Conditioning in Probabilistic Programming

Nils Jansen
and Benjamin Lucien Kaminski
and Joost-Pieter Katoen
and Federico Olmedo
RWTH Aachen University
Aachen, Germany

Friedrich Gretz
and Annabelle McIver
Macquarie University
Sydney, Australia

***Abstract*—We investigate the semantic intricacies of conditioning, a main feature in probabilistic programming. We provide a weakest (liberal) pre–condition (w(l)p) semantics for the elementary probabilistic programming language pGCL extended with conditioning. We prove that quantitative weakest (liberal) pre–conditions coincide with conditional (liberal) expected rewards in Markov chains and show that semantically conditioning is a truly conservative extension. We present two program transformations which entirely eliminate conditioning from any program and prove their correctness using the w(l)p–semantics. Finally, we show how the w(l)p–semantics can be used to determine conditional probabilities in a parametric anonymity protocol and show that an inductive w(l)p–semantics for conditioning in non–deterministic probabilistic programs cannot exist.**

I. Introduction

Probabilistic programming is *en vogue* [1], [2]. It is mainstream in machine learning for describing distribution functions; Bayesian inference is pivotal in their analysis. It is used in security for describing both cryptographic constructions such as randomized encryption and experiments defining security notions [3]. Probabilistic programs, being an extension of familiar notions, render these various fields accessible to programming communities. A rich palette of probabilistic programming languages exists including `Church` [4] as well as modern approaches like probabilistic `C` [5], `Tabular` [6] and `R2` [7].

Probabilistic programs are sequential programs having two main features: (1) the ability to draw values at random from probability distributions, and (2) the ability to condition values of variables in a program through observations. The semantics of languages without conditioning is well–understood. Kozen [8] considered denotational semantics, whereas McIver and Morgan [9] provided a weakest (liberal) precondition (w(l)p) semantics; a corresponding operational semantics is given by Gretz *et al*. [10]. Other relevant works include probabilistic power–domains [11], semantics of constraint probabilistic programming languages [12], and semantics for stochastic λ–calculi [13].

Conditioning of variables through observations is less well–understood and raises various semantic difficulties as we will discuss in this paper. Previous work on semantics for programs with `observe` statements [7], [14] do neither consider the possibility of non–termination nor the powerful feature of non–determinism. In this paper, we thoroughly study a more general setting which accounts for non–termination by means of a very simple yet powerful probabilistic programming language supporting non–determinism and observations. Let us first study a few examples that illustrate the semantic intricacies. The sample program snippet P_{obs_1}

$$\{x \coloneqq 0\}\ [1/2]\ \{x \coloneqq 1\};\ \texttt{observe}\ x = 1$$

assigns zero to the variable x with probability $1/2$ while x is assigned one with the same likelihood, after which we condition to the outcome x being one. The `observe` statement blocks all runs violating its condition and prevents those runs from happening. It differs, e.g., from program annotations like (probabilistic) *assertions* [15]. The interpretation of the program is the expected outcome conditioned on permitted runs. For the sample program P_{obs_1} this yields the outcome $1 \cdot 1$—there is one feasible run that happens with probability one with x being one. Whereas this is rather straightforward, a slight variant like P_{obs_2}

$$\{x \coloneqq 0;\ \texttt{observe}\ x = 1\}\ [1/2]\ \{x \coloneqq 1;\ \texttt{observe}\ x = 1\}$$

is somewhat more involved, as the entire left branch of the probabilistic choice is infeasible. Is this program equivalent to the sample program P_{obs_1}?

The situation becomes more intricate when considering loopy programs that may diverge. Consider the programs P_{div} (left) and P_{andiv} (right):

```
$x \coloneqq 1$;
while ($x = 1$) {
    $x \coloneqq 1$
}
```

```
$x \coloneqq 1$;
while ($x = 1$) {
    {$x \coloneqq 1$} [$1/2$] {$x \coloneqq 0$};
    observe $x = 1$
}
```

Program P_{div} diverges and therefore yields as expected outcome zero. Due to the conditioning on x=1, P_{andiv} admits just a single—diverging—feasible run but this run almost surely never happens. Its conditional expected outcome can thus not be measured. It should be noted that programs with (probabilistic) assertions must be loop–free to avoid similar problems [15]. Other approaches insist on the absence of diverging loops [16].

Intricacies also occur when conditioning is used in programs that may abort. Consider the program

$$\{\texttt{abort}\}\ [1/2]\{\{x \coloneqq 0\}\ [1/2]\ \{x \coloneqq 1\};$$

$$\{y \coloneqq 0\}\ [1/2]\ \{y \coloneqq 1\};\ \texttt{observe}\, x{=}0 \vee y{=}0\}$$

where abort is the faulty aborting program which by definition does nothing else but diverge. The above program tosses a fair coin and depending on the outcome either diverges or tosses a fair coin twice. It finally conditions on at least once heads (x=0 or y=0). What is the probability that the outcome of the last coin toss was heads? The main issue here is how to treat the possibility of abortion.

Combining conditioning with non–determinism is complicated, too.[1] Non–determinism is a powerful means to deal with unknown information, as well as to specify abstractions in situations where details are unimportant. Let program P_{nondet} be:

$$\{\{x \coloneqq 5\} \,\Box\, \{x \coloneqq 2\}\}\ [1/4]\ \{x \coloneqq 2\};$$
$$\texttt{observe}\, x > 3$$

where with probability $1/4$, x is set to either 5 or 2 non–deterministically (denoted $\{x \coloneqq 5\} \,\Box\, \{x \coloneqq 2\}$), while x is set to 2 with likelihood $3/4$. Resolving the non–deterministic choice in favour of setting x to five yields an expectation of 5 for x, obtained as $5 \cdot 1/4$ rescaled over the single feasible run of P_{nondet}. Taking the right branch however induces an infeasible run due to the violation of the condition $x > 3$, yielding a non–measurable outcome.

The above issues — loops, divergence, and non–determinism — indicate that conditioning in probabilistic programs is far from trivial. This paper presents a thorough semantic treatment of conditioning in a probabilistic extension of Dijkstra's guarded command language (known as pGCL [9]), an elementary though foundational language that includes (amongst others) parametric probabilistic choice. We take several semantic viewpoints. Reward Markov Decision Processes (RMDPs) [17] are used as the basis for an *operational* semantics. This semantics is rather simple and elegant while covering *all* aforementioned phenomena. In particular, it discriminates the programs P_{div} and P_{andiv} while it does not discriminate P_{obs_1} and P_{obs_2}.

We also provide a *weakest pre–condition* (wp) semantics à la [9]. This is typically defined inductively over the structure of the program. We show that combining both non–determinism and conditioning *cannot* be treated in this manner. Given this impossibility result we present a wp–semantics for fully probabilistic programs, i.e., programs without non–determinism. To treat possibly non–terminating programs, due to e.g., diverging loops or abortion, this is complemented by a weakest *liberal* pre–condition (wlp) semantics. The wlp–semantics yields the weakest pre–expectation — the probabilistic pendant of weakest pre–condition — under which program P either does not terminate or establishes a post–expectation. It thus differs from the wp–semantics in not guaranteeing termination. The *conditional* weakest pre–expectation (cwp) of P with respect to post–expectation f is then given by normalizing $\mathsf{wp}[P](f)$ with respect to $\mathsf{wlp}[P](\mathbf{1})$. The latter yields the wp under which P either does not terminate or terminates while passing all observe statements. This is proven to correspond to conditional expected rewards in the RMDP–semantics, extending a similar result for pGCL [10]. Our semantic viewpoints are thus consistent for fully probabilistic programs. Besides, we show that conditioning is semantically a truly conservative extension. That is to say, our semantics is backward compatible with the (usual) pGCL semantics; this does not apply to alternative approaches such as R2 [7].

Finally, we show several practical applications of our results. We present two program transformations which entirely eliminate conditioning from any program and prove their correctness using the w(l)p–semantics. In addition, we show how the w(l)p–semantics can be used to determine conditional probabilities in a simplified version of the parametric anonymity protocol Crowds [18].

Summarized, we provide the first operational semantics for imperative probabilistic programming languages with conditioning and both probabilistic and non–deterministic choice. Furthermore we give a denotational semantics for the fully probabilistic case, which in contrast to [7], [14], where every program is assumed to terminate almost surely, takes the probability of non–termination into account. Finally, our semantics enables to prove the correctness of several program transformations that eliminate observe statements.

II. Preliminaries

In this section we present the probabilistic programming language used for our approaches and recall the notions of expectation transformers and (conditional) expected reward over Markov decision processes used to endow the language with a formal semantics.

a) Probabilistic programs and expectation transformers: We adopt the *probabilistic guarded command language* (pGCL) [9] for describing probabilistic programs. pGCL is an extension of Dijkstra's guarded command language (GCL) [19] with a binary probabilistic choice operator and its syntax is given by clause

$$\begin{aligned}\mathcal{P} \ ::= \ & \texttt{skip} \mid \texttt{abort} \mid x \coloneqq E \mid \mathcal{P};\mathcal{P} \mid \texttt{ite}\,(G)\,\{\mathcal{P}\}\,\{\mathcal{P}\} \\ & \mid \{\mathcal{P}\}\ [p]\ \{\mathcal{P}\} \mid \{\mathcal{P}\} \,\Box\, \{\mathcal{P}\} \mid \texttt{while}\,(G)\,\{\mathcal{P}\}\ .\end{aligned}$$

Here, x belongs to $\mathcal{V}$, the set of program variables; E is an arithmetical expression over $\mathcal{V}$, G a Boolean expression over $\mathcal{V}$ and p a real–valued parameter with domain [0, 1]. Most of the pGCL instructions are self–explanatory; we elaborate only on the following: $\{P\}\ [p]\ \{Q\}$ represents a *probabilistic choice* where programs P is executed with probability p and program Q with probability $1{-}p$. $\{P\} \,\Box\, \{Q\}$ represents a *non–deterministic* choice between P and Q.

pGCL programs are given a formal semantics through the notion of *expectation transformers*. Let $\mathbb{S}$ be the set of *program states*, where a program state is a variable valuation. Now assume that P is a *fully probabilistic* program, i.e. a program without non–deterministic choices. We can see P as a mapping from an initial state σ to a distribution over final states $[\![P]\!](\sigma)$.

[1] As stated in [2], "representing and inferring sets of distributions is more complicated than dealing with a single distribution, and hence there are several technical challenges in adding non–determinism to probabilistic programs".

Given a random variable $f\colon \mathbb{S} \to \mathbb{R}_{\geq 0}$, transformer $\mathsf{wp}[P]$ maps every initial state σ to the expected value $\mathbf{E}_{[\![P]\!](\sigma)}(f)$ of f with respect to the distribution of final states $[\![P]\!](\sigma)$. Symbolically,

$$\mathsf{wp}[P](f)(\sigma) = \mathbf{E}_{[\![P]\!](\sigma)}(f)\ .$$

In particular, if $f = \chi_A$ is the characteristic function of some event A, $\mathsf{wp}[P](f)$ retrieves the probability that the event occurred after the execution of P. (Moreover, if P is a deterministic program in GCL, $\mathbf{E}_{[\![P]\!](\sigma)}(\chi_A)$ is $\{0, 1\}$–valued and we recover the ordinary notion of weakest pre–condition introduced by Dijkstra [19].)

In contrast to the fully probabilistic case, the execution of a non–deterministic program P may lead to multiple—rather than a single—distributions of final states. To account for these kind of programs, the definition of $\mathsf{wp}[P]$ is extended as follows:

$$\mathsf{wp}[P](f)(\sigma) = \inf_{\mu' \in [\![P]\!](\sigma)} \mathbf{E}_{\mu'}(f)$$

In other words, $\mathsf{wp}[P](f)$ represents the tightest lower bound that can be guaranteed for the expected value of f (we assume that non-deterministic choices are resolved *demonically*[2], attempting to minimize the expected value of f).

In the following, we use the term *expectation* to refer to a random variable mapping program states to real values. The expectation transformer wp then transforms a post–expectation f into a pre–expectation $\mathsf{wp}[P](f)$ and can be defined inductively, following the rules in Figure 2 (second column), Page 7. The transformer wp also admits a liberal variant wlp, which differs from wp on the way in which non–termination is treated.

Formally, the transformer wp operates on *unbounded expectations* in $\mathbb{E} = \mathbb{S} \to \mathbb{R}^{\infty}_{\geq 0}$ and wlp operates on *bounded expectations* in $\mathbb{E}_{\leq 1} = \mathbb{S} \to [0, 1]$. Here $\mathbb{R}^{\infty}_{\geq 0}$ denotes the set of non–negative real values with the adjoined ∞ value. In order to guarantee the well–definedness of wp and wlp we need to provide $\mathbb{E}$ and $\mathbb{E}_{\leq 1}$ the structure of a directed–complete partial order. Expectations are ordered pointwise, i.e. $f \sqsubseteq g$ iff $f(\sigma) \leq g(\sigma)$ for every state $\sigma \in \mathbb{S}$. The least upper bound of directed subsets is also defined pointwise.

In what follows we use bold fonts for constant expectations, e.g. $\mathbf{1}$ denotes the constant expectation 1. Given an arithmetical expression E over program variables we simply write E for the expectation that in state σ returns $\sigma(E)$. Given a Boolean expression G over program variables we use χ_G to denote the $\{0, 1\}$–valued expectation that returns 1 if $\sigma \models G$ and 0 otherwise.

b) MDPs and conditional expected rewards: Let V be a finite set of parameters. A *parametric distribution* over a countable set S is a function $\mu\colon S \to \mathbb{Z}_V$ with $\sum_{s \in S} \mu(s) = 1$, where $\mathbb{Z}_V$ denotes the set of all polynomials[3] over V. $Distr(S)$ denotes the set of parametric distributions over S.

Definition II.1 (Parametric Discrete–time Reward Markov Decision Process)**.** Let AP be a set of atomic propositions. A *parametric discrete–time reward Markov decision process (RMDP)* is a tuple $\mathfrak{R} = (S,\, s_I,\, Act,\, \mathcal{P},\, L,\, r)$ with a countable set of states S, a unique initial state $s_I \in S$, a finite set of actions Act, a transition probability function $\mathcal{P}\colon S \times Act \to Distr(S)$ with $\forall (s, \alpha) \in S \times Act\bullet\ \sum_{s' \in S} \mathcal{P}(s, \alpha)(s') = 1$, a labeling function $L\colon S \to 2^{AP}$, and a reward function $r\colon S \to \mathbb{R}_{\geq 0}$.

A *path* of $\mathfrak{R}$ is a finite or infinite sequence $\pi = s_0\alpha_0 s_1\alpha_1 \ldots$ such that $s_i \in S$, $\alpha_i \in Act$, $s_0 = s_I$, and $\mathcal{P}(s_i,\, \alpha_i)(s_{i+1}) > 0$ for all $i \geq 0$. A finite path is denoted by $\hat{\pi} = s_0\alpha_0 \ldots s_n$ for $n \in \mathbb{N}$ with $last(\hat{\pi}) = s_n$ and $|\pi| = n$. The i-th state s_i of π is denoted $\pi(i)$. The set of all paths of $\mathfrak{R}$ is denoted by $\mathsf{Paths}^{\mathfrak{R}}$ and sets of infinite or finite paths by $\mathsf{Paths}^{\mathfrak{R}}_{inf}$ or $\mathsf{Paths}^{\mathfrak{R}}_{fin}$, respectively. $\mathsf{Paths}^{\mathfrak{R}}(s)$ is the set of paths starting in s and $\mathsf{Paths}^{\mathfrak{R}}(s, s')$ is the set of all finite paths starting in s and ending in s'. This is also lifted to sets of states. If clear from the context we omit the superscript $\mathfrak{R}$.

An MDP operates by a non–deterministic choice of an action $\alpha \in Act$ that is *enabled* at state s and a subsequent probabilistic determination of a successor state according to $\mathcal{P}(s, \alpha)$. We denote the set of actions that are enabled at s by $Act(s)$ and assume that $Act(s) \neq \emptyset$ for each state s. A state s with $|Act(s)| = 1$ is called *fully probabilistic*, and in this case we use $\mathcal{P}(s,\, s')$ as a shorthand for $\mathcal{P}(s,\, \alpha)(s')$ where $Act(s) = \{\alpha\}$. For resolving the non–deterministic choices, so–called *schedulers* are used. In our setting, *deterministic* schedulers suffice, which are partial functions $\mathfrak{S}\colon \mathsf{Paths}^{\mathfrak{R}}_{fin} \to Act$ with $\mathfrak{S}(\hat{\pi}) \in Act(last(\hat{\pi}))$. A deterministic scheduler is called *memoryless* if the choice depends only on the current state, yielding a function $\mathfrak{S}\colon S \to Act$. The class of all (deterministic) schedulers for $\mathfrak{R}$ is denoted by $Sched^{\mathfrak{R}}$.

A *parametric discrete–time reward Markov chain (RMC)* is an RMDP with only fully probabilistic states. For an RMC we use the notation $\mathcal{R} = (S,\, s_I,\, \mathcal{P},\, L,\, r)$ where $\mathcal{P}\colon S \to Distr(S)$ is called a *transition probability matrix*. For RMDP $\mathfrak{R}$, the fully probabilistic system ${}^{\mathfrak{S}}\mathfrak{R}$ induced by a scheduler $\mathfrak{S} \in Sched^{\mathfrak{R}}$ is an *induced RMC*. A *probability measure* is defined on the induced RMCs. The measure for RMC $\mathcal{R}$ is given by $\Pr^{\mathcal{R}}\colon \mathsf{Paths}^{\mathcal{R}}_{fin} \to [0, 1] \subseteq \mathbb{R}$ with $\Pr^{\mathcal{R}}(\hat{\pi}) = \prod_{i=0}^{n-1} \mathcal{P}(s_i, s_{i+1})$, for $\hat{\pi} = s_0 \ldots s_n$. The probability measure can be lifted to sets of (infinite) paths using a cylinder set construction, see [21, Ch. 10]. The *cumulated reward* of a finite path $\hat{\pi} = s_0 \ldots s_n$ is given by $r(\hat{\pi}) = \sum_{i=0}^{n-1} r(s_i)$ as the reward is "earned" when leaving the state.

We consider *reachability properties* of the form $\lozenge T$ for a set of target states $T = \{s \in S \mid T \in L(s)\}$ where T is overloaded to be a set of states and a label in AP. The set $\lozenge T = \{\pi \in \mathsf{Paths}(s_I, T) \mid \forall 0 \leq i < |\pi|\bullet\ \pi(i) \notin T\}$ shall be prefix–free and contain all paths of $\mathcal{R}$ that visit a target state. Analogously, the set $\neg\lozenge T = \{\pi \in \mathsf{Paths}^{\mathcal{R}}(s_I) \mid \forall i \geq 0\bullet\ \pi(i) \notin T\}$ contains all paths that never reach a state in T. Let us first consider reward objectives for fully probabilistic models, i.e., RMCs. The *expected reward* for a finite set of

[2] Demonic schedulers induce the most pessimistic expected outcome while in [20] also *angelic schedulers* are considered which guarantee the most optimistic outcome.

[3] Although parametric distributions are defined as polynomials over the parameters, we only use p and $1 - p$ for $p \in V$

paths $\Diamond T \in \mathsf{Paths}^{\mathcal{R}}_{\mathit{fin}}$ is

$$\mathsf{ExpRew}^{\mathcal{R}}(\Diamond T) \;\triangleq\; \sum_{\hat{\pi} \in \Diamond T} \Pr^{\mathcal{R}}(\hat{\pi}) \cdot r(\hat{\pi}) \ .$$

For a reward bounded by one, the notion of the *liberal* expected reward also takes the mere probability of *not* reaching the target states into account:

$$\mathsf{LExpRew}^{\mathcal{R}}(\Diamond T) \;\triangleq\; \mathsf{ExpRew}^{\mathcal{R}}(\Diamond T) + \Pr^{\mathcal{R}}(\neg\Diamond T)$$

A liberal expected reward will later represent the probability of either establishing some condition or not terminating.

To explicitly exclude the probability of paths that reach "undesired" states, we let $U = \{s \in S \mid \lightning \in L(s)\}$ and define the *conditional expected reward* for the condition $\neg\Diamond U$ by[4]

$$\mathsf{CExpRew}^{\mathcal{R}}(\Diamond T \mid \neg\Diamond U) \;\triangleq\; \frac{\mathsf{ExpRew}^{\mathcal{R}}(\Diamond T \cap \neg\Diamond U)}{\Pr^{\mathcal{R}}(\neg\Diamond U)} \ .$$

For details about conditional probabilities and expected rewards, we refer to [22]. Conditional *liberal* expected rewards are defined by

$$\mathsf{CLExpRew}^{\mathcal{R}}(\Diamond T \mid \neg\Diamond U) \;\triangleq\; \frac{\mathsf{LExpRew}^{\mathcal{R}}(\Diamond T \cap \neg\Diamond U)}{\Pr^{\mathcal{R}}(\neg\Diamond U)} \ .$$

Reward objectives for RMDPs are now defined using a *demonic* scheduler $\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}}$ minimizing probabilities and expected rewards for the induced RMC ${}^{\mathfrak{S}}\mathcal{R}$. For the expected reward this yields

$$\mathsf{ExpRew}^{\mathfrak{R}}(\Diamond T) \;\triangleq\; \inf_{\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}}} \mathsf{ExpRew}^{{}^{\mathfrak{S}}\mathcal{R}}(\Diamond T) \ .$$

The scheduler for conditional expected reward properties minimizes the value of the quotient:

$$\begin{aligned} & \mathsf{CExpRew}^{\mathfrak{R}}(\Diamond T \mid \neg\Diamond U) \\ \triangleq\;& \inf_{\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}}} \mathsf{CExpRew}^{{}^{\mathfrak{S}}\mathcal{R}}(\Diamond T \mid \neg\Diamond U) \\ =\;& \inf_{\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}}} \frac{\mathsf{ExpRew}^{{}^{\mathfrak{S}}\mathcal{R}}(\Diamond T \cap \neg\Diamond U)}{\Pr^{{}^{\mathfrak{S}}\mathcal{R}}(\neg\Diamond U)} \end{aligned}$$

The liberal reward notions for RMDPS are analogous. Regarding the quotient minimization we assume "$\frac{0}{0} < 0$" as we see $\frac{0}{0}$—being undefined—to be less favorable than 0.

III. Conditional pGCL

As mentioned in Section II, pGCL programs can be considered as distribution transformers. Inspired by [2], we extend pGCL with `observe` statements to obtain *conditional* pGCL (cpGCL, for short). This is done by extending the syntax of pGCL (p. 2) with `observe` G where G is a Boolean expression over the program variables. When a program's execution reaches `observe` G with a current variable valuation $\sigma \not\models G$, further execution of the program is blocked as with an `assert` statement [23]. In contrast to `assert`, however, the `observe` statements do not only block further execution but *condition* resulting distributions on the program's state to only those executions satisfying the observations. Consider two small example programs:

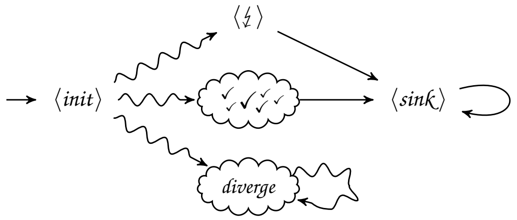
$$\begin{array}{ll} \{x := 0\}\ [p]\ \{x := 1\}; & \{x := 0\}\ [p]\ \{x := 1\}; \\ \{y := 0\}\ [q]\ \{y := -1\} & \{y := 0\}\ [q]\ \{y := -1\}; \\ & \texttt{observe}\ x + y = 0 \end{array}$$

The left program establishes that the probability of $x{=}0$ is p, whereas for the right program this probability is $\frac{pq}{pq+(1-p)(1-q)}$. The left program admits all (four) runs, two of which satisfy $x{=}0$. Due to the `observe` statement requiring $x{+}y{=}0$, the right program, however, admits only two runs ($x{=}0, y{=}0$ and $x{=}1, y{=}{-1}$), satisfying $x{=}0$.

