mathit{Paths}^{\mathfrak{P}}((P;Q)^{\surd},\eta,c)\right) \right. \\
&\quad \left. + \mathit{Pr}^{\mathfrak{P}}(s \not\models \Diamond(P;Q)^{\surd})\right) \\
&\overset{La22}{=} \min_{\mathfrak{P}} \left(\sum_{\pi \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s,P^{\surd})} \mathbf{P}(\widehat{\pi}) \cdot r_{P;Q^{\surd}}(\widehat{\pi}) \right. \\
&\quad \left. + \mathit{Pr}^{\mathfrak{P}}\{\langle P;Q,\eta\rangle \not\models \Diamond P;Q^{\surd}\}\right) \\
&= \min_{\mathfrak{P}} \left(\sum_{\pi \in \mathit{Paths}^{\mathfrak{P}}_{\min}(s,P^{\surd})} \mathbf{P}(\widehat{\pi}) \cdot r^{q}_{P^{\surd}}(\widehat{\pi}) \right. \\
&\quad \left. + \mathit{Pr}^{\mathfrak{P}}\{\langle P,\eta\rangle \not\models \Diamond P^{\surd}\}\right) \\
&\quad \text{where } r^{q}_{P^{\surd}}(\widehat{\pi}) \text{ is the sum of rewards } r_q \text{ along } \widehat{\pi} \text{ with} \\
&\quad r_q(s) = \min_{\mathfrak{P}'} \left(\sum_{\widehat{\pi}' \in \mathit{Paths}^{\mathfrak{P}'}_{\min}(s,Q^{\surd})} \mathbf{P}(\widehat{\pi}') \cdot r_{Q^{\surd}}(\widehat{\pi}') \right. \\
&\qquad \left. + \mathit{Pr}^{\mathfrak{P}}\{\langle Q,\eta'\rangle \not\models \Diamond Q^{\surd}\}\right) \\
&\quad \text{if } s = \langle \mathit{exit},\eta'\rangle \in P^{\surd} \text{and } r_q(s) = 0 \text{ otherwise} \\
&= \mathit{LExpRew}^{\mathcal{R}_g[\![P]\!]}(\langle P,\eta\rangle \models \Diamond P^{\surd}) \\
&\quad \text{where } g(\eta) = \mathit{LExpRew}^{\mathcal{R}_f[\![Q]\!]}(\langle Q,\eta\rangle \models \Diamond Q^{\surd}) \\
&\overset{I.H.}{=} \mathit{wlp}(P;\mathit{wlp}(Q,f))(\eta) \\
&= \mathit{wlp}(P;Q,f)(\eta) \ .
\end{aligned}
$$

- Consider the while loop $\mathsf{while}(G)\{P\}$. Again we prove this case by induction on the number of iterations that a while-loop performs. Let $(\mathsf{while}(G)\{P\})^k$ be defined as in the proof of the previous theorem. We show for every k that

$$
\begin{aligned}
&\mathit{wlp}((\mathsf{while}(G)\{P\})^k, f)(\eta) \\
&= \mathit{LExpRew}^{\mathcal{R}_f[\![(\mathsf{while}(G)\{P\})^k]\!]}(\eta) \ .
\end{aligned} \tag{3}
$$

The only difference is now that

$$
\begin{aligned}
&\mathit{wlp}((\mathsf{while}(G)\{P\})^{k+1}, f)(\eta) \\
&\leq \mathit{wlp}((\mathsf{while}(G)\{P\})^{k}, f)(\eta) \ .
\end{aligned}
$$

Using this we again know that the bounded while loop approximates the fixpoint given in Def. 14 (only this time from above). Formally this means

$$
\begin{aligned}
&\lim_{k\to\infty} \mathit{wlp}((\mathsf{while}(G)\{P\})^k, f)(\eta) \\
&= \mathit{wlp}(\mathsf{while}(G)\{P\}, f)(\eta) \ .
\end{aligned} \tag{4}
$$

From (3) we know that for every k *LExpRew* behaves identically to *wlp*. Thus with (4) it follows that

$$
\begin{aligned}
&\mathit{wlp}((\mathsf{while}(G)\{P\}), f)(\eta) \\
&= \mathit{LExpRew}^{\mathcal{R}_f[\![(\mathsf{while}(G)\{P\})]\!]}(\eta) \ .
\end{aligned}
$$

It remains to prove (3). This is done by induction on k. Base case ($k = 0$):

$$
\begin{aligned}
&\mathit{wlp}((\mathsf{while}(G)\{P\})^0, f)(\eta) \\
&= \mathit{wlp}(\mathsf{abort}, f)(\eta) \\
&\overset{*}{=} \mathit{LExpRew}^{\mathcal{R}_f[\![\mathsf{abort}]\!]}(\eta) \\
&= \mathit{LExpRew}^{\mathcal{R}_f[\![(\mathsf{while}(G)\{P\})^0]\!]}(\eta)
\end{aligned}
$$

($*$) was already shown earlier in the case abort.