In Section V we will focus on the subclass of fully probabilistic programs in cpGCL, which we denote $\mathsf{cpGCL}^{\boxtimes}$.

IV. Operational Semantics for cpGCL

This section presents an operational semantics for cpGCL using RMDPs as underlying model inspired by [10]. Schematically, the operational RMDP of a cpGCL program shall have the following structure:



Terminating runs eventually end up in the $\langle \mathit{sink} \rangle$ state; other runs are diverging (never reach $\langle \mathit{sink} \rangle$). A program terminates either successfully, i.e. a run passes a $\checkmark$–labelled state, or terminates due to a false observation, i.e. a run passes $\langle \lightning \rangle$. Squiggly arrows indicate reaching certain states via possibly multiple paths and states; the clouds indicate that there might be several states of the particular kind. The $\checkmark$–labelled states are the *only ones* with positive reward. Note that the sets of paths that eventually reach $\langle \lightning \rangle$, eventually reach $\checkmark$, or diverge, are pairwise disjoint.

Definition IV.1 (Operational cpGCL semantics)**.** The *operational semantics* of $P \in \mathsf{cpGCL}$ for $\sigma \in \mathbb{S}$ and $f \in \mathbb{E}$ is the RMDP $\mathfrak{R}^f_\sigma[\![P]\!] = (S, \langle P, \sigma\rangle, \mathit{Act}, \mathcal{P}, L, r)$, such that S is the smallest set of states with $\langle \lightning \rangle \in S$, $\langle \mathit{sink} \rangle \in S$, and $\langle Q, \tau\rangle, \langle \downarrow, \tau\rangle \in S$ for $Q \in \mathsf{pGCL}$ and $\tau \in \mathbb{S}$. $\langle P, \sigma\rangle \in S$ is the initial state. $\mathit{Act} = \{\mathit{left}, \mathit{right}\}$ is the set of actions. $\mathcal{P}$ is formed according to the rules given in Figure 1. The labelling and the reward function are given by:

$$L(s) \;\triangleq\; \begin{cases} \{\checkmark\}, & \text{if } s = \langle \downarrow, \tau\rangle, \text{ for some } \tau \in \mathbb{S} \\ \{\mathit{sink}\}, & \text{if } s = \langle \mathit{sink} \rangle \\ \{\lightning\}, & \text{if } s = \langle \lightning \rangle \\ \emptyset, & \text{otherwise,} \end{cases}$$

[4] Note that strictly formal one would have to define the intersection of sets of finite and possibly infinite paths by means of a cylinder set construction considering all infinite extensions of finite paths.

$$\textbf{(terminal)}\ \frac{}{\langle \downarrow,\, \sigma\rangle \ \longrightarrow\ \langle \mathit{sink}\rangle} \qquad \textbf{(skip)}\ \frac{}{\langle \texttt{skip},\, \sigma\rangle \ \longrightarrow\ \langle \downarrow,\, \sigma\rangle} \qquad \textbf{(abort)}\ \frac{}{\langle \texttt{abort},\, \sigma\rangle \ \longrightarrow\ \langle \texttt{abort},\, \sigma\rangle} \qquad \textbf{(undesired)}\ \frac{}{\langle \lightning \rangle \ \longrightarrow\ \langle \mathit{sink}\rangle}$$

$$\textbf{(assign)}\ \frac{}{\langle x := E,\, \sigma\rangle \ \longrightarrow\ \langle \downarrow,\, \sigma[x \leftarrow [\![E]\!]_\sigma]\rangle} \qquad \textbf{(observe)}\ \frac{\sigma \models G}{\langle \texttt{observe}\, G,\, \sigma\rangle \ \longrightarrow\ \langle \downarrow,\, \sigma\rangle} \qquad \frac{\sigma \not\models G}{\langle \texttt{observe}\, G,\, \sigma\rangle \ \longrightarrow\ \langle \lightning \rangle}$$

$$\textbf{(concatenate)}\ \frac{}{\langle \downarrow; Q,\, \sigma\rangle \ \longrightarrow\ \langle Q,\, \sigma\rangle} \qquad \frac{\langle P,\, \sigma\rangle \ \longrightarrow\ \langle \lightning \rangle}{\langle P; Q,\, \sigma\rangle \ \longrightarrow\ \langle \lightning \rangle} \qquad \frac{\langle P,\, \sigma\rangle \ \longrightarrow\ \mu}{\langle P; Q,\, \sigma\rangle \ \longrightarrow\ \nu},\ \text{where } \forall P'.\, \nu(\langle P'; Q, \sigma'\rangle) := \mu(\langle P',\, \sigma'\rangle)$$

$$\textbf{(if)}\ \frac{\sigma \models G}{\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle \ \longrightarrow\ \langle P,\, \sigma\rangle} \qquad \frac{\sigma \not\models G}{\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle \ \longrightarrow\ \langle Q,\, \sigma\rangle}$$

$$\textbf{(while)}\ \frac{\sigma \models G}{\langle \texttt{while}\,(G)\,\{P\},\, \sigma\rangle \ \longrightarrow\ \langle P; \texttt{while}\,(G)\,\{P\},\, \sigma\rangle} \qquad \frac{\sigma \not\models G}{\langle \texttt{while}\,(G)\,\{P\},\, \sigma\rangle \ \longrightarrow\ \langle \downarrow,\, \sigma\rangle}$$

$$\textbf{(prob. choice)}\ \frac{}{\langle \{P\}\ [p]\ \{Q\},\, \sigma\rangle \ \longrightarrow\ \nu},\ \text{where } \nu(\langle P,\, \sigma\rangle) := p,\ \nu(\langle Q,\, \sigma\rangle) := 1 - p$$

$$\textbf{(non–det. choice)}\ \frac{}{\langle \{P\}\,\square\,\{Q\},\, \sigma\rangle \ \xrightarrow{\mathit{left}}\ \langle P,\, \sigma\rangle} \qquad \frac{}{\langle \{P\}\,\square\,\{Q\},\, \sigma\rangle \ \xrightarrow{\mathit{right}}\ \langle Q,\, \sigma\rangle}$$

Fig. 1. Rules for the construction of the operational RMDPs. If not stated otherwise, $\langle s\rangle \longrightarrow \langle t\rangle$ is a shorthand for $\langle s\rangle \longrightarrow \mu \in \mathit{Distr}(\mathbb{S})$ with $\mu(\langle t\rangle) = 1$. A terminal state of the form $\langle \downarrow,\, \sigma\rangle$ indicates successful termination. Terminal states and $\langle \lightning \rangle$ go to the $\langle \mathit{sink}\rangle$ state. skip without context terminates successfully. abort self–loops, i.e. diverges. $x := E$ alters the variable valuation according to the assignment then terminates successfully. For the concatenation, $\langle \downarrow; Q,\, \sigma\rangle$ indicates successful termination of the first program, so the execution continues with $\langle Q,\, \sigma\rangle$. If for P; Q the execution of P leads to $\langle \lightning \rangle$, P; Q does so, too. Otherwise, for $\langle P, \sigma\rangle \longrightarrow \mu$, μ is lifted such that Q is concatenated to the support of μ. If for the conditional choice $\sigma \models G$ holds, P is executed, otherwise Q. The case for while is similar. For the probabilistic choice, a distribution ν is created according to p. For $\{P\}\,\square\,\{Q\}$, we call P the *left* choice and Q the *right* choice for actions $\mathit{left}, \mathit{right} \in \mathit{Act}$. For the observe statement, if $\sigma \models G$ observe acts like skip. Otherwise, the execution leads directly to $\langle \lightning \rangle$ indicating a violation of the observe statement.

$$r(s) \ \triangleq\ \begin{cases} f(\tau), & \text{if } s = \langle \downarrow,\, \tau\rangle, \ \text{for some } \tau \in \mathbb{S} \\ 0, & \text{otherwise} \end{cases}$$

where a state of the form $\langle \downarrow,\, \tau\rangle$ denotes a terminal state in which no program is left to be executed.

To determine the *conditional expected outcome of program* P given that all observations are true, we need to determine the *expected reward to reach* $\langle \mathit{sink}\rangle$ *from the initial state conditioned on not reaching* $\langle \lightning \rangle$ under a demonic scheduler. For $\mathfrak{R}^f_\sigma[\![P]\!]$ this is given by $\mathsf{CExpRew}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\lozenge \mathit{sink} \mid \neg\lozenge \lightning)$. Recall for the condition $\neg\lozenge \lightning$ that all paths not eventually reaching $\langle \lightning \rangle$ either diverge (thus collect reward 0) or pass by a $\checkmark$–labelled state and eventually reach $\langle \mathit{sink}\rangle$. This gives us:

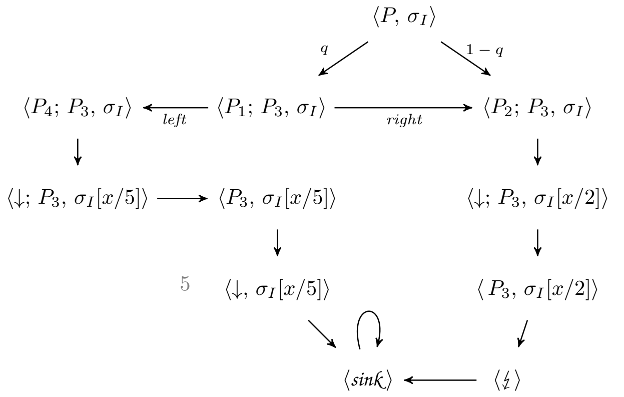
$$\begin{aligned} &\mathsf{CExpRew}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\lozenge \mathit{sink} \mid \neg\lozenge \lightning) \\ =\ &\inf_{\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}^f_\sigma[\![P]\!]}} \frac{\mathsf{ExpRew}^{\mathfrak{S}}{}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\lozenge \mathit{sink} \cap \neg\lozenge \lightning)}{\Pr^{\mathfrak{S}}{}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\neg\lozenge \lightning)} \\ =\ &\inf_{\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}^f_\sigma[\![P]\!]}} \frac{\mathsf{ExpRew}^{\mathfrak{S}}{}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\lozenge \mathit{sink})}{\Pr^{\mathfrak{S}}{}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\neg\lozenge \lightning)} \end{aligned}$$

This is analogous for $\mathsf{CLExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\lozenge \mathit{sink} \mid \neg\lozenge \lightning)$.

Example IV.1. Consider the program $P \in \mathsf{cpGCL}$:

$$\begin{aligned} &\{\{x := 5\}\,\square\,\{x := 2\}\}\ [q]\ \{x := 2\}; \\ &\texttt{observe}\, x > 3 \end{aligned}$$

where with parametrized probability q a non–deterministic choice between x being assigned 2 or 5 is executed, and with probability $1 - q$, x is directly assigned 2. Let for readability $P_1 = \{x := 5\}\,\square\,\{x := 2\}$, $P_2 = x := 2$, $P_3 = \texttt{observe}\, x > 3$, and $P_4 = x := 5$. The operational RMDP $\mathfrak{R}^x_{\sigma_I}[\![P]\!]$ for an arbitrary initial variable valuation σ_I and post–expectation x is depicted below.


$\langle P,\, \sigma_I\rangle$
q
$1 - q$
$\langle P_4;\, P_3,\, \sigma_I\rangle$
left
$\langle P_1;\, P_3,\, \sigma_I\rangle$
right
$\langle P_2;\, P_3,\, \sigma_I\rangle$
$\langle \downarrow;\, P_3,\, \sigma_I[x/5]\rangle$
$\langle P_3,\, \sigma_I[x/5]\rangle$
$\langle \downarrow;\, P_3,\, \sigma_I[x/2]\rangle$
5
$\langle \downarrow,\, \sigma_I[x/5]\rangle$
$\langle P_3,\, \sigma_I[x/2]\rangle$
$\langle \mathit{sink}\rangle$
$\langle \lightning \rangle$

The only state with positive reward is $s' := \langle \downarrow,\, \sigma_I[x/5]\rangle$ and its reward is indicated by number 5. Assume first a scheduler choosing action *left* in state $\langle P_1;\, P_3,\, \sigma_I\rangle$. In the induced RMC the only path accumulating positive reward is the path π going from $\langle P,\, \sigma_I\rangle$ via s' to $\langle \mathit{sink}\rangle$ with $r(\pi) = 5$ and $\Pr(\pi) = q$. This gives an expected reward of $5 \cdot q$. The overall probability of not reaching $\langle \lightning \rangle$ is also q. The conditional expected reward of eventually reaching $\langle \mathit{sink}\rangle$ given that $\langle \lightning \rangle$ is not reached is hence $\frac{5 \cdot q}{q} = 5$. Assume now the *minimizing* scheduler choosing right at state $\langle P_1;\, P_3,\, \sigma_I\rangle$. In this case there is no path having positive accumulated reward in the induced RMC, yielding an expected reward of 0. The probability of not reaching $\langle \lightning \rangle$ is also 0. The conditional expected reward in this case is undefined ($^0/_0$) thus the right branch is preferred over the *left* branch.

In general, the operational RMDP is not finite, even if the program terminates almost–surely (i.e. with probability 1).

V. Denotational Semantics for $\mathsf{cpGCL}^{\boxtimes}$

This section presents an expectation transformer semantics for the fully probabilistic fragment $\mathsf{cpGCL}^{\boxtimes}$ of cpGCL. We formally relate this to the wp/wlp–semantics of pGCL as well as to the operational semantics from the previous section.

A. Conditional Expectation Transformers

An expectation transformer semantics for the fully probabilistic fragment of cpGCL is defined using the operators:

$$\mathsf{cwp}[\,\cdot\,]\colon \mathbb{E}\times\mathbb{E}_{\leq 1} \to \mathbb{E}\times\mathbb{E}_{\leq 1}$$
$$\mathsf{cwlp}[\,\cdot\,]\colon \mathbb{E}_{\leq 1}\times\mathbb{E}_{\leq 1} \to \mathbb{E}_{\leq 1}\times\mathbb{E}_{\leq 1}$$

These functions can intuitively be viewed as the counterpart of wp and wlp respectively, as shortly shown. The weakest conditional pre–expectation $\underline{\mathsf{cw}}\mathsf{p}[P](f)$ of $P \in \mathsf{cpGCL}^{\boxtimes}$ with respect to post–expectation f is now given as

$$\underline{\mathsf{cw}}\mathsf{p}[P](f) \;\triangleq\; \frac{\mathsf{cwp}_1[P](f,\mathbf{1})}{\mathsf{cwp}_2[P](f,\mathbf{1})}\ ,$$

where $\mathsf{cwp}_1[P](f,g)$ (resp. $\mathsf{cwp}_2[P](f,g)$) denotes the first (resp. second) component of $\mathsf{cwp}[P](f,g)$ and $\mathbf{1}$ is the constant expectation one. The weakest liberal conditional pre–expectation $\underline{\mathsf{cwl}}\mathsf{p}[P](f)$ is defined analogously. In words, $\underline{\mathsf{cw}}\mathsf{p}[P](f)(\sigma)$ represents the expected value of f with respect to the distribution of final states obtained from executing P in state σ, given that all `observe` statements occurring along the runs of P were satisfied. The quotient defining $\underline{\mathsf{cw}}\mathsf{p}[P](f)$ is interpreted is the same way as the quotient

$$\frac{\Pr(A\cap B)}{\Pr(B)}$$

encoding the conditional probability $\Pr(A|B)$. However, here we measure the expected value of random variable f[5]. The denominator $\mathsf{cwp}_2[P](f,\mathbf{1})(\sigma)$ measures the probability that P satisfies all the observations (occurring along valid runs) from the initial state σ. If $\mathsf{cwp}_2[P](f,\mathbf{1})(\sigma)=0$, program P is *infeasible* from state σ and in this case $\underline{\mathsf{cw}}\mathsf{p}[P](f)(s)$ is not well–defined (due to the division by zero). This corresponds to the conditional probability $\Pr(A|B)$ being not well–defined when $\Pr(B)=0$.

The operators cwp and cwlp are defined inductively on the program structure, see Figure 2 (last column). Let us briefly explain this. $\mathsf{cwp}[\texttt{skip}]$ behaves as the identity since `skip` has no effect on the program state. $\mathsf{cwp}[\texttt{abort}]$ maps any pair of post–expectations to the pair of constant pre–expectations $(\mathbf{0},\mathbf{1})$. Assignments induce a substitution on expectations, i.e. $\mathsf{cwp}[x := E]$ maps (f,g) to pre–expectation $(f[x/E], g[x/E])$, where $h[x/E](\sigma) = h(\sigma[x/E])$ and $\sigma[x/E]$ denotes the usual variable update on states. $\mathsf{cwp}[P_1;P_2]$ is obtained as the functional composition (denoted $\circ$) of $\mathsf{cwp}[P_1]$ and $\mathsf{cwp}[P_2]$. $\mathsf{cwp}[\texttt{observe}\,G]$ restricts post–expectations to those states that satisfy G; states that do not satisfy G are mapped to 0. $\mathsf{cwp}[\texttt{ite}\,(G)\,\{P_1\}\,\{P_2\}]$ behaves either as $\mathsf{cwp}[P_1]$ or $\mathsf{cwp}[P_2]$ according to the evaluation of G. $\mathsf{cwp}[\{P_1\}\ [p]\ \{P_2\}]$ is obtained as a convex combination of $\mathsf{cwp}[P_1]$ and $\mathsf{cwp}[P_2]$, weighted according to p. $\mathsf{cwp}[\texttt{while}\,(G)\,\{P'\}]$ is defined using standard fixed point techniques.[6]

The cwlp transformer follows the same rules as cwp, except for the `abort` and `while` statements. $\mathsf{cwlp}[\texttt{abort}]$ takes any post–expectation to pre–expectation $(\mathbf{1},\mathbf{1})$ and $\mathsf{cwlp}[\texttt{while}\,(G)\,\{P\}]$ is defined as a *greatest* fixed point rather than a least fixed point.

Example V.1. Consider the program P'

```
1  $\{x := 0\}\ [1/2]\ \{x := 1\};$
2  ite $(x = 1)\ \{\{y := 0\}\ [1/2]\ \{y := 2\}\}$
        $\{\{y := 0\}\ [4/5]\ \{y := 3\}\};$
3  observe $y = 0$
```

Assume we want to compute the conditional expected value of the expression $10{+}x$ given that the observation $y{=}0$ is passed. This expected value is given by $\underline{\mathsf{cw}}\mathsf{p}[P'](10{+}x)$ and the computation of $\mathsf{cwp}[P'](10{+}x,\mathbf{1})$ goes as follows:

$$\begin{aligned}
&\mathsf{cwp}[P'](10{+}x,\mathbf{1})\\
&\quad= \mathsf{cwp}[P'_{1\text{-}2}](\mathsf{cwp}[\texttt{observe}\,y=0](10{+}x,\mathbf{1}))\\
&\quad= \mathsf{cwp}[P'_{1\text{-}2}](f,g) \text{ where } (f,g) = \chi_{y=0}\cdot(10{+}x,\mathbf{1})\\
&\quad= \mathsf{cwp}[P'_{1\text{-}1}](\mathsf{cwp}[\texttt{ite}\,(x{=}1)\,\{\ldots\}\,\{\ldots\}](f,g))\\
&\quad= \mathsf{cwp}[P'_{1\text{-}1}](\chi_{x=1}\cdot(h,i) + \chi_{x\neq 1}\cdot(h',i'))\ \text{ where}\\
&\qquad\quad (h,i) = \mathsf{cwp}[\{y{:=}0\}\ [1/2]\ \{y{:=}2\}](f,g)\\
&\qquad\qquad\quad = \tfrac{1}{2}\cdot(10+x,\,\mathbf{1})\ ,\ \text{and}\\
&\qquad\quad (h',i') = \mathsf{cwp}[\{y{:=}0\}\ [4/5]\ \{y{:=}3\}](f,g)\\
&\qquad\qquad\quad = \tfrac{4}{5}\cdot(10+x,\,\mathbf{1})\\
&\quad= \tfrac{1}{2}\cdot\tfrac{4}{5}\cdot(\mathbf{10}+0,\,\mathbf{1}) + \tfrac{1}{2}\cdot\tfrac{1}{2}\cdot(\mathbf{10}+1,\,\mathbf{1})\\
&\quad= \big(\mathbf{4},\,\tfrac{\mathbf{2}}{\mathbf{5}}\big) + \big(\tfrac{\mathbf{11}}{\mathbf{4}},\,\tfrac{\mathbf{1}}{\mathbf{4}}\big) = \big(\tfrac{\mathbf{27}}{\mathbf{4}},\,\tfrac{\mathbf{13}}{\mathbf{20}}\big)
\end{aligned}$$

Then $\underline{\mathsf{cw}}\mathsf{p}[P'](10{+}x) = \frac{\mathbf{135}}{\mathbf{13}}$ and the conditional expected value of $10{+}x$ is approximately 10.38.

In the rest of this section we investigate some properties of the expectation transformer semantics of $\mathsf{cpGCL}^{\boxtimes}$. As every fully probabilistic pGCL program is contained in $\mathsf{cpGCL}^{\boxtimes}$, we first study the relation of the cw(l)p– to the w(l)p–semantics of pGCL. To that end, we extend the weakest (liberal) pre–expectation operator to cpGCL as follows:

$$\mathsf{wp}[\texttt{observe}\,G](f) = \chi_G{\cdot}f \qquad \mathsf{wlp}[\texttt{observe}\,G](f) = \chi_G{\cdot}f\ .$$

To relate the cw(l)p– and w(l)p–semantics we heavily rely on the following result which says that cwp (resp. cwlp) can be *decoupled* as the product wp × wlp (resp. wlp × wlp).

Theorem V.1 (Decoupling of cw(l)p)**.** *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $f \in \mathbb{E}$, and $f', g \in \mathbb{E}_{\leq 1}$:*

$$\begin{aligned}
\mathsf{cwp}[P](f,\,g) &= \big(\mathsf{wp}[P](f),\,\mathsf{wlp}[P](g)\big)\\
\mathsf{cwlp}[P](f',\,g) &= \big(\mathsf{wlp}[P](f),\,\mathsf{wlp}[P](g)\big)
\end{aligned}$$

[5] In fact, $\underline{\mathsf{cw}}\mathsf{p}[P](f)(\sigma)$ corresponds to the notion of *conditional expected value* or in simpler terms, the expected value over a conditional distribution.

[6] We define $\mathsf{cwp}[\texttt{while}\,(G)\,\{P\}]$ by the least fixed point w.r.t. the order $(\sqsubseteq,\sqsupseteq)$ in $\mathbb{E}\times\mathbb{E}_{\leq 1}$. This way we encode the greatest fixed point in the second component w.r.t. the order $\sqsubseteq$ over $\mathbb{E}_{\leq 1}$ as the least fixed point w.r.t. the dual order $\sqsupseteq$.

$\boldsymbol{P}$	$\mathsf{\mathbf{wp}}[\boldsymbol{P}](\boldsymbol{f})$	$\mathsf{\mathbf{cwp}}[\boldsymbol{P}](\boldsymbol{f},\,\boldsymbol{g})$
$\mathtt{skip}$	f	$(f,\, g)$
$\mathtt{abort}$	$\mathbf{0}$	$(\mathbf{0},\, \mathbf{1})$
$x \coloneqq E$	$f[x/E]$	$(f[x/E],\, g[x/E])$
$\mathtt{observe}\, G$	$\chi_G \cdot f$	$\chi_G \cdot (f,\, g)$
$P_1;\ P_2$	$(\mathsf{wp}[P_1] \circ \mathsf{wp}[P_2])(f)$	$(\mathsf{cwp}[P_1] \circ \mathsf{cwp}[P_2])(f, g)$
$\mathtt{ite}\,(G)\,\{P_1\}\,\{P_2\}$	$\chi_G \cdot \mathsf{wp}[P_1](f) + \chi_{\neg G} \cdot \mathsf{wp}[P_2](f)$	$\chi_G \cdot \mathsf{cwp}[P_1](f,\, g) + \chi_{\neg G} \cdot \mathsf{cwp}[P_2](f,\, g)$
$\{P_1\}\ [p]\ \{P_2\}$	$p \cdot \mathsf{wp}[P_1](f) + (1-p) \cdot \mathsf{wp}[P_2](f)$	$p \cdot \mathsf{cwp}[P_1](f,\, g) + (1-p) \cdot \mathsf{cwp}[P_2](f,\, g)$
$\{P_1\}\,\Box\,\{P_2\}$	$\lambda\sigma\text{.}\ \min\{\mathsf{wp}[P_1](f)(\sigma),\, \mathsf{wp}[P_2](f)(\sigma)\}$	— not defined —
$\mathtt{while}\,(G)\,\{P'\}$	$\boldsymbol{\mu}\, \hat{f}\text{.}\ \Big(\chi_G \cdot \mathsf{wp}[P'](\hat{f}) + \chi_{\neg G} \cdot f\Big)$	$\boldsymbol{\mu}_{\sqsubseteq,\sqsupseteq}(\hat{f},\, \hat{g})\text{.}\ \Big(\chi_G \cdot \mathsf{cwp}[P'](\hat{f},\, \hat{g}) + \chi_{\neg G} \cdot (f,\, g)\Big)$

$\boldsymbol{P}$	$\mathsf{\mathbf{wlp}}[\boldsymbol{P}](\boldsymbol{f})$	$\mathsf{\mathbf{cwlp}}[\boldsymbol{P}](\boldsymbol{f},\,\boldsymbol{g})$
$\mathtt{abort}$	$\mathbf{1}$	$(\mathbf{1},\, \mathbf{1})$
$\mathtt{while}\,(G)\,\{P'\}$	$\boldsymbol{\nu}\, \hat{f}\text{.}\ \Big(\chi_G \cdot \mathsf{wp}[P'](\hat{f}) + \chi_{\neg G} \cdot f\Big)$	$\boldsymbol{\nu}_{\sqsubseteq,\sqsubseteq}(\hat{f},\, \hat{g})\text{.}\ \Big(\chi_G \cdot \mathsf{cwp}[P'](\hat{f},\, \hat{g}) + \chi_{\neg G} \cdot (f,\, g)\Big)$

Fig. 2. Definitions for the wp/wlp and $\mathsf{cwp}/\mathsf{cwlp}$ operators. The wlp (cwlp) operator differs from wp (cwp) only for $\mathtt{abort}$ and the $\mathtt{while}$–loop. A scalar multiplication $\alpha \cdot (f,\, g)$ is meant componentwise yielding $(\alpha \cdot f,\, \alpha \cdot g)$. Likewise an addition $(f,\, g) + (f',\, g')$ is also meant componentwise yielding $(f + f',\, g + g')$.

Proof. By induction on the program structure. See Appendix B for details. □

Let $\mathsf{pGCL}^{\boxtimes}$ denote the fully probabilistic fragment of pGCL. We show that the $\underline{\mathsf{cw}}\mathsf{p}$–semantics is a *conservative extension* of the wp–semantics for $\mathsf{pGCL}^{\boxtimes}$. The same applies to the weakest liberal pre–expectation semantics.

Theorem V.2 (Compatibility with the $\mathsf{w(l)p}$–semantics)**.** *For $P \in \mathsf{pGCL}^{\boxtimes}$, $f \in \mathbb{E}$, and $g \in \mathbb{E}_{\leq 1}$:*

$$\mathsf{wp}[P](f) = \underline{\mathsf{cw}}\mathsf{p}[P](f) \quad \textit{and} \quad \mathsf{wlp}[P](g) = \underline{\mathsf{cwl}}\mathsf{p}[P](g)$$

Proof. By Theorem V.1 and the fact that $\underline{\mathsf{cwl}}\mathsf{p}[P](\mathbf{1}) = \mathbf{1}$ (see Lemma V.3). □

We now investigate some elementary properties of $\underline{\mathsf{cw}}\mathsf{p}$ and $\underline{\mathsf{cwl}}\mathsf{p}$ such as monotonicity and linearity.

Lemma V.3 (Elementary properties of $\underline{\mathsf{cw}}\mathsf{p}$ and $\underline{\mathsf{cwl}}\mathsf{p}$)**.** *For every $P \in \mathsf{cpGCL}^{\boxtimes}$ with at least one feasible execution (from every initial state), post–expectations $f, g \in \mathbb{E}$ and non–negative real constants α, β:*

i) $f \sqsubseteq g$ *implies* $\underline{\mathsf{cw}}\mathsf{p}[P](f) \sqsubseteq \underline{\mathsf{cw}}\mathsf{p}[P](g)$ *and likewise for* $\underline{\mathsf{cwl}}\mathsf{p}$.

ii) $\underline{\mathsf{cw}}\mathsf{p}[P](\alpha \cdot f + \beta \cdot g) = \alpha \cdot \underline{\mathsf{cw}}\mathsf{p}[P](f) + \beta \cdot \underline{\mathsf{cw}}\mathsf{p}[P](g)$.

iii) $\underline{\mathsf{cw}}\mathsf{p}[P](\mathbf{0}) = \mathbf{0}$ *and* $\underline{\mathsf{cwl}}\mathsf{p}[P](\mathbf{1}) = \mathbf{1}$.

Proof. Using Theorem V.1 one can show that the transformers $\underline{\mathsf{cw}}\mathsf{p}/\underline{\mathsf{cwl}}\mathsf{p}$ inherit these properties from the transformers wp/wlp. For details we refer to Appendix D. □

We conclude this section by discussing alternative approaches for providing an expectation transformer semantics for $P \in \mathsf{cpGCL}^{\boxtimes}$. By Theorem V.1, the transformers $\underline{\mathsf{cwl}}\mathsf{p}[P]$ and $\underline{\mathsf{cwl}}\mathsf{p}[P]$ can be recast as:

$$f \mapsto \frac{\mathsf{wp}[P](f)}{\mathsf{wlp}[P](\mathbf{1})} \quad \text{and} \quad f \mapsto \frac{\mathsf{wlp}[P](f)}{\mathsf{wlp}[P](\mathbf{1})}\ ,$$

respectively. Recall that $\mathsf{wlp}[P](\mathbf{1})$ yields the weakest pre–expectation under which P either does not terminate or does terminate while passing all $\mathtt{observe}$–statements. An alternative is to normalize using wp in the denominator instead of wlp, yielding:

$$f \mapsto \frac{\mathsf{wp}[P](f)}{\mathsf{wp}[P](\mathbf{1})} \qquad \text{and} \qquad f \mapsto \frac{\mathsf{wlp}[P](f)}{\mathsf{wp}[P](\mathbf{1})}$$

The transformer on the right is not meaningful, as the denominator $\mathsf{wp}[P](\mathbf{1})(\sigma)$ may be smaller than the numerator $\mathsf{wlp}[P](f)(\sigma)$ for some state $\sigma \in \mathbb{S}$. This would lead to probabilities exceeding one. The transformer on the left normalizes w.r.t. the terminating executions. This interpretation corresponds to the semantics of the probabilistic programming language $\mathtt{R2}$ [7], [14] and is only meaningful if programs terminate almost surely (i.e. with probability one).

A noteworthy consequence of adopting this semantics is that $\mathtt{observe}\, G$ is equivalent to $\mathtt{while}\,(\neg G)\,\{\mathtt{skip}\}$ [14], see the discussion in Section VI.

Let us briefly compare the four alternatives. To that end consider the program P below

$$\begin{aligned}\{\mathtt{abort}\}\ [\nicefrac{1}{2}]\big\{\{x \coloneqq 0\}\ [\nicefrac{1}{2}]\ \{x \coloneqq 1\};\\ \{y \coloneqq 0\}\ [\nicefrac{1}{2}]\ \{y \coloneqq 1\};\ \mathtt{observe}\, x = 0 \vee y = 0\big\}\end{aligned}$$

P tosses a fair coin and according to the outcome either diverges or tosses a fair coin twice and observes at least once heads ($y{=}0 \vee x{=}0$). We measure the probability that the outcome of the last coin toss was heads according to each transformer:

$$\frac{\mathsf{wp}[P](\chi_{y=0})}{\mathsf{wlp}[P](\mathbf{1})} = \frac{2}{7} \qquad \frac{\mathsf{wlp}[P](\chi_{y=0})}{\mathsf{wlp}[P](\mathbf{1})} = \frac{6}{7}$$

$$\frac{\mathsf{wp}[P](\chi_{y=0})}{\mathsf{wp}[P](\mathbf{1})} = \frac{2}{3} \qquad \frac{\mathsf{wlp}[P](\chi_{y=0})}{\mathsf{wp}[P](\mathbf{1})} = 2$$

As mentioned before, the transformer $f \mapsto \frac{\mathsf{wlp}[P](f)}{\mathsf{wp}[P](\mathbf{1})}$ is not significant as it yields a "probability" exceeding one. Note that our cwp–semantics yields a probability of $y{=}0$ on termination—while passing all `observe`–statements—of $\frac{2}{7}$. As shown before, this is a conservative and natural extension of the wp–semantics. This does not apply to the `R2`–semantics, as this would require an adaptation of rules for `abort` and `while`.

B. Correspondence Theorem

We now investigate the connection between the operational semantics of Section IV (for fully probabilistic programs) and the cwp–semantics. We start with some auxiliary results. The first result establishes a relation between (liberal) expected rewards and weakest (liberal) pre–expectations.

Lemma V.4. *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $f \in \mathbb{E}$, $g \in \mathbb{E}_{\leq 1}$, and $\sigma \in \mathbb{S}$:*

$$\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}\left(\lozenge\langle \mathit{sink}\rangle\right) = \mathsf{wp}[P](f)(\sigma) \qquad \text{(i)}$$

$$\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![P]\!]}\left(\lozenge\langle \mathit{sink}\rangle\right) = \mathsf{wlp}[P](g)(\sigma) \qquad \text{(ii)}$$

Proof. By induction on P, see Appendix E and F. □

The next result establishes that the probability to never reach $\langle \lightning \rangle$ in the RMC of program P coincides with the weakest liberal pre–expectation of P w.r.t. post–expectation $\mathbf{1}$:

Lemma V.5. *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $g \in \mathbb{E}_{\leq 1}$, and $\sigma \in \mathbb{S}$:*

$$\Pr^{\mathcal{R}^g_\sigma[\![P]\!]}(\neg\lozenge\lightning) \;=\; \mathsf{wlp}[P](\mathbf{1})(\sigma)$$

Proof. See Appendix G □

We now have all prerequisites in order to present the main result of this section: the correspondence between the operational and expectation transformer semantics of $\mathsf{cpGCL}^{\boxtimes}$ programs. It turns out that the weakest (liberal) pre–expectation $\underline{\mathsf{cw}}\mathsf{p}[P](f)(\sigma)$ (respectively $\underline{\mathsf{cwl}}\mathsf{p}[P](f)(\sigma)$) coincides with the conditional (liberal) expected reward in the RMC $\mathcal{R}^f_\sigma[\![P]\!]$ of terminating while never violating an `observe`-statement, i.e., avoiding the $\langle \lightning \rangle$ states.

Theorem V.6 (Correspondence theorem)**.** *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $f \in \mathbb{E}$, $g \in \mathbb{E}_{\leq 1}$ and $\sigma \in \mathbb{S}$,*

$$\mathsf{CExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}\left(\lozenge \mathit{sink} \,|\, \neg\lozenge\lightning\right) \;=\; \underline{\mathsf{cw}}\mathsf{p}[P](f)(\sigma)$$

$$\mathsf{CLExpRew}^{\mathcal{R}^g_\sigma[\![P]\!]}\left(\lozenge \mathit{sink} \,|\, \neg\lozenge\lightning\right) \;=\; \underline{\mathsf{cwl}}\mathsf{p}[P](g)(\sigma)\ .$$

Proof. The proof makes use of Lemmas V.4, V.5, and Theorem V.1. For details see Appendix H. □

Theorem V.6 extends a previous result [10] that established a connection between an operational and the wp/wlp semantics for pGCL programs to the fully probabilistic fragment of cpGCL.

VI. Applications

In this section we study approaches that make use of our semantics in order to analyze fully probabilistic programs with observations. We first present a program transformation based on *hoisting* `observe` statements in a way that probabilities of conditions are extracted, allowing for a subsequent analysis on an observation–free program. Furthermore, we discuss how observations can be replaced by loops and vice versa. Finally, we use a well–known case study to demonstrate the direct applicability of our cwp–semantics.

A. Observation Hoisting

In what follows we give a semantics–preserving transformation for removing observations from $\mathsf{cpGCL}^{\boxtimes}$ programs. Intuitively, the program transformation "hoists" the `observe` statements while updating the probabilities in case of probabilistic choices. Given $P \in \mathsf{cpGCL}^{\boxtimes}$, the transformation delivers a semantically equivalent `observe`–free program $\hat{P} \in \mathsf{pGCL}^{\boxtimes}$ and—as a side product—an expectation $\hat{h} \in \mathbb{E}_{\leq 1}$ that captures the probability of the original program to establish all `observe` statements. For intuition, reconsider the program from Example V.1. The transformation yields the program

$$\{x := 0\}\ [{}^{8}\!/\!{}_{13}]\ \{x := 1\};$$
$$\mathtt{ite}\,(x = 1)\,\{\{y := 0\}\ [1]\ \{y := 2\}\}$$
$$\{\{y := 0\}\ [1]\ \{y := 3\}\}$$

and expectation $\hat{h} = \frac{\mathbf{13}}{\mathbf{20}}$. By eliminating dead code in both probabilistic choices and coalescing the branches in the conditional, we can simplify the program to:

$$\{x := 0\}\ [{}^{8}\!/\!{}_{13}]\ \{x := 1\};\ y := 0$$

As a sanity check note that the expected value of $10{+}x$ in this program is equal to $10 \cdot \frac{8}{13} + 11 \cdot \frac{5}{13} = \frac{135}{13}$, which agrees with the result obtained in Example V.1 by analyzing the original program. Formally, the program transformation is given by a function

$$\mathcal{T} : \mathsf{cpGCL}^{\boxtimes} \times \mathbb{E}_{\leq 1} \to \mathsf{cpGCL}^{\boxtimes} \times \mathbb{E}_{\leq 1}\ .$$

To apply the transformation to a program P we need to determine $\mathcal{T}(P, \mathbf{1})$, which gives the semantically equivalent program $\hat{P}$ and the expectation $\hat{h}$.

The transformation is defined in Figure 3 and works by inductively computing the weakest pre–expectation that guarantees the establishment of all `observe` statements and updating the probability parameter of probabilistic choices so that the pre–expectations of their branches are established in accordance with the original probability parameter. The computation of these pre–expectations is performed following the same rules as the wlp operator. The correctness of the transformation is established by the following Theorem, which states that a program and its transformed version share the same terminating and non–terminating behavior.

Theorem VI.1 (Program Transformation Correctness)**.** *Let $P \in \mathsf{cpGCL}^{\boxtimes}$ admit at least one feasible run for every initial state and $\mathcal{T}(P, \mathbf{1}) = (\hat{P}, \hat{h})$. Then for any $f \in \mathbb{E}$ and $g \in \mathbb{E}_{\leq 1}$,*

$$\mathsf{wp}[\hat{P}](f) = \underline{\mathsf{cw}}\mathsf{p}[P](f) \quad \textit{and} \quad \mathsf{wlp}[\hat{P}](g) = \underline{\mathsf{cwl}}\mathsf{p}[P](g).$$

Proof. See Appendix I. □

A similar program transformation has been given in [7]. Whereas they use random assignments to introduce randomization in their programming model, we use probabilistic choices. Consequently, they can hoist `observe` statements only until the occurrence of a random assignment, while we are able to hoist `observe` statements through probabilistic choices and completely remove them from programs. Another difference is that their semantics only accounts for terminating program behaviors and thus can guarantee the correctness of the program transformation for terminating behaviors only. Our semantics is more expressive and enables establishing the correctness of the program transformation for non–terminating program behavior, too.

B. Replacing Observations by Loops

For semantics that normalize with respect to the terminating behavior of programs, `observe` statements can readily be replaced by a loop [24], [14]. In our setting a more intricate transformation is required to eliminate observations from programs. Briefly stated, the idea is to restart a violating run from the initial state until it satisfies all encountered observations. To achieve this we consider a fresh variable *rerun* and transform a given program $P \in \mathsf{cpGCL}^{\boxtimes}$ into a new program P' as described below:

$$\begin{array}{rcl} \texttt{observe}\, G & \rightarrow & \texttt{ite}\,(\neg G)\,\{\textit{rerun} \coloneqq \mathsf{true}\}\,\{\texttt{skip}\} \\ \texttt{abort} & \rightarrow & \texttt{ite}\,(\neg \textit{rerun})\,\{\texttt{abort}\}\,\{\texttt{skip}\} \\ \texttt{while}\,(G)\,\{\ldots\} & \rightarrow & \texttt{while}\,(G \wedge \neg \textit{rerun})\,\{\ldots\} \end{array}$$

For conditional and probabilistic choice, we apply the above rules recursively to the subprograms.

The aim of the transformation is twofold. First, the program P' flags the violation of an `observe` statement through the variable *rerun*. If a violation occurs, *rerun* is set to true while in contrast to the original program we continue the program execution. As a side effect, we may introduce some subsequent diverging behavior which would not be present in the original program (since the execution would have already been blocked). The second aim of the transformation is to avoid this possible diverging–behavior. This is achieved by blocking `while`–loops and `abort` statements once *rerun* is set to true.

Now we can get rid of the observations in P by repeatedly executing P' from the same initial state till *rerun* is set to false (which would intuitively correspond to P passing all its observations).

This is implemented by program P'' below:

$$s_1, \ldots, s_n \coloneqq x_1, \ldots, x_n;\ \textit{rerun} \coloneqq \mathsf{true};$$
$$\texttt{while}(\textit{rerun})\,\{\, x_1, \ldots, x_n \coloneqq s_1, \ldots, s_n;\ P' \,\}$$

Here, $s_1, \ldots, s_n$ are fresh variables and $x_1, \ldots, x_n$ are all program variables of P. The first assignment stores the initial state in the variables s_i and the first line of the loop body, ensures that the loop always starts with the same (initial) values.

Theorem VI.2. *Let programs P and P'' be as above. Then*

$$\underline{\mathsf{cw}}\mathsf{p}[P](f) = \mathsf{wp}[P''](f) \ .$$

Proof. See Appendix J. □

Example VI.1. Consider the following cpGCL program:

```
{$x \coloneqq 0$} [$p$] {$x \coloneqq 1$}; {$y \coloneqq 0$} [$p$] {$y \coloneqq 1$}
observe $x \neq y$;
```

We apply the program transformation to it and obtain:

```
$s_1, s_2 \coloneqq x, y$; $\textit{rerun} \coloneqq \mathsf{true}$;
while($\textit{rerun}$){
  $x, y \coloneqq s_1, s_2$; $\textit{rerun} \coloneqq \mathsf{false}$;
  {$x \coloneqq 0$} [$p$] {$x \coloneqq 1$};
  {$y \coloneqq 0$} [$p$] {$y \coloneqq 1$};
  if($x = y$){ $\textit{rerun} \coloneqq \mathsf{true}$}
}
```

This program is simplified by a data flow analysis: The variables s_1 and s_2 are irrelevant because x and y are overwritten in every iteration. Furthermore, there is only one observation so that its predicate can be pushed directly into the loop's guard. Then the initial values of x and y may be arbitrary but they must be equal to make sure the loop is entered. This gives the final result

```
$x, y \coloneqq 0, 0$;
while($x = y$){
  {$x \coloneqq 0$} [$p$] {$x \coloneqq 1$}; {$y \coloneqq 0$} [$p$] {$y \coloneqq 1$}
}
```

This program is a simple algorithm that repeatedly uses a biased coin to simulate an unbiased coin flip. A proof that x is indeed distributed uniformly over $\{0, 1\}$ has been previously shown e.g. in [25].

Theorem VI.2 shows how to define and effectively calculate the conditional expectation using a straightforward program transformation and the well established notion of wp. However in practice it will often be infeasible to calculate the fixed point of the outer loop or to find a suitable loop invariant – even though it exists.

C. Replacing Loops by Observations

In this section we provide an overview on how the aforementioned result can be "applied backwards" in order to replace a loop by an `observe` statement. This is useful as it is easier to analyze a loop–free program with observations

$$
\begin{array}{lcl}
\mathcal{T}(\texttt{skip}, f) & = & (\texttt{skip}, f) \\
\mathcal{T}(\texttt{abort}, f) & = & (\texttt{abort}, \mathbf{1}) \\
\mathcal{T}(x \coloneqq E, f) & = & (x \coloneqq E, f[E/x]) \\
\mathcal{T}(\texttt{observe}\, G, f) & = & (\texttt{skip}, \chi_G \cdot f) \\
\mathcal{T}(\texttt{ite}\,(G)\,\{P\}\,\{Q\}, f) & = & (\texttt{ite}\,(G)\,\{P'\}\,\{Q'\}, \chi_G \cdot f_P + \chi_{\neg G} \cdot f_Q) \\
 & & \text{where } (P', f_P) = \mathcal{T}(P, f),\ (Q', f_Q) = \mathcal{T}(Q, f) \\
\mathcal{T}(\{P\}\ [p]\ \{Q\}, f) & = & (\{P'\}\ [p']\ \{Q'\}, p \cdot f_P + (\mathbf{1}{-}p) \cdot f_Q) \\
 & & \text{where } (P', f_P) = \mathcal{T}(P, f),\ (Q', f_Q) = \mathcal{T}(Q, f), \text{and } p' = \frac{p \cdot f_P}{p \cdot f_P + (\mathbf{1}-p) \cdot f_Q} \\
\mathcal{T}(\texttt{while}\,(G)\,\{P\}, f) & = & (\texttt{while}\,(G)\,\{P'\}, f') \\
 & & \text{where } f' = \boldsymbol{\nu} X\boldsymbol{\cdot}\ (\chi_G \cdot (\pi_2 \circ \mathcal{T})(P, X) + \chi_{\neg G} \cdot f)\,, \text{and } (P', \_) = \mathcal{T}(P, f') \\
\mathcal{T}(P; Q, f) & = & (P'; Q', f'') \text{ where } (Q', f') = \mathcal{T}(Q, f),\ (P', f'') = \mathcal{T}(P, f')
\end{array}
$$

Fig. 3. Program transformation for eliminating observe statements in $\mathsf{cpGCL}^{\boxtimes}$.

than a program with loops for which fixed points need to be determined.