Induction hypothesis: equation (3) holds for some unspecified but fixed value of k.

Induction step:

$$
\begin{aligned}
&\mathit{wlp}((\mathsf{while}(G)\{P\})^{k+1}, f)(\eta) \\
&= \mathit{wlp}(\mathsf{if}(G)\{P;(\mathsf{while}(G)\{P\})^k\}\mathsf{else}\{\mathsf{skip}\}, f)(\eta) \\
&= [G] \cdot \mathit{wlp}(P;(\mathsf{while}(G)\{P\})^k) + [\neg G] \cdot \mathit{wlp}(\mathsf{skip}, f)(\eta) \\
&\overset{*}{=} [G] \cdot \mathit{LExpRew}^{\mathcal{R}_f[\![P;(\mathsf{while}(G)\{P\})^k]\!]}(\eta) \\
&\quad + [\neg G] \cdot \mathit{LExpRew}^{\mathcal{R}_f[\![\mathsf{skip}]\!]}(\eta) \\
&= \mathit{LExpRew}^{\mathcal{R}_f[\![\mathsf{if}(G)\{P;(\mathsf{while}(G)\{P\})^k\}\mathsf{else}\{\mathsf{skip}\}]\!]}(\eta) \\
&= \mathit{LExpRew}^{\mathcal{R}_f[\![(\mathsf{while}(G)\{P\})^{k+1}]\!]}(\eta)
\end{aligned}
$$

($*$) follows from the induction hypothesis and the previously shown cases for skip and sequential composition.

■

The weakest liberal pre-expectation $\mathit{wlp}(P,f)$ is thus the least expected value of f over any of the result distributions of P plus the probability that P does not terminate.

Example 25. (Duelling cowboys.) Consider again the duelling cowboys example from Lst. 1. Assume we are interested in the probability that cowboy A wins the duel. In terms of the MDP semantics this means we are interested in

$$
\mathit{LExpRew}^{(\mathcal{M},r)}(\langle 2,*,*\rangle \models \Diamond(\mathcal{M},r)^{\surd})
$$

where $\mathcal{M}$ is the MDP from Fig. 1 and r is the reward function that indicates whether cowboy A has won or not, i.e.

$$
r(s) = \begin{cases} 1 & \text{if } s = \langle 11, A, 0\rangle \\ 0 & \text{otherwise} \end{cases}
$$

In this example the MDP is finite and this allows us to compute the desired expected cumulative reward easily. That is, cowboy A wins with probability at least

$$
\frac{(1-b)a}{a+b-ab} \ .
$$

Figure 2 visualises this result. ■

V. Analysis

Although the computation of (liberal) expected rewards on MDPs may be numerically involved, it is intuitive in principle. However, pGCL programs will usually have an infinite state space due to the infinite domain of the program variables. It is then not possible to compute the expected reward on the reward model in general. In contrast to this, the denotational

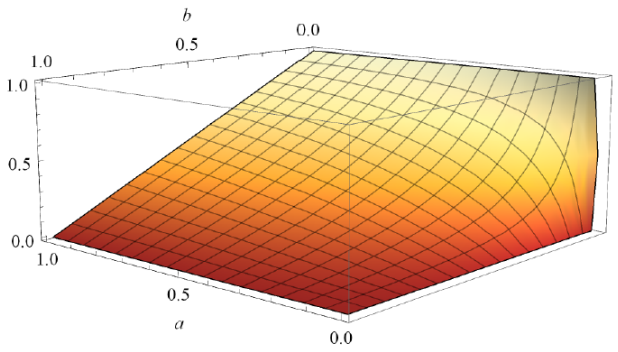


Fig. 2. Probability that A wins the duel, depending on a and b. Bear in mind that this is the least guaranteed probability that A wins. In the worst case (for A) cowboy B will shoot first and therefore as b tends to 1 the plot goes to 0, i.e. cowboy A has no chances. However for smaller values of b the influence of a increases.

semantics do not depend on the underlying state space but on the structure of the program. In this section we show how to determine a pre-expectation using *wlp*-semantics.

Again let us determine the probability that cowboy A wins the duel. Therefore we choose $[t = A]$ as the post-expectation and want to find *wlp*(cowboyDuel, $[t = A]$). Listing 2 shows the cowboy duelling program with annotations.