The transformation presented in Section VI-B yields programs of a certain form: In every loop iteration the variable values are initialized independently from their values after the previous iteration. Hence the loop iterations generate a sequence of program variable valuations that are *independent and identically distributed* (iid loop), cf. Example VI.1 where no "data flow" between iterations of the loop occurs.

In general, if $\mathsf{loop} = \texttt{while}(G)\{P\}$ is an iid loop we can obtain a program $Q = P; \texttt{observe}\,\neg G$ with

$$\mathsf{wp}[\mathsf{loop}](f) = \underline{\mathsf{cw}}\mathsf{p}[Q](f)$$

for any expectation $f \in \mathbb{E}$. To see this, apply Theorem VI.2 to program Q. Let the resulting program be loop'. As in Example VI.1, note that there is only one observe statement at the end of loop' and furthermore there is no data flow between iterations of loop'. Hence by the same simplification steps we arrive at the desired program loop.

D. The Crowds *Protocol*

To demonstrate the applicability of the cwp-semantics to a practical example, consider the Crowds-protocol [18]. A set of nodes forms a fully connected network called the *crowd*. Crowd members would like to exchange messages with a server without revealing their identity to the server. To achieve this, a node *initiates communication* by sending its message to a randomly chosen crowd member, possibly itself. Upon receiving a message a node probabilistically decides to either *forward* the message once again to a randomly chosen node in the network or to relay it to the server directly. A commonly studied attack scenario is that some malicious nodes called *collaborators* join the crowd and participate in the protocol with the aim to reveal the identity of the sender. The following cpGCL-program P models this protocol where p is the forward probability and c is the fraction of collaborating nodes in the crowd. The initialization corresponds to the communication initiation.

```
init :   $\{intercepted \coloneqq 1\}\ [c]\ \{intercepted \coloneqq 0\}$;
         $delivered \coloneqq 0;\ counter \coloneqq 1$
loop :   while$(delivered = 0)\,\{$
             $\{counter \coloneqq counter + 1;$
             $\{intercepted \coloneqq 1\}\ [c]\ \{$skip$\}\}$
             $[p]$
             $\{delivered \coloneqq 1\}$
         $\}$;
         observe$(counter \leq k)$
```

Our goal is to determine the probability of a message not being intercepted by a collaborator. We condition this by the observation that a message is forwarded at most k times.

Note that the operational semantics of P produce an *infinite parametric RMC* since the value of k is fixed but arbitrary. Using Theorem V.1 we express the probability that a message is not intercepted given that it was rerouted no more than k times by

$$\underline{\mathsf{cw}}\mathsf{p}[P]([\neg intercepted]) = \frac{\mathsf{wp}[P]([\neg intercepted])}{\mathsf{wlp}[P](\mathbf{1})} \qquad (1)$$

The computation of this quantity requires to find fixed points, cf. Appendix K for details. As a result we obtain a closed form solution parametrized in p, c, and k:

$$(1-c)(1-p)\frac{1-(p(1-c))^k}{1-p(1-c)} \cdot \frac{1}{1-p^k}$$

The automation of such analyses remains a challenge and is part of ongoing and future work.

VII. Denotational Semantics for Full cpGCL

In this section we argue why (under mild assumptions) it is not possible to come up with a denotational semantics in the style of conditional pre–expectation transformers (CPETs for short) for full cpGCL. To show this, it suffices to consider a simple fragment of cpGCL containing only assignments, observations, probabilistic and non–deterministic choices. Let x be the only program variable that can be written or read in this fragment. We denote this fragment by cpGCL^{-}. Assume

D is some appropriate domain for *representing* conditional expectations of the program variable x with respect to some *fixed* initial state σ_0 and let $[\![\,\cdot\,]\!]\colon D \to \mathbb{R} \cup \{\bot\}$ be an interpretation function such that for any $d \in D$ we have that $[\![d]\!]$ *is equal to* the (possibly undefined) conditional expected value of x.

Definition VII.1 (Inductive CPETs)**.** A *CPET* is a function $\mathsf{cwp}^*\colon \mathsf{cpGCL}^- \to D$ such that for any $P \in \mathsf{cpGCL}^-$, $[\![\mathsf{cwp}[P]]\!] = \mathsf{CExpRew}^{\mathfrak{R}^x_{\sigma_0}[\![P]\!]}\,(\Diamond\, \mathit{sink} \mid \neg\Diamond\, \lightning)$. cwp^* is called *inductive*, if there exists some function $\mathcal{K}\colon \mathsf{cpGCL}^- \times [0,\,1] \times \mathsf{cpGCL}^- \to D$ such that for any $P_1, P_2 \in \mathsf{cpGCL}^-$,

$$\mathsf{cwp}^*[\{P_1\}\ [p]\ \{P_2\}] \;=\; \mathcal{K}(\mathsf{cwp}^*[P_1],\, p,\, \mathsf{cwp}^*[P_2])\ ,$$

and some function $\mathcal{N}\colon \mathsf{cpGCL}^- \times \mathsf{cpGCL}^- \to D$ with

$$\mathsf{cwp}^*[\{P_1\} \,\square\, \{P_2\}] \;=\; \mathcal{N}(\mathsf{cwp}^*[P_1],\, \mathsf{cwp}^*[P_2])\ ,$$

where $\forall d_1, d_2 \in D\colon \mathcal{N}(d_1,\, d_2) \;\in\; \{d_1,\, d_2\}$.

This definition suggests that the conditional pre–expectation of $\{P_1\}\ [p]\ \{P_2\}$ is determined only by the conditional pre–expectation of P_1, the conditional pre–expectation of P_2, and the probability p. Furthermore the above definition suggests that the conditional pre–expectation of $\{P_1\} \,\square\, \{P_2\}$ is also determined by the conditional pre–expectation of P_1 and the conditional pre–expectation of P_2 only. Consequently, the non–deterministic choice can be resolved by replacing it either by P_1 or P_2. While this might seem like a strong limitation, the above definition is compatible with the interpretation of non–deterministic choice as demonic choice: The choice is deterministically driven towards the worst option. The requirement $N(d_1,\, d_2) \;\in\; \{d_1,\, d_2\}$ is also necessary for interpreting non–deterministic choice as an abstraction where implementational details are not important.

As we assume a fixed initial state and a fixed post–expectation, the non–deterministic choice turns out to be deterministic once the pre–expectations of P_1 and P_2 are known. Under the above assumptions (which do apply to the wp and wlp transformers) we claim:

Theorem VII.1. *There exists no inductive CPET.*

Proof. The proof goes by contradiction. Consider the program $P = \{P_1\}\ [1/2]\ \{P_5\}$ with

$$\begin{aligned}
P_1 &= x \coloneqq 1\\
P_5 &= \{P_2\} \,\square\, \{P_4\}\\
P_2 &= x \coloneqq 2\\
P_4 &= \{\mathtt{observe\ false}\}\ [1/2]\ \{P_{2+\varepsilon}\}\\
P_{2+\varepsilon} &= x \coloneqq 2 + \varepsilon\ ,
\end{aligned}$$

where $\varepsilon > 0$. A schematic depiction of the RMDP $\mathfrak{R}^x_{\sigma_0}[\![P]\!]$ is given in Figure 4. Assume there exists an inductive CPET cwp^* over some appropriate domain D. Then,

$$\begin{aligned}
\mathsf{cwp}^*[P_1] &= d_1,\ \text{with } [\![d_1]\!] = 1\\
\mathsf{cwp}^*[P_2] &= d_2,\ \text{with } [\![d_2]\!] = 2\\
\mathsf{cwp}^*[P_{2+\varepsilon}] &= d_{2+\varepsilon},\ \text{with } [\![d_{2+\varepsilon}]\!] = 2 + \varepsilon\\
\mathsf{cwp}^*[\mathtt{observe\ false}] &= \mathfrak{of},\ \text{with } [\![\mathfrak{of}]\!] = \bot
\end{aligned}$$

for some appropriate $d_1,\, d_2,\, d_{2+\varepsilon},\, \mathfrak{of} \in D$. By Definition VII.1, cwp^* being inductive requires the existence of a function $\mathcal{K}$, such that

$$\begin{aligned}
\mathsf{cwp}^*[P_4] &= \mathcal{K}\,(\mathsf{cwp}^*[\mathtt{observe\ false}],\, 1/2,\, \mathsf{cwp}^*[P_{2+\varepsilon}])\\
&= \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2+\varepsilon})\ .
\end{aligned}$$

In addition, there must be an $\mathcal{N}$ with:

$$\begin{aligned}
\mathsf{cwp}^*[P_5] &= \mathcal{N}\,(\mathsf{cwp}^*[P_2],\, \mathsf{cwp}^*[P_4])\\
&= \mathcal{N}(d_2,\, \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2+\varepsilon}))\ .
\end{aligned}$$

Since P_4 is a probabilistic choice between an infeasible branch and $P_{2+\varepsilon}$, the expected value for x has to be rescaled to the feasible branch. Hence P_4 yields $[\![\mathsf{cwp}^*[P_4]]\!] = 2+\varepsilon$, whereas $[\![\mathsf{cwp}^*[P_2]]\!] = 2$. Thus:

$$[\![d_2]\!] \;\lneq\; [\![\mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2+\varepsilon})]\!] \tag{2}$$

As non–deterministic choice is demonic, we have:

$$\mathsf{cwp}^*[P_5] \;=\; \mathcal{N}(d_2,\, \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2+\varepsilon})) \;=\; d_2 \tag{3}$$

As $\mathcal{N}\,(\mathsf{cwp}^*[P_2],\, \mathsf{cwp}^*[P_4]) \in \{\mathsf{cwp}^*[P_2],\, \mathsf{cwp}^*[P_4]\}$ we can resolve non–determinism in P by either rewriting P to $\{P_1\}\ [1/2]\ \{P_2\}$ which gives

$$[\![\mathsf{cwp}^*\{P_1\}\ [1/2]\ \{P_2\}]\!] \;=\; \frac{3}{2}\ ,$$

or we rewrite P to $\{P_1\}\ [1/2]\ \{P_4\}$, which gives

$$[\![\mathsf{cwp}^*\{P_1\}\ [1/2]\ \{P_4\}]\!] \;=\; \frac{4+\varepsilon}{3}\ .$$

For a sufficiently small ε the second option should be preferred by a demonic scheduler. This, however, suggests:

$$\begin{aligned}
\mathsf{cwp}^*[P_5] &= \mathcal{N}(d_2,\, \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2+\varepsilon}))\\
&= \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2+\varepsilon})
\end{aligned}$$

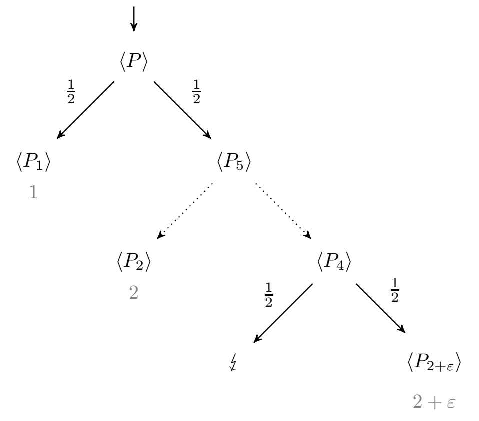


Fig. 4. Schematic depiction of the RMDP $\mathfrak{R}^x_{\sigma_0}[\![P]\!]$

Together with Equality (3) we get $d_2 = \mathcal{K}(\mathfrak{of},\, {}^1\!/_2,\, d_{2+\varepsilon})$, which implies $[\![d_2]\!] = [\![\mathcal{K}(\mathfrak{of},\, {}^1\!/_2,\, d_{2+\varepsilon})]\!]$. This is a contradiction to Inequality (2). □

As an immediate corollary of Theorem VII.1 we obtain the following statement:

Corollary VII.2. *We cannot extend the* cwp *rules in Figure 2 for non–deterministic programs such that Theorem V.6 extends to full* cpGCL.

This result is related to the fact that for minimizing conditional (reachability) probabilities in RMDPs positional, i.e. history–independent, schedulers are insufficient [26]. Intuitively speaking, if a *history–dependent* scheduler is required, this necessitates the inductive definition of cwp^* to take the context of a statement (if any) into account. This conflicts with the principle of an inductive definition. Investigating the precise relationship with the result of [26] requires further study.

VIII. Conclusion and Future Work

This paper presented an extensive treatment of semantic issues in probabilistic programs with conditioning. Major contributions are the treatment of non–terminating programs (both operationally and for weakest liberal pre–expectations), our results on combining non–determinism with conditioning, as well as the presented program transformations. We firmly believe that a thorough understanding of these semantic issues provides a main cornerstone for enabling automated analysis techniques such as loop invariant synthesis [16], [27], program analysis [28] and model checking [22] to the class of probabilistic programs with conditioning. Future work consists of investigating conditional invariants and a further investigation of non–determinism in combination with conditioning.

Acknowledgment

This work was supported by the Excellence Initiative of the German federal and state government. Moreover, we would like to thank Pedro d'Argenio and Tahiry Rabehaja for the valuable discussions preceding this paper.

References

[1] N. D. Goodman and A. Stuhlmüller, *The Design and Implementation of Probabilistic Programming Languages*. (electronic), 2014, http://dippl.org.

[2] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani, "Probabilistic programming," in *Proc. of FOSE*. ACM Press, 2014, pp. 167–181.

[3] G. Barthe, B. Köpf, F. Olmedo, and S. Z. Béguelin, "Probabilistic relational reasoning for differential privacy," *ACM Trans. Program. Lang. Syst.*, vol. 35, no. 3, p. 9, 2013.

[4] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum, "Church: a language for generative models," in *Proc. of UAI*. AUAI Press, 2008, pp. 220–229.

[5] B. Paige and F. Wood, "A compilation target for probabilistic programming languages," in *Proc. of ICML*, vol. 32. JMLR.org, 2014, pp. 1935–1943.

[6] A. D. Gordon, T. Graepel, N. Rolland, C. V. Russo, J. Borgström, and J. Guiver, "Tabular: a schema-driven probabilistic programming language," in *Proc. of POPL*. ACM Press, 2014, pp. 321–334.

[7] A. V. Nori, C.-K. Hur, S. K. Rajamani, and S. Samuel, "R2: An efficient MCMC sampler for probabilistic programs," in *Proc. of AAAI*. AAAI Press, July 2014.

[8] D. Kozen, "Semantics of probabilistic programs," *J. Comput. Syst. Sci.*, vol. 22, no. 3, pp. 328–350, 1981.

[9] A. McIver and C. Morgan, *Abstraction, Refinement And Proof For Probabilistic Systems*. Springer, 2004.

[10] F. Gretz, J.-P. Katoen, and A. McIver, "Operational versus weakest preexpectation semantics for the probabilistic guarded command language," *Perform. Eval.*, vol. 73, pp. 110–132, 2014.

[11] C. Jones and G. D. Plotkin, "A probabilistic powerdomain of evaluations," in *Logic in Computer Science*. IEEE Computer Society, 1989, pp. 186–195.

[12] V. Gupta, R. Jagadeesan, and V. A. Saraswat, "Probabilistic concurrent constraint programming," in *Concurrency Theory*, ser. LNCS, vol. 1243. Springer, 1997, pp. 243–257.

[13] D. S. Scott, "Stochastic λ-calculi: An extended abstract," *J. Applied Logic*, vol. 12, no. 3, pp. 369–376, 2014.

[14] C.-K. Hur, A. V. Nori, S. K. Rajamani, and S. Samuel, "Slicing probabilistic programs," in *Proc. of PLDI*. ACM Press, 2014, pp. 133–144.

[15] A. Sampson, P. Panchekha, T. Mytkowicz, K. S. McKinley, D. Grossman, and L. Ceze, "Expressing and verifying probabilistic assertions," in *ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2014, p. 14.

[16] A. Chakarov and S. Sankaranarayanan, "Expectation invariants for probabilistic program loops as fixed points," in *Proc. of SAS*, ser. LNCS, vol. 8723. Springer, 2014, pp. 85–100.

[17] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[18] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 66–92, 1998.

[19] E. W. Dijkstra, *A Discipline of Programming*. Prentice Hall, 1976.

[20] A. McIver and C. Morgan, "Partial correctness for probabilistic demonic programs," *Theoretical Computer Science*, vol. 266, no. 12, pp. 513 – 541, 2001.

[21] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[22] C. Baier, J. Klein, S. Klüppelholz, and S. Märcker, "Computing conditional probabilities in Markovian models efficiently," in *Proc. of TACAS*, ser. LNCS, vol. 8413. Springer, 2014, pp. 515–530.

[23] G. Nelson, "A generalization of Dijkstra's calculus," *ACM Trans. Program. Lang. Syst.*, vol. 11, no. 4, pp. 517–561, 1989.

[24] G. Claret, S. K. Rajamani, A. V. Nori, A. D. Gordon, and J. Borgström, "Bayesian inference using data flow analysis," in *Proc. of ESEC/SIGSOFT FSE*. ACM Press, 2013, pp. 92–102.

[25] F. Gretz, J.-P. Katoen, and A. McIver, "Prinsys - on a quest for probabilistic loop invariants," in *Proc. of QEST*, ser. LNCS, vol. 8054. Springer, 2013, pp. 193–208.

[26] M. E. Andrés and P. van Rossum, "Conditional probabilities over probabilistic and nondeterministic systems," in *Proc. of TACAS*, ser. LNCS, vol. 4963. Springer, 2008, pp. 157–172.

[27] J. Katoen, A. McIver, L. Meinicke, and C. C. Morgan, "Linear-invariant generation for probabilistic programs: - automated support for proof-based methods," in *Proc. of SAS*, ser. LNCS, vol. 6337. Springer, 2010, pp. 390–406.

[28] P. Cousot and M. Monerau, "Probabilistic abstract interpretation," in *Proc. of ESOP*, ser. LNCS, H. Seidl, Ed., vol. 7211. Springer, 2012, pp. 169–193.

[29] H. Bekic, "Definable operation in general algebras, and the theory of automata and flowcharts," in *Programming Languages and Their Definition*. Springer, 1984, pp. 30–55.

APPENDIX

A. *Continuity of* wp *and* wlp

Lemma A.1 (Continuity of wp/wlp)**.** *Consider the extension of* wp *and* wlp *to* cpGCL *given by*

$$\begin{aligned} \mathsf{wp}[\mathtt{observe}\, G](f) &= \chi_G \cdot f \\ \mathsf{wlp}[\mathtt{observe}\, G](g) &= \chi_G \cdot g \ . \end{aligned}$$

Then for every $P \in \mathsf{cpGCL}$ *the expectation transformers* $\mathsf{wp}[P] \colon \mathbb{E} \to \mathbb{E}$ *and* $\mathsf{wlp}[P] \colon \mathbb{E}_{\leq 1} \to \mathbb{E}_{\leq 1}$ *are continuous mappings over* $(\mathbb{E}, \sqsubseteq)$ *and* $(\mathbb{E}_{\leq 1}, \sqsupseteq)$*, respectively.*

Proof. For proving the continuity of wp we have to show that for any directed subset $D \subseteq \mathbb{E}$ we have

$$\sup_{f \in D} \mathsf{wp}[P](f) \;=\; \mathsf{wp}[P]\left(\sup_{f \in D} f\right) \ . \qquad (4)$$

This can be shown by structural induction on P. All cases except for the $\mathtt{observe}$ statement have been covered in [10]. It remains to show that Equality (4) holds for $P = \mathtt{observe}\ G$:

$$\begin{aligned} \sup_{f \in D} \mathsf{wp}[\mathtt{observe}\ G](f) &= \sup_{f \in D} \chi_G \cdot f \\ &= \chi_G \cdot \sup_{f \in D} f \\ &= \mathsf{wp}[\mathtt{observe}\, G](\sup_{f \in D} f) \end{aligned}$$

The proof for the liberal transformer wlp is analogous. □

B. *Proof of Theorem V.1*

Theorem V.1 (Decoupling of cwp/cwlp)**.** *For* $P \in \mathsf{cpGCL}^{\boxtimes}$*,* $f \in \mathbb{E}$*, and* $f', g \in \mathbb{E}_{\leq 1}$*:*

$$\begin{aligned} \mathsf{cwp}[P](f,\, g) &= \big(\mathsf{wp}[P](f),\, \mathsf{wlp}[P](g)\big) \\ \mathsf{cwlp}[P](f',\, g) &= \big(\mathsf{wlp}[P](f'),\, \mathsf{wlp}[P](g)\big) \end{aligned}$$

Proof. The proof of Theorem V.1 goes by induction over all $\mathsf{cpGCL}^{\boxtimes}$ programs. For the induction base we have:

a) The Effectless Program $\mathtt{skip}$*.:* For cwp we have:

$$\begin{aligned} \mathsf{cwp}[\mathtt{skip}](f,\, g) &= (f,\, g) \\ &= \big(\mathsf{wp}[\mathtt{skip}](f),\, \mathsf{wlp}[\mathtt{skip}](g)\big) \end{aligned}$$

The argument for cwlp is completely analogous.

b) The Faulty Program $\mathtt{abort}$*.:* For cwp we have:

$$\begin{aligned} \mathsf{cwp}[\mathtt{abort}](f,\, g) &= (\mathbf{0},\, \mathbf{1}) \\ &= \big(\mathsf{wp}[\mathtt{abort}](f),\, \mathsf{wlp}[\mathtt{abort}](g)\big) \end{aligned}$$

Analogously for cwlp we have:

$$\begin{aligned} \mathsf{cwlp}[\mathtt{abort}](f',\, g) &= (\mathbf{1},\, \mathbf{1}) \\ &= \big(\mathsf{wlp}[\mathtt{abort}](f'),\, \mathsf{wlp}[\mathtt{abort}](g)\big) \end{aligned}$$

c) The Assignment $x := E$*.:* For cwp we have:

$$\begin{aligned} \mathsf{cwp}[x := E](f,\, g) &= (f[x/E],\, g[x/E]) \\ &= \big(\mathsf{wp}[x := E](f),\, \mathsf{wlp}[x := E](g)\big) \end{aligned}$$

The argument for cwlp is completely analogous.

d) The Observation $\mathtt{observe}\ G$*.:* For cwp we have:

$$\begin{aligned} &\mathsf{cwp}[\mathtt{observe}\ G](f,\, g) \\ &= (f \cdot \chi_G,\, g \cdot \chi_G) \\ &= \big(\mathsf{wp}[\mathtt{observe}\ G](f),\, \mathsf{wlp}[\mathtt{observe}\ G](g)\big) \end{aligned}$$

The argument for cwlp is completely analogous.

e) The Induction Hypothesis:: Assume in the following that for two arbitrary but fixed programs $P, Q \in \mathsf{cpGCL}^{\boxtimes}$ it holds that both

$$\begin{aligned} \mathsf{cwp}[P](f,\, g) &= \big(\mathsf{wp}[P](f), \mathsf{wlp}[P](g)\big), \text{ and} \\ \mathsf{cwlp}[P](f',\, g) &= \big(\mathsf{wlp}[P](f'), \mathsf{wlp}[P](g)\big) \ . \end{aligned}$$

Then for the induction step we have:

f) The Concatenation $P;\ Q$*.:* For cwp we have:

$$\begin{aligned} &\mathsf{cwp}[P;\ Q](f,\, g) \\ &= \mathsf{cwp}[P](\mathsf{cwp}[Q](f,\, g) \\ &= \mathsf{cwp}[P]\big(\mathsf{wp}[Q](f),\, \mathsf{wlp}[Q](g)\big) && \text{(I.H. on } Q\text{)} \\ &= \big(\mathsf{wp}[P](\mathsf{wp}[Q](f)),\, \mathsf{wlp}[P](\mathsf{wlp}[Q](g))\big) && \text{(I.H. on } P\text{)} \\ &= \big(\mathsf{wp}[P;\ Q](f),\, \mathsf{wlp}[P;\ Q](g)\big) \end{aligned}$$

The argument for cwlp is completely analogous.

g) The Conditional Choice $\mathtt{ite}\,(G)\,\{P\}\,\{Q\}$*.:* For cwp we have:

$$\begin{aligned} &\mathsf{cwp}[\mathtt{ite}\,(G)\,\{P\}\,\{Q\}](f,\, g) \\ &= \chi_G \cdot \mathsf{cwp}[P](f,\, g) + \chi_{\neg G} \cdot \mathsf{cwp}[Q](f,\, g) \\ &= \chi_G \cdot \big(\mathsf{wp}[P](f),\, \mathsf{wlp}[P](g)\big) && \text{(I.H.)} \\ &\qquad + \chi_{\neg G} \cdot \big(\mathsf{wp}[Q](f),\, \mathsf{wlp}[Q](g)\big) \\ &= \big(\chi_G \cdot \mathsf{wp}[P](f) + \chi_{\neg G} \cdot \mathsf{wp}[Q](f), \\ &\qquad \chi_G \cdot \mathsf{wlp}[P](g) + \chi_{\neg G} \cdot \mathsf{wlp}[Q](g)\big) \\ &= \big(\mathsf{wp}[\mathtt{ite}\,(G)\,\{P\}\,\{Q\}](f), \\ &\qquad \mathsf{wlp}[\mathtt{ite}\,(G)\,\{P\}\,\{Q\}](g)\big) \end{aligned}$$

The argument for cwlp is completely analogous.

h) The Probabilistic Choice $\{P\}\ [p]\ \{Q\}$*.:* For cwp we have:

$$\begin{aligned} &\mathsf{cwp}[\{P\}\ [p]\ \{Q\}](f,\, g) \\ &= p \cdot \mathsf{cwp}[P](f,\, g) + (1-p) \cdot \mathsf{cwp}[Q](f,\, g) \\ &= p \cdot \big(\mathsf{wp}[P](f),\, \mathsf{wlp}[P](g)\big) && \text{(I.H.)} \\ &\qquad + (1-p) \cdot \big(\mathsf{wp}[Q](f),\, \mathsf{wlp}[Q](g)\big) \\ &= \big(p \cdot \mathsf{wp}[P](f) + (1-p) \cdot \mathsf{wp}[Q](f), \\ &\qquad p \cdot \mathsf{wlp}[P](g) + (1-p) \cdot \mathsf{wlp}[Q](g)\big) \\ &= \big(\mathsf{wp}[\{P\}\ [p]\ \{Q\}](f),\, \mathsf{wlp}[\{P\}\ [p]\ \{Q\}](g)\big) \end{aligned}$$

The argument for cwlp is completely analogous.

i) The Loop $\mathtt{while}\,(G)\,\{P\}$*.:* For cwp we have:

$$\begin{aligned} &\mathsf{cwp}[\mathtt{while}\,(G)\,\{P\}](f,\, g) \\ &= \boldsymbol{\mu}_{\sqsubseteq,\sqsupseteq}(X_1,\, X_2)\text{•}\ \chi_G \cdot \mathsf{cwp}[P](X_1,\, X_2) + \chi_{\neg G} \cdot (f,\, g) \\ &= \boldsymbol{\mu}_{\sqsubseteq,\sqsupseteq}(X_1,\, X_2)\text{•}\ \chi_G \cdot \big(\mathsf{wp}[P](X_1),\, \mathsf{wlp}[P](X_2)\big) \\ &\qquad + \chi_{\neg G} \cdot (f,\, g) && \text{(I.H.)} \end{aligned}$$

$$\begin{aligned} &= \boldsymbol{\mu}_{\sqsubseteq,\sqsupseteq}(X_1,\, X_2)\bullet\ \big(\chi_G \cdot \mathsf{wp}[P](X_1) + \chi_{\neg G} \cdot f,\\ &\qquad \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g\big) \end{aligned}$$

Now let $H(X_1,\, X_2) = \big(\chi_G \cdot \mathsf{wp}[P](X_1) + \chi_{\neg G} \cdot f,\, \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g\big)$ and let $H_1(X_1,\, X_2)$ be the projection of $H(X_1,\, X_2)$ to the first component and let $H_2(X_1,\, X_2)$ be the projection of $H(X_1,\, X_2)$ to the second component.

Notice that the value of $H_1(X_1,\, X_2)$ does not depend on X_2 and that it is given by

$$H_1(X_1,\, \_) \;=\; \chi_G \cdot \mathsf{wp}[P](X_1) + \chi_{\neg G} \cdot f \enspace .$$

By the continuity of wp (Lemma A.1) we can establish that H_1 is continuous. Analogously the value of $H_2(X_1,\, X_2)$ does not depend on X_1 and it is given by

$$H_2(\_,\, X_2) \;=\; \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g \enspace .$$

By the continuity of wlp (Lemma A.1) we can establish that H_2 is continuous.

As both H_1 and H_2 are continuous, we can apply Bekić's Theorem [29] which tells us that the least fixed point of H is given as $\left(\widehat{X_1},\, \widehat{X_2}\right)$ with

$$\begin{aligned} \widehat{X_1} &= \boldsymbol{\mu}_{\sqsubseteq}\, X_1\bullet\ H_1\big(X_1,\, \boldsymbol{\mu}_{\sqsupseteq}\, X_2\bullet\ H_2(X_1,\, X_2)\big)\\ &= \boldsymbol{\mu}_{\sqsubseteq}\, X_1\bullet\ H_1\big(X_1,\, \_\big)\\ &= \boldsymbol{\mu}_{\sqsubseteq}\, X_1\bullet\ \chi_G \cdot \mathsf{wp}[P](X_1) + \chi_{\neg G} \cdot f\\ &= \mathsf{wp}[\texttt{while}\,(G)\,\{P\}](f) \end{aligned}$$

and

$$\begin{aligned} \widehat{X_2} &= \boldsymbol{\mu}_{\sqsupseteq}\, X_2\bullet\ H_2\big(\boldsymbol{\mu}_{\sqsupseteq}\, X_1\bullet\ H_1(X_1,\, X_2),\, X_2\big)\\ &= \boldsymbol{\mu}_{\sqsupseteq}\, X_2\bullet\ H_2\big(\_,\, X_2\big)\\ &= \boldsymbol{\mu}_{\sqsupseteq}\, X_2\bullet\ \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g\\ &= \boldsymbol{\nu}_{\sqsubseteq}\, X_2\bullet\ \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g\\ &= \mathsf{wlp}[\texttt{while}\,(G)\,\{P\}](g) \enspace , \end{aligned}$$

which gives us in total

$$\begin{aligned} &\mathsf{cwp}[\texttt{while}\,(G)\,\{P\}](f,\, g) \;=\; \left(\widehat{X_1},\, \widehat{X_2}\right)\\ &= \big(\mathsf{wp}[\texttt{while}\,(G)\,\{P\}](f),\, \mathsf{wlp}[\texttt{while}\,(G)\,\{P\}](f)\big) \enspace . \end{aligned}$$

The argument for cwlp is completely analogous. □

C. Linearity of wp

Lemma A.2 (Linearity of wp)**.** *For any $P \in \mathsf{cpGCL}^{\boxtimes}$, any post–expectations $f, g \in \mathbb{E}$ and any non–negative real constants α, β,*

$$\mathsf{wp}[P](\alpha \cdot f + \beta \cdot g) = \alpha \cdot \mathsf{wp}[P](f) + \beta \cdot \mathsf{wp}[P](g) \enspace .$$

Proof. The proof proceeds by induction on the structure of P.

j) The Effectless Program $\texttt{skip}$*.:*

$$\begin{aligned} &\mathsf{wp}[\texttt{skip}](\alpha \cdot f + \beta \cdot g)\\ &\qquad = \alpha \cdot f + \beta \cdot g\\ &\qquad = \alpha \cdot \mathsf{wp}[\texttt{skip}](f) + \beta \cdot \mathsf{wp}[\texttt{skip}](g) \end{aligned}$$

k) The Faulty Program $\texttt{abort}$*.:*

$$\begin{aligned} &\mathsf{wp}[\texttt{abort}](\alpha \cdot f + \beta \cdot g)\\ &\qquad = \mathbf{0}\\ &\qquad = \alpha \cdot \mathsf{wp}[\texttt{abort}](f) + \beta \cdot \mathsf{wp}[\texttt{abort}](g) \end{aligned}$$

l) The Assignment $x \coloneqq E$.:

$$\begin{aligned} &\mathsf{wp}[x \coloneqq E](\alpha \cdot f + \beta \cdot g)\\ &\qquad = (\alpha \cdot f + \beta \cdot g)[x/E]\\ &\qquad = \alpha \cdot f[x/E] + \beta \cdot g[x/E]\\ &\qquad = \alpha \cdot \mathsf{wp}[x \coloneqq E](f) + \beta \cdot \mathsf{wp}[x \coloneqq E](g) \end{aligned}$$

m) The Observation $\texttt{observe}\ G$*.:*

$$\begin{aligned} &\mathsf{wp}[\texttt{observe}\ G](\alpha \cdot f + \beta \cdot g)\\ &\qquad = \chi_G \cdot (\alpha \cdot f + \beta \cdot g)\\ &\qquad = \alpha \cdot \chi_G \cdot f + \beta \cdot \chi_G \cdot g\\ &\qquad = \alpha \cdot \mathsf{wp}[\texttt{observe}\ G](f) + \beta \cdot \mathsf{wp}[\texttt{observe}\ G](g) \end{aligned}$$

n) The Concatenation $P;\, Q$.:

$$\begin{aligned} &\mathsf{wp}[P; Q](\alpha \cdot f + \beta \cdot g)\\ &\qquad = \mathsf{wp}[P](\mathsf{wp}[Q](\alpha \cdot f + \beta \cdot g))\\ &\qquad = \mathsf{wp}[P](\alpha \cdot \mathsf{wp}[Q](f) + \beta \cdot \mathsf{wp}[Q](g)) && \text{(I.H. on } Q\text{)}\\ &\qquad = \alpha \cdot \mathsf{wp}[P](\mathsf{wp}[Q](f))\\ &\qquad\quad + \beta \cdot \mathsf{wp}[P](\mathsf{wp}[Q](g)) && \text{(I.H. on } P\text{)}\\ &\qquad = \alpha \cdot \mathsf{wp}[P; Q](f) + \beta \cdot \mathsf{wp}[P; Q](g) \end{aligned}$$

o) The Conditional Choice $\texttt{ite}\,(G)\,\{P\}\,\{Q\}$*.:*

$$\begin{aligned} &\mathsf{wp}[\texttt{ite}\,(G)\,\{P\}\,\{Q\}](\alpha \cdot f + \beta \cdot g)\\ &\qquad = \chi_G \cdot \mathsf{wp}[P](\alpha \cdot f + \beta \cdot g)\\ &\qquad\quad + \chi_{\neg G} \cdot \mathsf{wp}[Q](\alpha \cdot f + \beta \cdot g)\\ &\qquad = \chi_G \cdot (\alpha \cdot \mathsf{wp}[P](f) + \beta \cdot \mathsf{wp}[P](g))\\ &\qquad\quad + \chi_{\neg G} \cdot (\alpha \cdot \mathsf{wp}[Q](f) + \beta \cdot \mathsf{wp}[Q](g)) && \text{(I.H.)}\\ &\qquad = \alpha \cdot (\chi_G \cdot \mathsf{wp}[P](f) + \chi_{\neg G} \cdot \mathsf{wp}[Q](f))\\ &\qquad\quad + \beta \cdot (\chi_G \cdot \mathsf{wp}[P](g) + \chi_{\neg G} \cdot \mathsf{wp}[Q](g))\\ &\qquad = \alpha \cdot \mathsf{wp}[\texttt{ite}\,(G)\,\{P\}\,\{Q\}](f)\\ &\qquad\quad + \beta \cdot \mathsf{wp}[\texttt{ite}\,(G)\,\{P\}\,\{Q\}](g) \end{aligned}$$

p) The Probabilistic Choice $\{P\}\ [p]\ \{Q\}$.:

$$\begin{aligned} &\mathsf{wp}[\{P\}\ [p]\ \{Q\}](\alpha \cdot f + \beta \cdot g)\\ &\qquad = p \cdot \mathsf{wp}[P](\alpha \cdot f + \beta \cdot g)\\ &\qquad\quad + (\mathbf{1} - p) \cdot \mathsf{wp}[Q](\alpha \cdot f + \beta \cdot g)\\ &\qquad = p \cdot (\alpha \cdot \mathsf{wp}[P](f) + \beta \cdot \mathsf{wp}[P](g))\\ &\qquad\quad + (\mathbf{1} - p) \cdot (\alpha \cdot \mathsf{wp}[Q](f) + \beta \cdot \mathsf{wp}[Q](g)) && \text{(I.H.)}\\ &\qquad = \alpha \cdot (p \cdot \mathsf{wp}[P](f) + (\mathbf{1} - p) \cdot \mathsf{wp}[Q](f))\\ &\qquad\quad + \beta \cdot (p \cdot \mathsf{wp}[P](g) + (\mathbf{1} - p) \cdot \mathsf{wp}[Q](g))\\ &\qquad = \alpha \cdot \mathsf{wp}[\{P\}\ [p]\ \{Q\}](f)\\ &\qquad\quad + \beta \cdot \mathsf{wp}[\{P\}\ [p]\ \{Q\}](g) \end{aligned}$$

q) The Loop $\mathtt{while}\,(G)\,\{P\}$*.:* The main idea of the proof is to show that linearity holds for the n-th unrolling of the loop and then use a continuity argument to show that the property carries over to the loop.

The fact that linearity holds for the n–unrolling of the loop is formalized by formula $H^n(\mathbf{0}) = \alpha \cdot I^n(\mathbf{0}) + \beta \cdot J^n(\mathbf{0})$, where

$$\begin{aligned}
H(X) &= \chi_G \cdot \mathsf{wp}[P](X) + \chi_{\neg G} \cdot (\alpha \cdot f + \beta \cdot g)\\
I(X) &= \chi_G \cdot \mathsf{wp}[P](X) + \chi_{\neg G} \cdot f\\
J(X) &= \chi_G \cdot \mathsf{wp}[P](X) + \chi_{\neg G} \cdot g
\end{aligned}$$

We prove this formula by induction on n. The base case $n = 0$ is immediate. For the inductive case we reason as follows

$$\begin{aligned}
&H^{n+1}(\mathbf{0})\\
&\quad = H(H^n(\mathbf{0}))\\
&\quad = H(\alpha \cdot I^n(\mathbf{0}) + \beta \cdot J^n(\mathbf{0})) && \text{(I.H. on } n\text{)}\\
&\quad = \chi_G \cdot \mathsf{wp}[P](\alpha \cdot I^n(\mathbf{0}) + \beta \cdot J^n(\mathbf{0}))\\
&\qquad + \chi_{\neg G} \cdot (\alpha \cdot f + \beta \cdot g)\\
&\quad = \chi_G \cdot (\alpha \cdot \mathsf{wp}[P](I^n(\mathbf{0})) + \beta \cdot \mathsf{wp}[P](J^n(\mathbf{0})))\\
&\qquad + \chi_{\neg G} \cdot (\alpha \cdot f + \beta \cdot g) && \text{(I.H. on } P\text{)}\\
&\quad = \alpha \cdot (\chi_G \cdot \mathsf{wp}[P](I^n(\mathbf{0})) + \chi_{\neg G} \cdot f)\\
&\qquad + \beta \cdot (\chi_G \cdot \mathsf{wp}[P](J^n(\mathbf{0})) + \chi_{\neg G} \cdot g)\\
&\quad = \alpha \cdot I(I^n(\mathbf{0})) + \beta \cdot J(J^n(\mathbf{0}))\\
&\quad = \alpha \cdot I^{n+1}(\mathbf{0}) + \beta \cdot J^{n+1}(\mathbf{0})
\end{aligned}$$

Now we turn to the proof of the main claim. We apply the Kleene Fixed Point Theorem to deduce that the least fixed points of H, I and J can be built by iteration from expectation $\mathbf{0}$ since the three transformers are continuous (due to the continuity of wp established in Lemma A.1). Then we have

$$\begin{aligned}
&\mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](\alpha \cdot f + \beta \cdot g)\\
&\quad = \bigsqcup_n H^n(\mathbf{0})\\
&\quad = \bigsqcup_n \alpha \cdot I^n(\mathbf{0}) + \beta \cdot J^n(\mathbf{0})\\
&\quad = \alpha \cdot \bigsqcup_n I^n(\mathbf{0}) + \beta \cdot \bigsqcup_n J^n(\mathbf{0})\\
&\quad = \alpha \cdot \mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](f)\\
&\qquad + \beta \cdot \mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](g) \qquad \square
\end{aligned}$$

D. *Proof of Lemma V.3*

Lemma V.3 (Elementary properties of $\underline{\mathsf{cw}}\mathsf{p}$ and $\underline{\mathsf{cwl}}\mathsf{p}$)**.** *For every $P \in \mathsf{cpGCL}^{\boxtimes}$ with at least one feasible execution (from every initial state), post–expectations $f, g \in \mathbb{E}$ and non–negative real constants α, β:*

i) $f \sqsubseteq g$ *implies* $\underline{\mathsf{cw}}\mathsf{p}[P](f) \sqsubseteq \underline{\mathsf{cw}}\mathsf{p}[P](g)$ *and likewise for* $\underline{\mathsf{cwl}}\mathsf{p}$ *(monotonicity).*

ii) $\underline{\mathsf{cw}}\mathsf{p}[P](\alpha \cdot f + \beta \cdot g) = \alpha \cdot \underline{\mathsf{cw}}\mathsf{p}[P](f) + \beta \cdot \underline{\mathsf{cw}}\mathsf{p}[P](g)$*.*

iii) $\underline{\mathsf{cw}}\mathsf{p}[P](\mathbf{0}) = \mathbf{0}$ *and* $\underline{\mathsf{cwl}}\mathsf{p}[P](\mathbf{1}) = \mathbf{1}$*.*

r) Proof of i): We do the proof for transformer $\underline{\mathsf{cw}}\mathsf{p}$; the proof for $\underline{\mathsf{cw}}\mathsf{p}$ is analogous. On view of Theorem V.1, the monotonicity of $\underline{\mathsf{cw}}\mathsf{p}$ reduces to the monotonicity of wp which follows immediately from its continuity (see Lemma A.1).

s) Proof of ii): Once again, on view of Theorem V.1, the linearity of $\underline{\mathsf{cw}}\mathsf{p}$ follows from the linearity of wp, which we prove in Lemma A.2.[7]

t) Proof of iii): Let us begin by proving that $\underline{\mathsf{cw}}\mathsf{p}[P](\mathbf{0}) = \mathbf{0}$. On account of Theorem V.1 this assertion reduces to $\mathsf{wp}[P](\mathbf{0}) = \mathbf{0}$, which has already been proved for pGCL programs (see e.g. [9]). Therefore we only have to deal with the case of $\mathtt{observe}$ statements and the claim holds since $\mathsf{wp}[\mathtt{observe}\,G](\mathbf{0}) = \chi_G \cdot \mathbf{0} = \mathbf{0}$. Finally formula $\underline{\mathsf{cwl}}\mathsf{p}[P](\mathbf{1}) = \mathbf{1}$ follows immediately from Theorem V.1. $\square$

E. *Proof of Lemma V.4 (i)*

For proving Lemma V.4 (i) we rely on the fact that allowing a bounded while–loop to be executed for an increasing number of times approximates the behavior of an unbounded while–loop. We first define bounded while–loops formally:

Definition A.1 (Bounded $\mathtt{while}$–Loops)**.** Let $P \in \mathsf{pGCL}$. Then we define:

$$\begin{aligned}
\mathtt{while}^{<0}\,(G)\,\{P\} &\triangleq \mathtt{abort}\\
\mathtt{while}^{<k+1}\,(G)\,\{P\} &\triangleq \mathtt{ite}\,(G)\,\{P^k\}\,\{\mathtt{skip}\}\\
P^k &\triangleq P;\, \mathtt{while}^{<k}\,(G)\,\{P\}
\end{aligned}$$

We can now establish that by taking the supremum on the bound k we obtain the full behavior of the unbounded while–loop:

Lemma A.4. *Let G be a predicate, $P \in \mathsf{pGCL}$, and $f \in \mathbb{E}$. Then it holds that*

$$\sup_{k \in \mathbb{N}} \mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f) = \mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](f)\ .$$

Proof. For any predicate G, any program $P \in \mathsf{pGCL}$, and any expectation $f \in \mathbb{E}$ let

$$F(X) \ = \ \chi_G \cdot \mathsf{wp}[P](X) + \chi_{\neg G} \cdot f\ .$$

We first show by induction on $k \in \mathbb{N}$ that

$$\mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f) \ = \ F^k(\mathbf{0})\ .$$

For the induction base we have $k = 0$. In that case we have

$$\begin{aligned}
&\mathsf{wp}[\mathtt{while}^{<0}\,(G)\,\{P\}](f)\\
=\ &\mathsf{wp}[\mathtt{abort}](f)\\
=\ &\mathbf{0}\\
=\ &F^0(\mathbf{0})\ .
\end{aligned}$$

As the induction hypothesis assume now that

$$\mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f) \ = \ F^k(\mathbf{0})(f)$$

[7] We cannot adopt the results from the original work [9] because their analyses is restricted to bounded expectations.

holds for some arbitrary but fixed k. Then for the induction step we have

$$\begin{aligned}
& \mathsf{wp}[\mathtt{while}^{<k+1}\,(G)\,\{P\}](f) \\
=\ & \mathsf{wp}[P;\ \mathtt{ite}\,(G)\,\{\mathtt{while}^{<k}\,(G)\,\{P\}\}\,\{\mathtt{skip}\}](f) \\
=\ & (\chi_G \cdot \mathsf{wp}[P] \circ \mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}] \\
& \quad + \chi_{\neg G} \cdot \mathsf{wp}[\mathtt{skip}])(f) \\
=\ & \chi_G \cdot \mathsf{wp}[P](\mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f)) \\
& \quad + \chi_{\neg G} \cdot \mathsf{wp}[\mathtt{skip}](f) \\
=\ & \chi_G \cdot \mathsf{wp}[P](F^k(\mathbf{0})) + \chi_{\neg G} \cdot f \qquad \text{(I.H.)} \\
=\ & F^{k+1}(\mathbf{0})(f)\ .
\end{aligned}$$

We have by now established that

$$\mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f) \ = \ F^k(\mathbf{0})$$

holds for every $k \in \mathbb{N}$. Ergo, we can also establish that

$$\begin{aligned}
& \sup_{k\in\mathbb{N}} \mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f) \\
=\ & \sup_{k\in\mathbb{N}} F^k(\mathbf{0}) \\
=\ & \boldsymbol{\mu} X.\ F(X) \\
=\ & \mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](f)\ . \qquad \square
\end{aligned}$$