Listing 2. The duelling cowboys, annotated with expectations

```
1  int cowboyDuel(a, b) {
2    ⟨(1−b)a / a+b−ab⟩
3    ⟨min{a / a+b−ab, (1−b)a / a+b−ab}⟩
4    (t := A [] t := B);
5    ⟨[t = A] · a / a+b−ab + [t = B] · (1−b)a / a+b−ab⟩
6    c := 1;
7    ⟨[t = A ∧ c = 0] · 1 + [t = A ∧ c = 1] · a / a+b−ab
           +[t = B ∧ c = 1] · (1−b)a / a+b−ab⟩
8    while (c = 1) {
9      ⟨[t = A ∧ c = 1] · a / a+b−ab + [t = B ∧ c = 1] · (1−b)a / a+b−ab⟩
10     ⟨[t = A ∧ c ≠ 1] · a + [t = A ∧ c = 1] · a / a+b−ab
             +[t = B ∧ c = 0] · (1 − b) + [t = B ∧ c = 1] · (1−b)a / a+b−ab⟩
11     if (t = A) {
12       (c := 0 [a] t := B);
13     } else {
14       (c := 0 [b] t := A);
15     }
16     ⟨[t = A ∧ c = 0] · 1 + [t = A ∧ c = 1] · a / a+b−ab
             +[t = B ∧ c = 1] · (1−b)a / a+b−ab⟩
17   }
18   ⟨[c ≠ 1] · ( [t = A ∧ c = 0] · 1 + [t = A ∧ c = 1] · a / a+b−ab
             +[t = B ∧ c = 1] · (1−b)a / a+b−ab )⟩
19   ⟨[t = A]⟩
20   return t; // the survivor
21 }
```

The program is annotated backwards according to the rules from Def. 13 (and 14). In line 19 we start with the post-expectation that we are interested in. We finish with the sought probability in line 2. The only non-trivial step is to discover the so-called invariant which appears in line 7 and 16. But let us assume for the moment that it is given. Then all other annotations are obtained by applying the syntactic rules from Def. 13. In particular the calculation from line 16 to line 10 was already shown in Example 15. This means that the analysis can be automatically carried out by a computer once we have found the aforementioned invariant - irrespective of the underlying state space size.

The annotation in line 7 and 16 which we call invariant is an expectation that over-approximates the fixed point solution in Def. 14. More precisely, an annotation f is called invariant if

$$f \cdot [G] \leq \mathit{wlp}(\mathit{loop\_body}, f) \ . \tag{5}$$

In our example, f is the expectation in line 7, G is the loop guard $c = 1$ and *loop_body* is the code in lines 11–15. In line 9 the expectation represents $f \cdot [G]$ and line 10 is $\mathit{wlp}(\mathit{loop\_body}, f)$. Clearly, (5) is satisfied in our example.

The difficulty in reasoning with denotational semantics is to find suitable invariants. The invariant generation process is a topic on its own and beyond the scope of this paper. We refer to [4], [10] for this matter. Our recently developed tool PRINSYS[2] helps the user to find certain kinds of invariants semi-automatically.

VI. CONCLUSION

This paper provided a formal connection between the expectation transformer semantics of pGCL by McIver and Morgan [10] and a simple operational semantics using (parametric) MDPs. This yields an insightful relationship between semantics used for formal reasoning for probabilistic programs and the notion of a computation in terms of an MDP. Our approach assigns rewards to terminal states (only), and establishes that expected cumulative rewards correspond to wp-semantics. A slight variant of expected rewards yields a connection to the wlp-semantics.

Possible future work is to establish a relation to a denotational semantics in terms of metric spaces, like in [6] or to link our semantics to the seminal work by Kozen [5] where probabilistic programs are interpreted as partial measurable functions on a measurable space.

REFERENCES

[1] Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
[2] Dijkstra, E.W.: A Discipline of Programming. Prentice Hall (1976)
[3] Howard, R.A.: Dynamic Programming and Markov Processes. MIT Press (1960)
[4] Katoen, J.P., McIver, A., Meinicke, L., Morgan, C.: Linear-Invariant Generation for Probabilistic Programs. In: SAS. LNCS, Springer (2010)
[5] Kozen, D.: Semantics of probabilistic programs. J. Comput. Syst. Sci. 22(3), 328–350 (1981)
[6] Kwiatkowska, M.Z., Norman, G.: Probabilistic Metric Semantics for a Simple Language with Recursion. In: MFCS. LNCS, Springer (1996)
[7] Lassez, J.L., Nguyen, V.L., Sonenberg, L.: Fixed point theorems and semantics: A folk tale. Inf. Process. Lett. 14(3), 112–116 (1982)
[8] Lukkien, J.J.: An Operational Semantics for the Guarded Command Language. In: MPC. LNCS, vol. 669, pp. 233–249. Springer (1992)
[9] Lukkien, J.J.: Operational Semantics and Generalized Weakest Preconditions. Sci. Comput. Program. 22(1-2), 137–155 (1994)
[10] McIver, A., Morgan, C.: Abstraction, Refinement And Proof For Probabilistic Systems (Monographs in Computer Science). Springer (2004)

[2]Available at: http://www-i2.informatik.rwth-aachen.de/prinsys/.