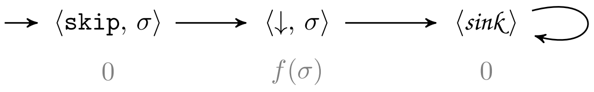
With Lemma A.4 in mind, we can now restate and prove Lemma V.4 (i):

Lemma V.4 (i). *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $f \in \mathbb{E}, g \in \mathbb{E}_{\leq 1}$, and $\sigma \in \mathbb{S}$:*

$$\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}\,(\Diamond\langle \mathit{sink}\rangle) = \mathsf{wp}[P](f)(\sigma)$$

Proof. The proof goes by induction over all $\mathsf{cpGCL}^{\boxtimes}$ programs. For the induction base we have:

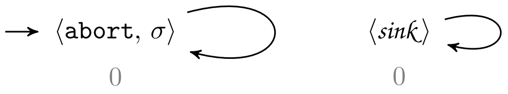
The Effectless Program `skip`**.** The RMC for this program is of the following form:[8]



In the above RMC we have $\Pi := \mathsf{Paths}(\langle \mathtt{skip},\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \mathtt{skip},\, \sigma\rangle \to \langle \downarrow,\, \sigma\rangle \to \langle \mathit{sink}\rangle$. Then we have for the expected reward:

$$\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\mathtt{skip}]\!]}\,(\Diamond \mathit{sink}) \\
=\ & \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) \\
=\ & \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) \\
=\ & 1 \cdot f(\sigma) \\
=\ & f(\sigma) \\
=\ & \mathsf{wp}[\mathtt{skip}](f)(\sigma)
\end{aligned}$$

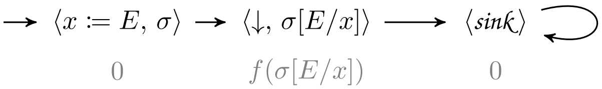
The Faulty Program `abort`**.** The RMC for this program is of the following form:



In this RMC we have $\Pi := \mathsf{Paths}(\langle \mathtt{abort},\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \emptyset$. Then we have for the expected reward:

$$\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\mathtt{abort}]\!]}\,(\Diamond \mathit{sink}) \\
=\ & \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) \\
=\ & \sum_{\hat{\pi}\in\emptyset} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) \\
=\ & 0 \\
=\ & \mathbf{0}(\sigma) \\
=\ & \mathsf{wp}[\mathtt{abort}](f)(\sigma)
\end{aligned}$$

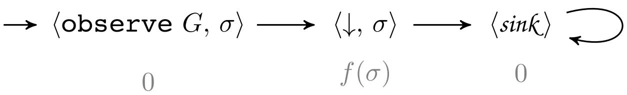
The Assignment $x \coloneqq E$**.** The RMC for this program is of the following form:



In this RMC we have $\Pi := \mathsf{Paths}(\langle x \coloneqq E,\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle x \coloneqq E,\, \sigma\rangle \to \langle \downarrow,\, \sigma[E/x]\rangle \to \langle \mathit{sink}\rangle$. Then we have for the expected reward:

$$\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![x\coloneqq E]\!]}\,(\Diamond \mathit{sink}) \\
=\ & \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) \\
=\ & \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) \\
=\ & 1 \cdot f(\sigma[E/x]) \\
=\ & f(\sigma[E/x]) \\
=\ & f[E/x](\sigma) \\
=\ & \mathsf{wp}[x \coloneqq E](f)(\sigma)
\end{aligned}$$

The Observation `observe` G**.** For this program there are two cases: In Case 1 we have $\sigma \models G$, so we have $\chi_G(\sigma) = 1$. The RMC in this case is of the following form:

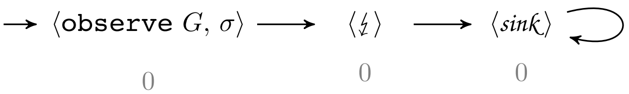


In this RMC we have $\Pi := \mathsf{Paths}(\langle \mathtt{observe}\ G,\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \mathtt{observe}\ G,\, \sigma\rangle \to \langle \downarrow,\, \sigma\rangle \to \langle \mathit{sink}\rangle$. Then we have for the expected reward:

$$\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\mathtt{observe}\ G]\!]}\,(\Diamond \mathit{sink}) \\
=\ & \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) \\
=\ & \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) \\
=\ & 1 \cdot f(\sigma) \\
=\ & \chi_G(\sigma) \cdot f(\sigma) \\
=\ & (\chi_G \cdot f)(\sigma) \\
=\ & \mathsf{wp}[\mathtt{observe}\ G](f)(\sigma)
\end{aligned}$$

[8] If transitions have probability 1, we omit this in our figures. Moreover, all states—with the exception of $\langle \mathit{sink}\rangle$—are left out if they are not reachable from the initial state.

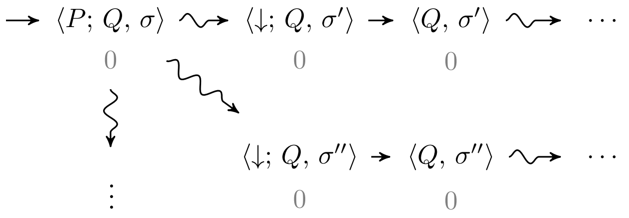
In Case 2 we have $\sigma \not\models G$, so we have $\chi_G(\sigma) = 0$. The RMC in this case is of the following form:



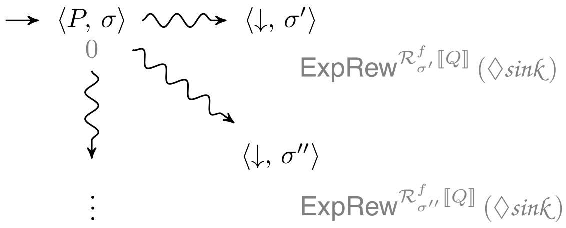
In this RMC we have $\Pi := \mathsf{Paths}(\langle \texttt{observe}\ G,\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \texttt{observe}\ G,\, \sigma\rangle \rightarrow \langle \lightning \rangle \rightarrow \langle \mathit{sink}\rangle$. Then for the expected reward we also have:

$$
\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\texttt{observe}\ G]\!]}(\Diamond \mathit{sink}) \\
= {} & \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) \\
= {} & \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) \\
= {} & 1 \cdot 0 \\
= {} & 0 \\
= {} & 0 \cdot f(\sigma) \\
= {} & \chi_G(\sigma) \cdot f(\sigma) \\
= {} & (\chi_G \cdot f)(\sigma) \\
= {} & \mathsf{wp}[\texttt{observe}\ G](f)(\sigma)
\end{aligned}
$$

The Concatenation $P;\, Q$**.** For this program the RMC is of the following form:



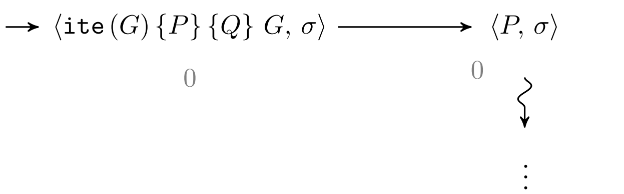
In this RMC every path in $\mathsf{Paths}(\langle P;\, Q,\, \sigma\rangle,\, \langle \mathit{sink}\rangle)$ starts with $\langle P;\, Q,\, \sigma\rangle$, eventually reaches $\langle \downarrow;\, Q,\, \sigma'\rangle$, and then immediately after that reaches $\langle Q,\, \sigma'\rangle$ which is the initial state of $\mathcal{R}^f_{\sigma'}[\![Q]\!]$ for which the expected reward is given by $\mathsf{ExpRew}^{\mathcal{R}^f_{\sigma'}[\![Q]\!]}(\Diamond \mathit{sink})$. By this insight we can transform the above RMC into the RMC with equal expected reward below:



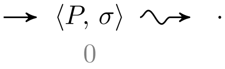
But the above RMC is exactly $\mathcal{R}_\sigma^{\lambda\tau.\,\mathsf{ExpRew}^{\mathcal{R}^f_\tau[\![Q]\!]}(\Diamond \mathit{sink})}[\![P]\!]$ for which the expected reward is also known by the induction hypothesis. So we have

$$
\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P;\, Q]\!]}(\Diamond \mathit{sink}) \\
= {} & \mathsf{ExpRew}^{\mathcal{R}_\sigma^{\lambda\tau.\,\mathsf{ExpRew}^{\mathcal{R}^f_\tau[\![Q]\!]}(\Diamond \mathit{sink})}[\![P]\!]}(\Diamond \mathit{sink}) \\
= {} & \mathsf{ExpRew}^{\mathcal{R}_\sigma^{\mathsf{wp}[Q](f)}[\![P]\!]}(\Diamond \mathit{sink}) && \text{(I.H. on } Q\text{)} \\
= {} & \mathsf{wp}[P](\mathsf{wp}[Q](f))(\sigma) && \text{(I.H. on } P\text{)} \\
= {} & \mathsf{wp}[P;\, Q](f)
\end{aligned}
$$

The Conditional Choice $\texttt{ite}\,(G)\,\{P\}\,\{Q\}$**.** For this program there are two cases: In Case 1 we have $\sigma \models G$, so we have $\chi_G(\sigma) = 1$ and $\chi_{\neg G}(\sigma) = 0$. The RMC in this case is of the following form:



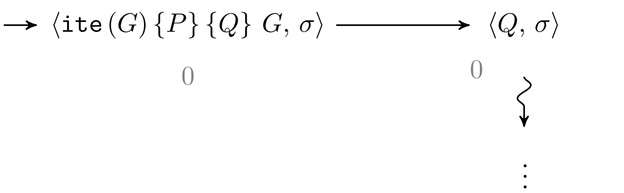
In this RMC every path in $\mathsf{Paths}(\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle,\, \langle \mathit{sink}\rangle)$ starts with $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle \rightarrow \langle P,\, \sigma\rangle \rightarrow \cdots$. As the state $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle$ collects zero reward, the expected reward of the above RMC is equal to the expected reward of the following RMC:



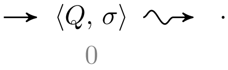
But the above RMC is exactly $\mathcal{R}^f_\sigma[\![P]\!]$ for which the expected reward is known by the induction hypothesis. So we have

$$
\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\
= {} & \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink}) \\
= {} & \mathsf{wp}[P](f)(\sigma) && \text{(I.H.)} \\
= {} & 1 \cdot \mathsf{wp}[P](f)(\sigma) + 0 \cdot \mathsf{wp}[Q](f)(\sigma) \\
= {} & \chi_G(\sigma) \cdot \mathsf{wp}[P](f)(\sigma) + \chi_{\neg G}(\sigma) \cdot \mathsf{wp}[Q](f)(\sigma) \\
= {} & \mathsf{wp}[\texttt{ite}\,(G)\,\{P\}\,\{Q\}](f)(\sigma)\ .
\end{aligned}
$$

In Case 2 we have $\sigma \not\models G$, so we have $\chi_G(\sigma) = 0$ and $\chi_{\neg G}(\sigma) = 1$. The RMC in this case is of the following form:



In this RMC every path in $\mathsf{Paths}(\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle,\, \langle \mathit{sink}\rangle)$ starts with $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle \rightarrow \langle Q,\, \sigma\rangle \rightarrow \cdots$. As the state $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle$ collects zero reward, the expected reward of the above RMC is equal to the expected reward of the following RMC:

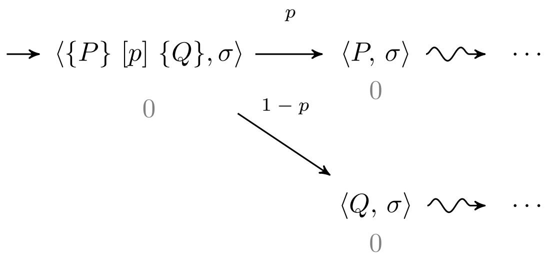


But the above RMC is exactly $\mathcal{R}^f_\sigma[\![Q]\!]$ for which the expected reward is known by the induction hypothesis. So we also have

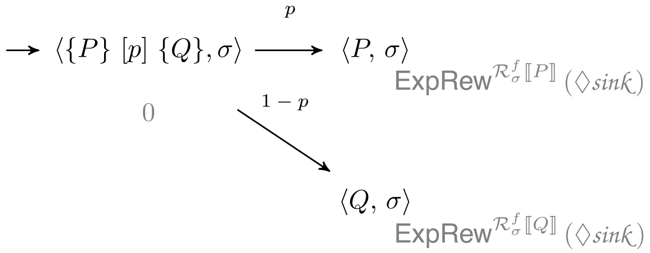
$$
\begin{aligned}
& \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\
= {} & \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}(\Diamond \mathit{sink}) \\
= {} & \mathsf{wp}[Q](f)(\sigma) && \text{(I.H.)}
\end{aligned}
$$

$$\begin{aligned} &= 0 \cdot \mathsf{wp}[P](f)(\sigma) + 1 \cdot \mathsf{wp}[Q](f)(\sigma) \\ &= \chi_G(\sigma) \cdot \mathsf{wp}[P](f)(\sigma) + \chi_{\neg G}(\sigma) \cdot \mathsf{wp}[Q](f)(\sigma) \\ &= \mathsf{wp}[\mathtt{ite}\,(G)\,\{P\}\,\{Q\}](f)(\sigma)\ . \end{aligned}$$

The Probabilistic Choice $\{P\}\ [p]\ \{Q\}$. For this program the RMC is of the following form:


$\langle \{P\}\ [p]\ \{Q\}, \sigma\rangle$
p
$\langle P, \sigma\rangle$
$\cdots$
0
0
$1-p$
$\langle Q, \sigma\rangle$
$\cdots$
0

In this RMC every path in $\mathsf{Paths}(\langle \{P\}\ [p]\ \{Q\}, \sigma\rangle, \langle \mathit{sink}\rangle)$ starts with $\langle \{P\}\ [p]\ \{Q\}, \sigma\rangle$ and immediately after that reaches $\langle P, \sigma\rangle$ with probability p or $\langle Q, \sigma\rangle$ with probability $1-p$. $\langle P, \sigma\rangle$ is the initial state of $\mathcal{R}^f_\sigma[\![P]\!]$ and $\langle Q, \sigma\rangle$ is the initial state of $\mathcal{R}^f_\sigma[\![Q]\!]$. By this insight we can transform the above RMC into the RMC with equal expected reward below:


$\langle \{P\}\ [p]\ \{Q\}, \sigma\rangle$
p
$\langle P, \sigma\rangle$
$\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink})$
0
$1-p$
$\langle Q, \sigma\rangle$
$\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}(\Diamond \mathit{sink})$

The expected reward of the above RMC is given by $p \cdot \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink}) + (1-p) \cdot \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}(\Diamond \mathit{sink})$, so in total we have for the expected reward:

$$\begin{aligned} &\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\{P\}\ [p]\ \{Q\}]\!]}(\Diamond \mathit{sink}) \\ &= p \cdot \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink}) \\ &\qquad + (1-p) \cdot \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}(\Diamond \mathit{sink}) \\ &= p \cdot \mathsf{wp}[P](f)(\sigma) + (1-p) \cdot \mathsf{wp}[Q](f)(\sigma) \qquad \text{(I.H.)} \\ &= \mathsf{wp}[\{P\}\ [p]\ \{Q\}](f)\ . \end{aligned}$$

The Loop $\mathtt{while}\,(G)\,\{Q\}$**.** By Lemma A.4 we have

$$\mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](f) \;=\; \sup_{k \in \mathbb{N}} \mathsf{wp}[\mathtt{while}^{<k}\,(G)\,\{P\}](f)$$

and as $\mathtt{while}^{<k}\,(G)\,\{P\}$ is a purely syntactical construct (made up from `abort`, `skip`, conditional choice, and P) we can (using what we have already established on `abort`, `skip`, conditional choice, and using the induction hypothesis on P) also establish that

$$\begin{aligned} &\mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](f) \\ &= \sup_{k \in \mathbb{N}} \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\mathtt{while}^{<k}\,(G)\,\{P\}]\!]}(\Diamond \mathit{sink})\ . \end{aligned}$$

It is now left to show that

$$\sup_{k \in \mathbb{N}} \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\mathtt{while}^{<k}\,(G)\,\{P\}]\!]}(\Diamond \mathit{sink}) \qquad (5)$$

$$= \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\mathtt{while}\,(G)\,\{P\}]\!]}(\Diamond \mathit{sink})\ . \qquad (6)$$

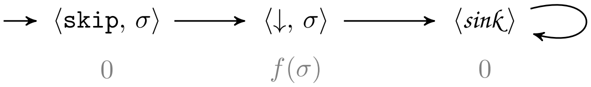
While the above is intuitively evident, it is a tedious and technically involved task to prove it. Herefore we just provide an intuition thereof: For showing (5) $\leq$ (6), we know that every path in the RMDP $\mathcal{R}^f_\sigma[\![\mathtt{while}^{<k}\,(G)\,\{P\}]\!]$ either terminates properly or is prematurely aborted (yielding 0 reward) due to the fact that the bound of less than k loop iterations was reached. The RMDP $\mathcal{R}^f_\sigma[\![\mathtt{while}\,(G)\,\{P\}]\!]$ for the unbounded while–loop does not prematurely abort executions, so left–hand–side is upper bounded by the right–hand–side of the equation. For showing (5) $\geq$ (6), we know that a path that collects positive reward is necessarily finite. Therefore there exists some $k \in \mathbb{N}$ such that $\mathcal{R}^f_\sigma[\![\mathtt{while}^{<k}\,(G)\,\{P\}]\!]$ includes this path. Taking the supremum over k we eventually include every path in $\mathcal{R}^f_\sigma[\![\mathtt{while}\,(G)\,\{P\}]\!]$ that collects positive reward. □

F. Proof of Lemma V.4 (ii)

Lemma V.4 (ii). *For* $P \in \mathsf{cpGCL}^{\boxtimes}$*,* $f \in \mathbb{E}, g \in \mathbb{E}_{\leq 1}$*, and* $\sigma \in \mathbb{S}$*:*

$$\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![P]\!]}(\Diamond\langle \mathit{sink}\rangle) = \mathsf{wlp}[P](g)(\sigma)$$

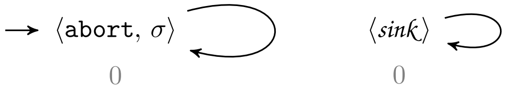
Proof. The proof goes by induction over all $\mathsf{cpGCL}^{\boxtimes}$ programs. For the induction base we have: **The Effectless Program** `skip`**.** The RMC for this program is of the following form:


$\langle \mathtt{skip}, \sigma\rangle$
$\langle \downarrow, \sigma\rangle$
$\langle \mathit{sink}\rangle$
0
$f(\sigma)$
0

In this RMC we have $\Pi := \mathsf{Paths}(\langle \mathtt{skip}, \sigma\rangle, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \mathtt{skip}, \sigma\rangle \to \langle \downarrow, \sigma\rangle \to \langle \mathit{sink}\rangle$. Then we have for the liberal expected reward:

$$\begin{aligned} &\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![\mathtt{skip}]\!]}(\Diamond \mathit{sink}) \\ &= \sum_{\hat{\pi} \in \Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \mathit{sink}\rangle) \\ &= \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + 0 \\ &= 1 \cdot g(\sigma) \\ &= g(\sigma) \\ &= \mathsf{wlp}[\mathtt{skip}](g)(\sigma) \end{aligned}$$

The Faulty Program `abort`**.** The RMC for this program is of the following form:

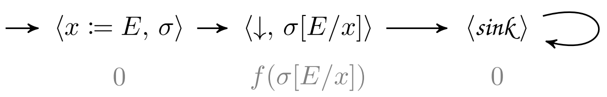

$\langle \mathtt{abort}, \sigma\rangle$
$\langle \mathit{sink}\rangle$
0
0

In this RMC we have $\Pi := \mathsf{Paths}(\langle \mathtt{abort}, \sigma\rangle, \langle \mathit{sink}\rangle) = \emptyset$. Then we have for the liberal expected reward:

$$\begin{aligned} &\mathsf{ExpRew}^{\mathcal{R}^g_\sigma[\![\mathtt{abort}]\!]}(\Diamond \mathit{sink}) \\ &= \sum_{\hat{\pi} \in \Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \mathit{sink}\rangle) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{\hat{\pi}\in\emptyset} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + 1 \\
&= 0 + 1 \\
&= 1 \\
&= \mathbf{1}(\sigma) \\
&= \mathsf{wlp}[\mathtt{abort}](g)(\sigma)
\end{aligned}
$$

The Assignment $x \coloneqq E$. The RMC for this program is of the following form:

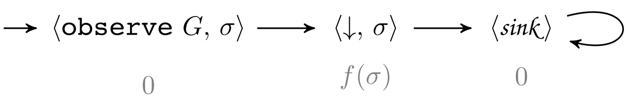


$\langle x \coloneqq E,\, \sigma\rangle$
$\langle \downarrow,\, \sigma[E/x]\rangle$
$\langle \mathit{sink}\rangle$
0
$f(\sigma[E/x])$
0

In this RMC we have $\Pi \coloneqq \mathsf{Paths}(\langle x \coloneqq E,\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle x \coloneqq E,\, \sigma\rangle \to \langle \downarrow,\, \sigma[E/x]\rangle \to \langle \mathit{sink}\rangle$. Then we have for the liberal expected reward:

$$
\begin{aligned}
&\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![x:=E]\!]}\,(\Diamond \mathit{sink}) \\
&= \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \mathit{sink}\rangle) \\
&= \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) + 0 \\
&= 1 \cdot g(\sigma[E/x]) \\
&= g(\sigma[E/x]) \\
&= g[E/x](\sigma) \\
&= \mathsf{wlp}[x \coloneqq E](g)(\sigma)
\end{aligned}
$$

The Observation $\mathtt{observe}\ G$. For this program there are two cases: In Case 1 we have $\sigma \models G$, so we have $\chi_G(\sigma) = 1$. The RMC in this case is of the following form:

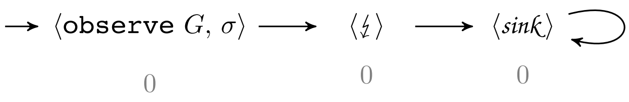


$\langle \mathtt{observe}\ G,\, \sigma\rangle$
$\langle \downarrow,\, \sigma\rangle$
$\langle \mathit{sink}\rangle$
0
$f(\sigma)$
0

In this RMC we have $\Pi \coloneqq \mathsf{Paths}(\langle \mathtt{observe}\ G,\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \mathtt{observe}\ G,\, \sigma\rangle \to \langle \downarrow,\, \sigma\rangle \to \langle \mathit{sink}\rangle$. Then we have for the liberal expected reward:

$$
\begin{aligned}
&\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![\mathtt{observe}\ G]\!]}\,(\Diamond \mathit{sink}) \\
&= \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \mathit{sink}\rangle) \\
&= \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) + 0 \\
&= 1 \cdot g(\sigma) \\
&= \chi_G(\sigma) \cdot g(\sigma) \\
&= (\chi_G \cdot g)(\sigma) \\
&= \mathsf{wlp}[\mathtt{observe}\ G](g)(\sigma)
\end{aligned}
$$

In Case 2 we have $\sigma \not\models G$, so we have $\chi_G(\sigma) = 0$. The RMC in this case is of the following form:

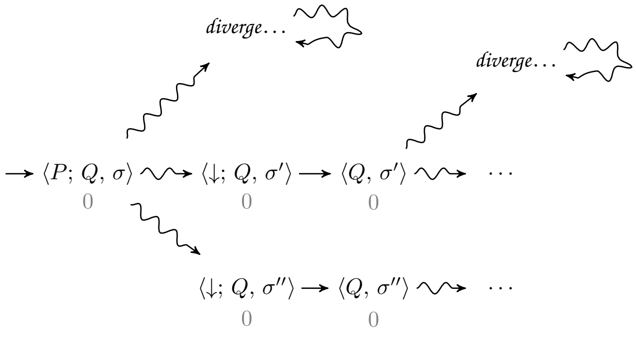


$\langle \mathtt{observe}\ G,\, \sigma\rangle$
$\langle \lightning \rangle$
$\langle \mathit{sink}\rangle$
0
0
0

In this RMC we have $\Pi \coloneqq \mathsf{Paths}(\langle \mathtt{observe}\ G,\, \sigma\rangle,\, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \mathtt{observe}\ G,\, \sigma\rangle \to \langle \lightning \rangle \to \langle \mathit{sink}\rangle$. Then we have for the liberal expected reward:

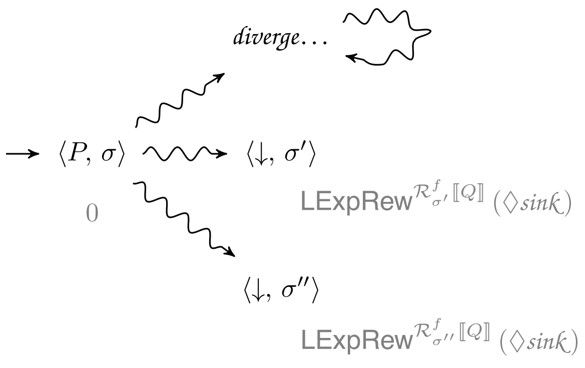
$$
\begin{aligned}
&\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![\mathtt{observe}\ G]\!]}\,(\Diamond \mathit{sink}) \\
&= \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \mathit{sink}\rangle) \\
&= \Pr(\hat{\pi}_1) \cdot r(\hat{\pi}_1) + 0 \\
&= 1 \cdot 0 \\
&= 0 \\
&= 0 \cdot g(\sigma) \\
&= \chi_G(\sigma) \cdot g(\sigma) \\
&= (\chi_G \cdot g)(\sigma) \\
&= \mathsf{wlp}[\mathtt{observe}\ G](g)(\sigma)
\end{aligned}
$$

The Concatenation $P;\, Q$. For this program the RMC is of the following form:



diverge…
diverge…
$\langle P;\, Q,\, \sigma\rangle$
$\langle \downarrow;\, Q,\, \sigma'\rangle$
$\langle Q,\, \sigma'\rangle$
$\cdots$
0
0
0
$\langle \downarrow;\, Q,\, \sigma''\rangle$
$\langle Q,\, \sigma''\rangle$
$\cdots$
0
0

In this RMC every path in $\mathsf{Paths}(\langle P;\, Q,\, \sigma\rangle,\, \langle \mathit{sink}\rangle)$ starts with $\langle P;\, Q,\, \sigma\rangle$, eventually reaches $\langle \downarrow;\, Q,\, \sigma\rangle$, and then immediately after that reaches $\langle Q,\, \sigma\rangle$ which is the initial state of $\mathcal{R}^g_\sigma[\![Q]\!]$. Every diverging path either diverges because the program P diverges or because the program Q diverges. If we attempt to make the RMC smaller (while preserving the liberal expected reward) by cutting it off at states of the form $\langle \downarrow;\, Q,\, \tau\rangle$, we have to assign to them the liberal expected reward $\mathsf{LExpRew}^{\mathcal{R}^g_\tau[\![Q]\!]}\,(\Diamond \mathit{sink})$ in order to not loose the non–termination probability caused by Q. By this insight we can now transform the above RMC into the RMC with equal liberal expected reward below:



diverge…
$\langle P,\, \sigma\rangle$
$\langle \downarrow,\, \sigma'\rangle$
0
$\mathsf{LExpRew}^{\mathcal{R}^f_{\sigma'}[\![Q]\!]}\,(\Diamond \mathit{sink})$
$\langle \downarrow,\, \sigma''\rangle$
$\mathsf{LExpRew}^{\mathcal{R}^f_{\sigma''}[\![Q]\!]}\,(\Diamond \mathit{sink})$

But the above RMC is exactly $\mathcal{R}_\sigma^{\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![Q]\!]}(\Diamond \mathit{sink})}[\![P]\!]$ for which the liberal expected reward is known by the induction hypothesis. So we have for the liberal expected reward:

$$\begin{aligned} & \mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![P;\,Q]\!]}(\Diamond \mathit{sink}) \\ ={}& \mathsf{LExpRew}^{\mathcal{R}_\sigma^{\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![Q]\!]}(\Diamond \mathit{sink})}[\![P]\!]}(\Diamond \mathit{sink}) \\ ={}& \mathsf{LExpRew}^{\mathcal{R}_\sigma^{\mathsf{wlp}[Q](g)}[\![P]\!]}(\Diamond \mathit{sink}) && \text{(I.H. on } Q\text{)} \\ ={}& \mathsf{wlp}[P](\mathsf{wlp}[Q](g))(\sigma) && \text{(I.H. on } P\text{)} \\ ={}& \mathsf{wlp}[P;\,Q](g)\ . \end{aligned}$$

The Conditional Choice $\texttt{ite}\,(G)\,\{P\}\,\{Q\}$**.** For this program there are two cases: In Case 1 we have $\sigma \models G$, so we have $\chi_G(\sigma) = 1$ and $\chi_{\neg G}(\sigma) = 0$. The RMC in this case is of the following form:

$$\begin{tikzcd} {} \arrow[r] & \langle \texttt{ite}\,(G)\,\{P\}\,\{Q\}\ G,\,\sigma\rangle \arrow[r] & \langle P,\,\sigma\rangle \arrow[d, squiggly] \\ & 0 & 0 \quad \vdots \end{tikzcd}$$

As the state $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\,\sigma\rangle$ collects zero reward, the expected reward of the above RMC is equal to the expected reward of the following RMC:

$$\begin{tikzcd} {} \arrow[r] & \langle P,\,\sigma\rangle \arrow[r, squiggly] & \cdots \\ & 0 & \end{tikzcd}$$

But the above RMC is exactly $\mathcal{R}^g_\sigma[\![P]\!]$ for which the expected reward is known by Lemma . A similar argument can be applied to the probability of not eventually reaching $\langle \mathit{sink}\rangle$. So we have for the liberal expected reward:

$$\begin{aligned} & \mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\ ={}& \mathsf{ExpRew}^{\mathcal{R}^g_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\ & + \Pr{}^{\mathcal{R}^g_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\neg\Diamond\langle \mathit{sink}\rangle) \\ ={}& \mathsf{ExpRew}^{\mathcal{R}^g_\sigma[\![P]\!]}(\Diamond \mathit{sink}) + \Pr{}^{\mathcal{R}^g_\sigma[\![P]\!]}(\neg\Diamond\langle \mathit{sink}\rangle) \\ ={}& \mathsf{wlp}[P](g)(\sigma) && \text{(I.H.)} \\ ={}& 1\cdot \mathsf{wlp}[P](g)(\sigma) + 0\cdot \mathsf{wlp}[Q](g)(\sigma) \\ ={}& \chi_G(\sigma)\cdot \mathsf{wlp}[P](g)(\sigma) + \chi_{\neg G}(\sigma)\cdot \mathsf{wlp}[Q](g)(\sigma) \\ ={}& \mathsf{wlp}[\texttt{ite}\,(G)\,\{P\}\,\{Q\}](g)(\sigma)\ . \end{aligned}$$

In Case 2 we have $\sigma \not\models G$, so we have $\chi_G(\sigma) = 0$ and $\chi_{\neg G}(\sigma) = 1$. The RMC in this case is of the following form:

$$\begin{tikzcd} {} \arrow[r] & \langle \texttt{ite}\,(G)\,\{P\}\,\{Q\}\ G,\,\sigma\rangle \arrow[r] & \langle Q,\,\sigma\rangle \arrow[d, squiggly] \\ & 0 & 0 \quad \vdots \end{tikzcd}$$

In this RMC every path in $\mathsf{Paths}(\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\,\sigma\rangle,\,\langle \mathit{sink}\rangle)$ starts with $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\,\sigma\rangle \to \langle Q,\,\sigma\rangle \to \cdots$. As the state $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\,\sigma\rangle$ collects zero reward, the expected reward of the above RMC is equal to the expected reward of the following RMC:

$$\begin{tikzcd} {} \arrow[r] & \langle Q,\,\sigma\rangle \arrow[r, squiggly] & \cdots \\ & 0 & \end{tikzcd}$$

But the above RMC is exactly $\mathcal{R}^g_\sigma[\![Q]\!]$ for which the expected reward is known by the induction hypothesis. A similar argument can be applied to the probability of not eventually reaching $\langle \mathit{sink}\rangle$. So we also have for the liberal expected reward:

$$\begin{aligned} & \mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\ ={}& \mathsf{ExpRew}^{\mathcal{R}^g_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\ & + \Pr{}^{\mathcal{R}^g_\sigma[\![\texttt{ite}\,(G)\,\{P\}\,\{Q\}]\!]}(\neg\Diamond\langle \mathit{sink}\rangle) \\ ={}& \mathsf{ExpRew}^{\mathcal{R}^g_\sigma[\![Q]\!]}(\Diamond \mathit{sink}) + \Pr{}^{\mathcal{R}^g_\sigma[\![Q]\!]}(\neg\Diamond\langle \mathit{sink}\rangle) \\ ={}& \mathsf{wlp}[Q](g)(\sigma) && \text{(I.H.)} \\ ={}& 0\cdot \mathsf{wlp}[P](g)(\sigma) + 1\cdot \mathsf{wlp}[Q](g)(\sigma) \\ ={}& \chi_G(\sigma)\cdot \mathsf{wlp}[P](g)(\sigma) + \chi_{\neg G}(\sigma)\cdot \mathsf{wlp}[Q](g)(\sigma) \\ ={}& \mathsf{wlp}[\texttt{ite}\,(G)\,\{P\}\,\{Q\}](g)(\sigma)\ . \end{aligned}$$

The Probabilistic Choice $\{P\}\ [p]\ \{Q\}$**.** For this program the RMC is of the following form:

$$\begin{tikzcd} {} \arrow[r] & \langle \{P\}\ [p]\ \{Q\},\sigma\rangle \arrow[r, "p"] \arrow[dr, "1-p"] & \langle P,\,\sigma\rangle \arrow[r, squiggly] & \cdots \\ & 0 & 0 & \\ & & \langle Q,\,\sigma\rangle \arrow[r, squiggly] & \cdots \\ & & 0 & \end{tikzcd}$$

In this RMC every path in $\mathsf{Paths}(\langle \{P\}\ [p]\ \{Q\},\,\sigma\rangle,\,\langle \mathit{sink}\rangle)$ starts with $\langle \{P\}\ [p]\ \{Q\},\,\sigma\rangle$ and immediately after that reaches $\langle P,\,\sigma\rangle$ with probability p or $\langle Q,\,\sigma\rangle$ with probability $1-p$. $\langle P,\,\sigma\rangle$ is the initial state of $\mathcal{R}^f_\sigma[\![P]\!]$ and $\langle Q,\,\sigma\rangle$ is the initial state of $\mathcal{R}^f_\sigma[\![Q]\!]$. The same holds for all paths that do not eventually reach $\langle \mathit{sink}\rangle$. By this insight we can transform the above RMC into the RMC with equal liberal expected reward below:

$$\begin{tikzcd} {} \arrow[r] & \langle \{P\}\ [p]\ \{Q\},\sigma\rangle \arrow[r, "p"] \arrow[dr, "1-p"] & \langle P,\,\sigma\rangle \\ & 0 & \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink}) \\ & & \langle Q,\,\sigma\rangle \\ & & \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}(\Diamond \mathit{sink}) \end{tikzcd}$$

The liberal expected reward of the above RMC is given by $p\cdot\mathsf{LExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink}) + (1-p)\cdot\mathsf{LExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}(\Diamond \mathit{sink})$, so in total we have for the liberal expected reward:

$$\begin{aligned} & \mathsf{LExpRew}^{\mathcal{R}^f_\sigma[\![\{P\}\,[p]\,\{Q\}]\!]}(\Diamond \mathit{sink}) \\ ={}& p\cdot \mathsf{LExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink}) \end{aligned}$$

$$\begin{aligned} &\quad + (1-p)\cdot \mathsf{LExpRew}^{\mathcal{R}^f_\sigma[\![Q]\!]}\,(\Diamond \mathit{sink}) \\ &= p\cdot \mathsf{wlp}[P](f)(\sigma) + (1-p)\cdot \mathsf{wlp}[Q](f)(\sigma) \qquad \text{(I.H.)} \\ &= \mathsf{wlp}[\{P\}\ [p]\ \{Q\}](f)\ . \end{aligned}$$

The Loop `while` $(G)\,\{Q\}$**.**

The argument is dual to the case for the (non–liberal) expected reward. □

G. Proof of Lemma V.5

Lemma V.5. *For* $P \in \mathsf{cpGCL}^{\boxtimes}$*,* $g \in \mathbb{E}_{\leq 1}$*, and* $\sigma \in \mathbb{S}$*:*

$$\mathrm{Pr}^{\mathcal{R}^g_\sigma[\![P]\!]}(\neg\Diamond ↯) \;=\; \mathsf{wlp}[P](\mathbf{1})(\sigma)\ .$$

Proof. First, observe that paths on reaching ✓or ↯ immediately move to the state ⟨*sink*⟩. Moreover, all paths that never visit ↯ either (a) visit a terminal ✓–state (which are the only states that can possibly collect positive reward) or (b) diverge and never reach ⟨*sink*⟩ and therefore neither reach ✓nor ↯. Furthermore the set of "(a)–paths" and the set of "(b)–paths" are disjoint. Thus:

$$\begin{aligned} &\mathrm{Pr}^{\mathcal{R}^f_\sigma[\![P]\!]}(\neg\Diamond ↯) \\ &= \mathrm{Pr}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \checkmark) + \mathrm{Pr}^{\mathcal{R}^f_\sigma[\![P]\!]}(\neg\Diamond \mathit{sink}) \end{aligned}$$

and by assigning reward one to every ✓–state, and zero to all other states, we can turn the probability measure into an expected reward, yielding

$$= \mathsf{ExpRew}^{\mathcal{R}^{\mathbf{1}}_\sigma[\![P]\!]}\,(\Diamond \checkmark) + \mathrm{Pr}^{\mathcal{R}^g_\sigma[\![P]\!]}(\neg\Diamond \mathit{sink})$$

As every path that reaches sink over a ↯–state cumulates zero reward, we finally get:

$$\begin{aligned} &= \mathsf{ExpRew}^{\mathcal{R}^{\mathbf{1}}_\sigma[\![P]\!]}\,(\Diamond \mathit{sink}) + \mathrm{Pr}^{\mathcal{R}^g_\sigma[\![P]\!]}(\neg\Diamond \mathit{sink}) \\ &= \mathsf{LExpRew}^{\mathcal{R}^{\mathbf{1}}_\sigma[\![P]\!]}\,(\Diamond \mathit{sink}) \\ &= \mathsf{wlp}[P](\mathbf{1}) \qquad \text{(Lemma V.4)} \end{aligned}$$

□

H. Proof of Theorem V.6

Theorem V.6 (Correspondence theorem)**.** *For* $P \in \mathsf{cpGCL}^{\boxtimes}$*,* $f \in \mathbb{E}$*,* $g \in \mathbb{E}_{\leq 1}$ *and* $\sigma \in \mathbb{S}$*,*

$$\begin{aligned} \mathsf{CExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}\,(\Diamond \mathit{sink} \mid \neg\Diamond ↯) \;&=\; \underline{\mathsf{cw}}\mathsf{p}[P](f)(\sigma) \\ \mathsf{CLExpRew}^{\mathcal{R}^g_\sigma[\![P]\!]}\,(\Diamond \mathit{sink} \mid \neg\Diamond ↯) \;&=\; \underline{\mathsf{cwl}}\mathsf{p}[P](g)(\sigma)\ . \end{aligned}$$

Proof. We prove only the first equation. The proof of the second equation goes along the same arguments.

$$\begin{aligned} &\mathsf{CExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}\,(\Diamond \mathit{sink} \mid \neg\Diamond ↯) \\ &= \frac{\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}\,(\Diamond \mathit{sink})}{\mathrm{Pr}^{\mathcal{R}^f_\sigma[\![P]\!]}(\neg\Diamond ↯)} \\ &= \frac{\mathsf{wp}[P](f)}{\mathsf{wlp}[P](\mathbf{1})} \qquad \text{(Lemmas V.4, V.5)} \\ &= \frac{\mathsf{cwp}_1[P](f,\,\mathbf{1})}{\mathsf{cwp}_2[P](f,\,\mathbf{1})} \qquad \text{(Theorem V.1)} \\ &= \underline{\mathsf{cw}}\mathsf{p}[P](f) \end{aligned}$$

□

I. Proof of Theorem VI.1

Theorem VI.1 (Program Transformation Correctness)**.** *Let* $P \in \mathsf{cpGCL}^{\boxtimes}$ *admit at least one feasible run for every initial state and* $\mathcal{T}(P, \mathbf{1}) = (\hat{P},\, \hat{h})$*. Then for any* $f \in \mathbb{E}$ *and* $g \in \mathbb{E}_{\leq 1}$*,*

$$\mathsf{wp}[\hat{P}](f) = \underline{\mathsf{cw}}\mathsf{p}[P](f) \quad \textit{and} \quad \mathsf{wlp}[\hat{P}](g) = \underline{\mathsf{cwl}}\mathsf{p}[P](g).$$

In view of Theorem V.1, the proof reduces to showing equations $\hat{h}\cdot\mathsf{wp}[\hat{P}](f) = \mathsf{wp}[P](f)$, $\hat{h}\cdot\mathsf{wlp}[\hat{P}](f) = \mathsf{wlp}[P](f)$ and $\hat{h} = \mathsf{wlp}[P](\mathbf{1})$, which follow immediately from the auxiliary Lemma A.5 below by taking $h = \mathbf{1}$.

Lemma A.5. *Let* $P \in \mathsf{cpGCL}^{\boxtimes}$*. Then for all expectations* $f \in \mathbb{E}$ *and* $g, h \in \mathbb{E}_{\leq 1}$*, it holds*

$$\hat{h}\cdot\mathsf{wp}[\hat{P}](f) = \mathsf{wp}[P](h\cdot f) \tag{7}$$

$$\hat{h}\cdot\mathsf{wlp}[\hat{P}](g) = \mathsf{wlp}[P](h\cdot g) \tag{8}$$

$$\hat{h} = \mathsf{wlp}[P](h), \tag{9}$$

where $(\hat{P}, \hat{h}) = \mathcal{T}(P,\, h)$*.*

Proof. We prove only equations (7) and (9) since (8) follows a reasoning similar to (7). The proof proceeds by induction on the structure of P. In the remainder we will refer to the inductive hypothesis about (7) as to IH_1 and to the inductive hypothesis about (9) as to IH_2.

The Effectless Program `skip`**.** We have $\mathcal{T}(\mathtt{skip}, h) = (\mathtt{skip}, h)$ and the statement follows immediately since

$$h\cdot\mathsf{wp}[\mathtt{skip}](f) = h\cdot f = \mathsf{wp}[\mathtt{skip}](h\cdot f)$$

and

$$h = \mathsf{wlp}[\mathtt{skip}](h).$$

The Faulty Program `abort`**.** We have $\mathcal{T}(\mathtt{abort}, h) = (\mathtt{abort}, \mathbf{1})$ and the statement follows immediately since

$$\mathbf{1}\cdot\mathsf{wp}[\mathtt{abort}](f) = \mathbf{1}\cdot\mathbf{0} = \mathsf{wp}[\mathtt{abort}](h\cdot f)$$

and

$$\mathbf{1} = \mathsf{wlp}[\mathtt{abort}](h).$$

The Assignment $x \coloneqq E$**.** We have $\mathcal{T}(x \coloneqq E, h) = (x \coloneqq E, h[x/E])$ and the statement follows immediately since

$$\begin{aligned} h[x/E]\cdot\mathsf{wp}[x \coloneqq E](f) &= h[x/E]\cdot f[x/E] \\ = (h\cdot f)[x/E] &= \mathsf{wp}[x \coloneqq E](h\cdot f) \end{aligned}$$

and

$$h[x/E] = \mathsf{wlp}[x \coloneqq E](h).$$

The Observation `observe` G**.** We have $\mathcal{T}(\mathtt{observe}\, G, h) = (\mathtt{skip}, \chi_G\cdot h)$ and the statement follows immediately since

$$\begin{aligned} \chi_G\cdot h\cdot\mathsf{wp}[\mathtt{skip}](f) &= \chi_G\cdot h\cdot f \\ &= \mathsf{wp}[\mathtt{observe}\, G](h\cdot f) \end{aligned}$$

and

$$\chi_G \cdot h = \mathsf{wlp}[\mathtt{observe}\, G](h).$$

The Concatenation $P; Q$**.** Let $(\hat{Q}, \hat{h}_Q) = \mathcal{T}(Q, h)$ and $(\hat{P}, \hat{h}_P) = \mathcal{T}(P, \hat{h}_Q)$. In view of these definitions, we obtain

$$\mathcal{T}(P; Q, h) = (\hat{P}; \hat{Q}, \hat{h}_P).$$

Now

$$\begin{aligned}
&\hat{h}_P \cdot \mathsf{wp}[\hat{P}; \hat{Q}](f) \\
&\qquad = \hat{h}_P \cdot \mathsf{wp}[\hat{P}]\left(\mathsf{wp}[\hat{Q}](f)\right) \\
&\qquad = \mathsf{wp}[P](\hat{h}_Q \cdot \mathsf{wp}[\hat{Q}](f)) && (\text{IH}_1 \text{ on } P) \\
&\qquad = \mathsf{wp}[P](\mathsf{wp}[Q](h \cdot f)) && (\text{IH}_1 \text{ on } Q) \\
&\qquad = \mathsf{wp}[P; Q](h \cdot f)
\end{aligned}$$

and

$$\begin{aligned}
\hat{h}_P &= \mathsf{wlp}[P](\hat{h}_Q) && (\text{IH}_2 \text{ on } P) \\
&= \mathsf{wlp}[P](\mathsf{wlp}[Q](h)) && (\text{IH}_2 \text{ on } Q) \\
&= \mathsf{wlp}[P; Q](h).
\end{aligned}$$

The Conditional Choice $\mathtt{ite}\,(G)\,\{P\}\,\{Q\}$**.** Let $(\hat{P}, \hat{h}_P) = \mathcal{T}(P, h)$ and $(\hat{Q}, \hat{h}_Q) = \mathcal{T}(Q, h)$. On view of these definitions, we obtain

$$\begin{aligned}
&\mathcal{T}(\mathtt{ite}\,(G)\,\{P\}\,\{Q\}, h) = \\
&\quad (\mathtt{ite}\,(G)\,\{\hat{P}\}\,\{\hat{Q}\}, \chi_G \cdot \hat{h}_P + \chi_{\neg G} \cdot \hat{h}_Q).
\end{aligned}$$

Now

$$\begin{aligned}
&(\chi_G \cdot \hat{h}_P + \chi_{\neg G} \cdot \hat{h}_Q) \\
&\qquad \cdot \mathsf{wp}[\mathtt{ite}\,(G)\,\{\hat{P}\}\,\{\hat{Q}\}](f) \\
&\quad = (\chi_G \cdot \hat{h}_P + \chi_{\neg G} \cdot \hat{h}_Q) \\
&\qquad \cdot (\chi_G \cdot \mathsf{wp}[\hat{P}](f) + \chi_{\neg G} \cdot \mathsf{wp}[\hat{Q}](f)) \\
&\quad = \chi_G \cdot \hat{h}_P \cdot \mathsf{wp}[\hat{P}](f) + \chi_{\neg G} \cdot \hat{h}_Q \cdot \mathsf{wp}[\hat{Q}](f) \\
&\quad = \chi_G \cdot \mathsf{wp}[P](h \cdot f) + \chi_{\neg G} \cdot \mathsf{wp}[Q](h \cdot f) && (\text{IH}_1) \\
&\quad = \mathsf{wp}[\mathtt{ite}\,(G)\,\{P\}\,\{Q\}](h \cdot f)
\end{aligned}$$

and

$$\begin{aligned}
&\chi_G \cdot \hat{h}_P + \chi_{\neg G} \cdot \hat{h}_Q \\
&\quad = \chi_G \cdot \mathsf{wlp}[P](h) + \chi_{\neg G} \cdot \mathsf{wlp}[Q](h) && (\text{IH}_2) \\
&\quad = \mathsf{wlp}[\mathtt{ite}\,(G)\,\{P\}\,\{Q\}](h)
\end{aligned}$$

The Probabilistic Choice $\{P\}\ [p]\ \{Q\}$**.** Let $(\hat{P},\, \hat{h}_P) = \mathcal{T}(P,\, h)$ and $(\hat{Q},\, \hat{h}_Q) = \mathcal{T}(Q,\, h)$. On view of these definitions, we obtain

$$\begin{aligned}
&\mathcal{T}(\{P\}\ [\phi]\ \{Q\}, h) = \\
&\qquad (\{\hat{P}\}\ \left[{}^{\phi \cdot \hat{h}_P}/_{\hat{h}}\right]\ \{\hat{Q}\}, \phi \cdot \hat{h}_P + (\mathbf{1} - \phi) \cdot \hat{h}_Q)
\end{aligned}$$

with $\hat{h} = \phi \cdot \hat{h}_P + (\mathbf{1} - \phi) \cdot \hat{h}_Q$.

To prove the first claim

$$\hat{h} \cdot \mathsf{wp}[\{\hat{P}\}\ \left[{}^{\phi \cdot \hat{h}_P}/_{\hat{h}}\right]\ \{\hat{Q}\}](f) = \mathsf{wp}[\{P\}\ [\phi]\ \{Q\}](h \cdot f)$$

of the lemma we need to make a case distinction between those states that are mapped by $\hat{h}$ to a positive number and those that are mapped to 0. In the first case, i.e. if $\hat{h}(s) > 0$, we reason as follows:

$$\begin{aligned}
&\hat{h}(s) \cdot \mathsf{wp}[\{\hat{P}\}\ \left[{}^{\phi \cdot \hat{h}_P}/_{\hat{h}}\right]\ \{\hat{Q}\}](f)(s) \\
&\quad = \hat{h}(s) \cdot \Big(\tfrac{\phi \cdot \hat{h}_P}{\hat{h}}(s) \cdot \mathsf{wp}[\hat{P}](f)(s) \\
&\qquad\qquad + \tfrac{(\mathbf{1} - \phi) \cdot \hat{h}_Q}{\hat{h}}(s) \cdot \mathsf{wp}[\hat{Q}](f)(s)\Big) \\
&\quad = \phi(s) \cdot \hat{h}_P(s) \cdot \mathsf{wp}[\hat{P}](f)(s) \\
&\qquad + (\mathbf{1} - \phi)(s) \cdot \hat{h}_Q(s) \cdot \mathsf{wp}[\hat{Q}](f)(s) \\
&\quad = \phi(s) \cdot \mathsf{wp}[P](h \cdot f)(s) \\
&\qquad + (\mathbf{1} - \phi)(s) \cdot \mathsf{wp}[Q](h \cdot f)(s) && (\text{IH}_1) \\
&\quad = \mathsf{wp}[\{P\}\ [\phi]\ \{Q\}](h \cdot f)(s)
\end{aligned}$$

while in the second case, i.e. if $\hat{h}(s) = 0$, the claim holds because we will have $\mathsf{wp}[\{P\}\ [\phi]\ \{Q\}](h \cdot f)(s) = 0$. To see this note that if $\hat{h}(s) = 0$ then either $\phi(s) = 0 \wedge \hat{h}_Q(s) = 0$ or $\phi(s) = 1 \wedge \hat{h}_P(s) = 0$ holds. Now assume we are in the first case (an analogous argument works for the other case); using the IH_1 over Q we obtain

$$\begin{aligned}
\mathsf{wp}[\{P\}\ [0]\ \{Q\}](h \cdot f)(s) &= \mathsf{wp}[Q](h \cdot f)(s) \\
&= \hat{h}_Q(s) \cdot \mathsf{wp}[Q](f)(s) = 0.
\end{aligned}$$

The proof of the second claim of the lemma is straightforward:

$$\begin{aligned}
&\phi \cdot \hat{h}_P + (\mathbf{1} - \phi) \cdot \hat{h}_Q \\
&\quad = \phi \cdot \mathsf{wlp}[P](h) + (\mathbf{1} - \phi) \cdot \mathsf{wlp}[Q](h) && (\text{IH}_2) \\
&\quad = \mathsf{wlp}[\{P\}\ [\phi]\ \{Q\}](h).
\end{aligned}$$

The Loop $\mathtt{while}\,(G)\,\{Q\}$**.** Let $\hat{h} = \boldsymbol{\nu}\, F$ where $F(X) = \chi_G \cdot \mathcal{T}_P(X) + \chi_{\neg G} \cdot h$ and $\mathcal{T}_P(\cdot)$ is a short–hand for $\pi_2 \circ T(P, \cdot)$. Now if we let $(\hat{P}, \theta) = \mathcal{T}(P, \hat{h})$ by definition of $\mathcal{T}$ we obtain

$$\mathcal{T}(\mathtt{while}\,(G)\,\{P\}, h) = (\mathtt{while}\,(G)\,\{\hat{P}\}, \hat{h}).$$

The first claim of the lemma says that

$$\hat{h} \cdot \mathsf{wp}[\mathtt{while}\,(G)\,\{\hat{P}\}](f) = \mathsf{wp}[\mathtt{while}\,(G)\,\{P\}](h \cdot f).$$

Now if we let $H(X) = \chi_G \cdot \mathsf{wp}[\hat{P}](X) + \chi_{\neg G} \cdot f$ and $I(X) = \chi_G \cdot \mathsf{wp}[P](X) + \chi_{\neg G} \cdot h \cdot f$, the claim can be rewritten as $\hat{h} \cdot \boldsymbol{\mu}\, H = \boldsymbol{\mu}\, I$ and a straightforward argument using the Kleene fixed point theorem (and the continuity of wp established in Lemma A.1) shows that it is entailed by formula $\forall n \centerdot\ \hat{h} \cdot H^n(\mathbf{0}) = I^n(\mathbf{0})$. We prove the formula by induction on n. The case $n = 0$ is trivial. For the inductive case we reason as follows:

$$\begin{aligned}
&\hat{h} \cdot H^{n+1}(\mathbf{0}) \\
&\quad = F(\hat{h}) \cdot H^{n+1}(\mathbf{0}) && (\text{def. } \hat{h}) \\
&\quad = (\chi_G \cdot \mathcal{T}_P(\hat{h}) + \chi_{\neg G} \cdot h) \cdot H^{n+1}(\mathbf{0}) && (\text{def. } F) \\
&\quad = (\chi_G \cdot \mathcal{T}_P(\hat{h}) + \chi_{\neg G} \cdot h) \\
&\qquad\quad \cdot (\chi_G \cdot \mathsf{wp}[\hat{P}](H^n(\mathbf{0})) + \chi_{\neg G} \cdot f) && (\text{def. } H)
\end{aligned}$$

$$\begin{aligned}
&= \chi_G \cdot \mathcal{T}_P(\hat{h}) \cdot \mathsf{wp}[\hat{P}](H^n(\mathbf{0})) \\
&\quad + \chi_{\neg G} \cdot h \cdot f && \text{(algebra)} \\
&= \chi_G \cdot \theta \cdot \mathsf{wp}[\hat{P}](H^n(\mathbf{0})) + \chi_{\neg G} \cdot h \cdot f && \text{(def. } \theta\text{)} \\
&= \chi_G \cdot \mathsf{wp}[P](\hat{h} \cdot H^n(\mathbf{0})) + \chi_{\neg G} \cdot h \cdot f && \text{(IH}_1 \text{ on P)} \\
&= I(\hat{h} \cdot H^n(\mathbf{0})) && \text{(def. } I\text{)} \\
&= I^{n+1}(\mathbf{0}) && \text{(IH on } n\text{)}
\end{aligned}$$

We now turn to proving the second claim

$$\hat{h} = \mathsf{wlp}[\texttt{while}\,(G)\,\{P\}](h)$$

of the lemma. By letting $J(X) = \chi_G \cdot \mathsf{wlp}[P](X) + \chi_{\neg G} \cdot h$, the claim reduces to $\boldsymbol{\nu}\, F = \boldsymbol{\nu}\, J$, which we prove showing that $\hat{h} = \boldsymbol{\nu}\, F$ is a fixed point of J and $\boldsymbol{\nu}\, J$ is a fixed point of F. (These assertions basically imply that $\boldsymbol{\nu}\, F \geq \boldsymbol{\nu}\, J$ and $\boldsymbol{\nu}\, J \geq \boldsymbol{\nu}\, F$, respectively.)

$$\begin{aligned}
J(\hat{h}) &= \chi_G \cdot \mathsf{wlp}[P](\hat{h}) + \chi_{\neg G} \cdot h && \text{(def. } J\text{)} \\
&= \chi_G \cdot \theta + \chi_{\neg G} \cdot h && \text{(IH}_2 \text{ on } P\text{)} \\
&= \chi_G \cdot \mathcal{T}_P(\hat{h}) + \chi_{\neg G} \cdot h && \text{(def. } \theta\text{)} \\
&= F(\hat{h}) && \text{(def. } F\text{)} \\
&= \hat{h} && \text{(def. } \hat{h}\text{)}
\end{aligned}$$

$$\begin{aligned}
F(\boldsymbol{\nu}\, J) &= \chi_G \cdot \mathcal{T}_P(\boldsymbol{\nu}\, J) + \chi_{\neg G} \cdot h && \text{(def. } F\text{)} \\
&= \chi_G \cdot \mathsf{wlp}[P](\boldsymbol{\nu}\, J) + \chi_{\neg G} \cdot h && \text{(IH}_2 \text{ on } P\text{)} \\
&= J(\boldsymbol{\nu}\, J) && \text{(def. } J\text{)} \\
&= \boldsymbol{\nu}\, J && \text{(def. } \boldsymbol{\nu}\, J\text{)}
\end{aligned}$$

□

J. Proof of Theorem VI.2

Proof. Let us take the operational point of view. Let s_I be some initial state of P.

$$\begin{aligned}
&\underline{\mathsf{cw}}\mathsf{p}[P](f)(s_I) && (10) \\
&= \mathsf{CExpRew}^{\mathcal{R}^f_{s_I}[\![P]\!]}\,(\lozenge\, \mathit{sink} \mid \neg \lozenge \lightning) && (11) \\
&= \mathsf{CExpRew}^{\mathcal{R}^f_{s_I}[\![P']\!]}\,(\lozenge\, \mathit{sink} \mid \neg \lozenge\, \mathit{rerun}) && (12) \\
&= \frac{\mathsf{ExpRew}^{\mathcal{R}^f_{s_I}[\![P']\!]}\,(\lozenge\, \mathit{sink} \cap \neg \lozenge\, \mathit{rerun})}{\Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\neg \lozenge\, \mathit{rerun})} && (13) \\
&= \frac{\sum_{\hat{\pi} \in \lozenge\, \mathit{sink} \cap \neg \lozenge\, \mathit{rerun}} \Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\hat{\pi}) \cdot f(\hat{\pi})}{1 - \Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\lozenge\, \mathit{rerun})} && (14) \\
&= \sum_{i=0}^{\infty} \Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\lozenge\, \mathit{rerun})^i \\
&\qquad \cdot \sum_{\hat{\pi} \in \lozenge\, \mathit{sink} \cap \neg \lozenge\, \mathit{rerun}} \Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\hat{\pi}) \cdot f(\hat{\pi}) && (15) \\
&= \sum_{\hat{\pi} \in \lozenge\, \mathit{sink} \cap \neg \lozenge\, \mathit{rerun}} \sum_{i=0}^{\infty} \Big(\Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\lozenge\, \mathit{rerun})^i \\
&\qquad \cdot \Pr^{\mathcal{R}^f_{s_I}[\![P']\!]}(\hat{\pi}) \cdot f(\hat{\pi}) \Big) && (16) \\
&= \sum_{\hat{\pi} \in \lozenge\, \mathit{sink}} \Pr^{\mathcal{R}^f_{s_I}[\![P'']\!]}(\hat{\pi}) \cdot f(\hat{\pi}) && (17) \\
&= \mathsf{ExpRew}^{\mathcal{R}^f_{s_I}[\![P'']\!]}\,(\lozenge\, \mathit{sink}) && (18) \\
&= \mathsf{wp}(P'', f)(s_I)\ . && (19)
\end{aligned}$$

The equality (12) holds because, by construction, the probability to violate an observation in P agrees with the probability to reach a state in P' where *rerun* is true. In order to obtain equation (15) we use the fact that for a fixed real value r and probability a it holds

$$\frac{r}{1-a} = \sum_{i=0}^{\infty} a^i r\ .$$

Rewriting (15) into (16) precisely captures the expected cumulative reward of all terminating paths in P'' which is the expression in the following line. Finally we return from the operational semantics to the denotational semantics and obtain the desired result. □

K. Detailed calculations for Section VI-D

We refer to the labels init and loop introduced in the program P in Section VI-D. Further let body denote the program in the loop's body. For readability we abbreviate the variable names *delivered* as *del*, *counter* as *cntr* and *intercepted* as *int*. In the following we consider *del* and *int* as boolean variables. In order to determine (1) we first start with the numerator. This quantity is given by

$$\begin{aligned}
&\mathsf{wp}[\mathsf{init}; \mathsf{loop}; \texttt{observe}(\mathit{cntr} \leq k)]([\neg \mathit{int}]) && (20) \\
&= \mathsf{wp}[\mathsf{init}](\mathsf{wp}[\mathsf{loop}]([\mathit{cntr} \leq k \wedge \neg \mathit{int}])) && (21) \\
&= \mathsf{wp}[\mathsf{init}](\boldsymbol{\mu} F \bullet\ ([\neg \mathit{del}] \cdot \mathsf{wp}[\mathsf{body}](F) \\
&\qquad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}])) && (22) \\
&= \mathsf{wp}[\mathsf{init}](\sup_n ([\neg \mathit{del}] \cdot \mathsf{wp}[\mathsf{body}](\mathbf{0}) \\
&\qquad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}])^n) && (23)
\end{aligned}$$

where Φ^n denotes the n-fold application of Φ. Equation (21) is given directly by the semantics of sequential composition of cpGCL commands. In the next line we apply the definition of loop semantics in terms of the least fixed point. Finally, (23) is given by the Kleene fixed point theorem as a solution to the fixed point equation in (22). We can explicitly find the supremum by considering the expression for several n and deducing a pattern. Let $\Phi(F) = [\neg \mathit{del}] \cdot \mathsf{wp}[\mathsf{body}](F) + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]$. Then we have

$$\begin{aligned}
\Phi(\mathbf{0}) &= [\neg \mathit{del}] \cdot \mathsf{wp}[\mathsf{body}](\mathbf{0}) + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]
\end{aligned}$$

$$\begin{aligned}
\Phi^2(\mathbf{0}) &= \Phi([\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]) \\
&= [\neg \mathit{del}] \cdot \mathsf{wp}[\mathsf{body}]([\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]) \\
&\quad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&= [\neg \mathit{del}] \cdot (p(1-c) \cdot [\mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}]
\end{aligned}$$

$$\begin{aligned}
&\qquad\quad +(1-p)\cdot[\mathit{cntr} \leq k \wedge \neg \mathit{int}]) \\
&\qquad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&= [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\qquad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]
\end{aligned}$$

$$\begin{aligned}
\Phi^3(\mathbf{0}) &= \Phi([\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\quad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]) \\
&= \ldots \\
&= [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \cdot (1-p)p(1-c) \\
&\quad + [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}]
\end{aligned}$$

As we continue to compute $\Phi^n(\mathbf{0})$ in each step we add a summand of the form

$$[\neg \mathit{del} \wedge \mathit{cntr} + i \leq k \wedge \neg \mathit{int}] \cdot (1-p)(p(1-c))^i$$

However we see that the predicate evaluates to false for all $i > k - \mathit{cntr}$. Hence the non-zero part of the fixed point is given by

$$\begin{aligned}
&[\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + \sum_{i=0}^{k-\mathit{cntr}} [\neg \mathit{del} \wedge \mathit{cntr} + i \leq k \wedge \neg \mathit{int}] \cdot (1-p)(p(1-c))^i \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \cdot \sum_{i=0}^{k-\mathit{cntr}} (1-p)(p(1-c))^i \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\qquad \cdot (1-p)\frac{1-(p(1-c))^{k-\mathit{cntr}+1}}{1-p(1-c)} \quad .
\end{aligned}$$

where for the last equation we use a property of the finite geometric series, namely that for $r \neq 1$

$$\sum_{k=0}^{n-1} ar^k = a\,\frac{1-r^n}{1-r} \quad .$$

The result coincides with the intuition that in a state where $\mathit{del} = \mathsf{false}$, the probability to fail to reach the goal $\neg \mathit{int} \wedge \mathit{cntr} \leq k$ is distributed geometrically with probability $p(1-c)$. It is easy to verify that our educated guess is correct by checking that we indeed found a fixed point of Φ:

$$\begin{aligned}
&\Phi([\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\qquad \cdot (1-p)\frac{1-(p(1-c))^{k-\mathit{cntr}+1}}{1-p(1-c)}) \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del}] \cdot \Big((1-p) \cdot [\mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\qquad + p(1-c)\Big([\mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \\
&\qquad\quad + [\neg \mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \\
&\qquad\qquad \cdot (1-p)\frac{1-p(1-c)^{k-\mathit{cntr}}}{1-p(1-c)}\Big)\Big) \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \\
&\qquad \cdot (1-p)(p(1-c))\frac{1-p(1-c)^{k-\mathit{cntr}}}{1-p(1-c)} \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} = k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \\
&\qquad \cdot (1-p)(p(1-c))\frac{1-p(1-c)^{k-\mathit{cntr}}}{1-p(1-c)} \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} = k \wedge \neg \mathit{int}] \cdot (1-p) \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} + 1 \leq k \wedge \neg \mathit{int}] \\
&\qquad \cdot (1-p)\frac{1-p(1-c) + (p(1-c))\left(1-p(1-c)^{k-\mathit{cntr}}\right)}{1-p(1-c)} \\
&= [\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\quad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\qquad \cdot (1-p)\frac{1-p(1-c)^{k-\mathit{cntr}+1}}{1-p(1-c)}
\end{aligned}$$

Moreover this fixed point is the only fixed point and therefore the least. The justification is given by [9] where they show that loops which terminate almost surely have only one fixed point. We can now continue our calculation from (23).

$$\begin{aligned}
&= \mathsf{wp}[\mathsf{init}]([\mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \\
&\qquad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k \wedge \neg \mathit{int}] \qquad (24) \\
&\qquad\quad \cdot (1-p)\frac{1-(p(1-c))^{k-\mathit{cntr}+1}}{1-p(1-c)}) \\
&= (1-c)(1-p)\frac{1-(p(1-c))^k}{1-p(1-c)} \quad . \qquad (25)
\end{aligned}$$

This concludes the calculation of the numerator of (1). Analogously we find the denominator

$$\begin{aligned}
&\mathsf{wlp}[\mathsf{init}; \mathsf{loop}; \mathtt{observe}(\mathit{cntr} \leq k)](\mathbf{1}) \\
&= \mathsf{wlp}[\mathsf{init}](\mathsf{wlp}[\mathsf{loop}]([\mathit{cntr} \leq k])) \\
&= \mathsf{wlp}[\mathsf{init}](\boldsymbol{\nu} F\bullet\ ([\neg \mathit{del}] \cdot \mathsf{wlp}[\mathsf{body}](F) \\
&\qquad\qquad + [\mathit{del} \wedge \mathit{cntr} \leq k])) \\
&= \mathsf{wlp}[\mathsf{init}](\sup_n ([\neg \mathit{del}] \cdot \mathsf{wlp}[\mathsf{body}](\mathbf{1}) \\
&\qquad\qquad + [\mathit{del} \wedge \mathit{cntr} \leq k])^n) \\
&= \mathsf{wlp}[\mathsf{init}]([\mathit{del} \wedge \mathit{cntr} \leq k] \\
&\qquad + [\neg \mathit{del} \wedge \mathit{cntr} \leq k] \cdot (1-p^{k-counter+1})) \\
&= 1-p^k \quad . \qquad (26)
\end{aligned}$$

The only difference is that here the supremum is taken with respect to the reversed order $\geq$ in which $\mathbf{1}$ is the bottom

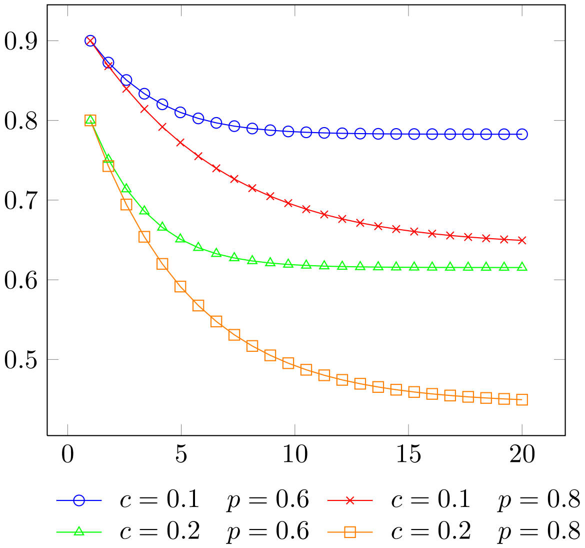


Fig. 5. The conditional probability that a message is intercepted as a function of k for fixed c and p.

and $\mathbf{0}$ is the top element. However as mentioned earlier loop terminates with probability one and the notions of wp and wlp coincide. We divide (25) by (26) to finally arrive at

$$\begin{aligned} &\underline{\mathsf{cwp}}[P]([\neg \mathit{intercepted}]) \\ &= (1-c)(1-p)\frac{1-(p(1-c))^k}{1-p(1-c)} \cdot \frac{1}{1-p^k} \quad . \end{aligned}$$

One can visualise it as a function in k by fixing the parameters c and p. For example, Figure 5 shows the conditional probability plotted for various parameter settings.