Conditioning in Probabilistic Programming

Nils Jansen, Benjamin Lucien Kaminski,
Joost-Pieter Katoen, Federico Olmedo

RWTH Aachen University, Aachen, Germany

Friedrich Gretz, Annabelle McIver

Macquarie University, Sydney, Australia

Abstract

In this paper, we investigate the semantic intricacies of conditioning in probabilistic programming, a major feature, e.g., in machine learning. We provide a quantitative weakest pre–condition semantics. In contrast to all other approaches, non–termination is taken into account by our semantics. We also present an operational semantics in terms of Markov models and show that expected rewards coincide with quantitative pre–conditions. A program transformation that entirely eliminates conditioning from programs is given; the correctness is shown using our semantics. Finally, we show that an inductive semantics for conditioning in non–deterministic probabilistic programs cannot exist.

Keywords: Probabilistic Programming, Semantics, Conditional Probabilities, Program Transformation

1 Introduction

In recent years, interest in probabilistic programming has rapidly grown [10,12]. This is due to its wide applicability, for example 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 properties [4]. Probabilistic programs, being extensions of familiar notions, render these fields accessible to programming communities. A rich palette of probabilistic programming languages exists including `Church` [9] as well as modern approaches like probabilistic `C` [24], `Tabular` [11] and `R2` [23].

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 the value of variables* in a program through so–called *observations*. The semantics of languages without conditioning is well–understood: In

⋆ This work was supported by the Excellence Initiative of the German federal and state government.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

his seminal work, Kozen [20] considered denotational semantics for probabilistic programs without non–determinism or observations. One of these semantics—the expectation transformer semantics—was adopted by McIver and Morgan [22], who added support for non–determinism; a corresponding operational semantics is given in [14]. Other relevant works include probabilistic power–domains [18], semantics of constraint probabilistic programming languages [16,15], and semantics for stochastic λ–calculi [27].

Semantic intricacies. The difficulties that arise when program variables are conditioned through observations is less well–understood. This gap is filled in this paper. Previous work on semantics for programs with `observe` statements [23,17] do neither consider the possibility of *non–termination* nor the powerful feature of *non–determinism*. In contrast, 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 analyze a few examples illustrating the different problems. We start with the problem of non–termination; consider the two program snippets

$$x \coloneqq 2 \qquad \text{and} \qquad \{x \coloneqq 2\}\ [1/2]\ \{\texttt{abort}\}\ .$$

The program on the left just assigns the value 2 to the program variable x, while the program on the right tosses a fair coin—which is modeled through a *probabilistic choice*—and depending on the outcome either performs the same variable assignment or diverges due to the `abort` instruction. The semantics given in [23,17] does not distinguish these two programs and is only sensible in the context of terminating programs. A programmer writing only terminating programs is already unrealistic in the non–probabilistic setting. Our semantics does not rely on the assumption that programs always terminate and is able to distinguish these two programs.

To discuss *observations*, consider the program snippet P_{obs_1}

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

which 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 of x being one. The `observe` statement blocks all invalid runs violating its condition and renormalizes the probabilities of the remaining valid runs. This differs, e.g., from program annotations like (probabilistic) *assertions* [26] as we will see later. The interpretation of the program is the expected outcome conditioned on the valid runs. For P_{obs_1}, this yields the outcome $1 \cdot 1$—there is one valid run that happens with probability one, with x being one.

More involved problems arise when programs are *infeasible* meaning all runs are blocked. Consider a slight variant of the program above, called P_{obs_2}:

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

The left branch of the probabilistic choice is infeasible. Is this program equivalent to the sample program P_{obs_1}? It will turn out that this is the case, meaning that setting an infeasible program into context can render it feasible.

The situation becomes more complicated when considering loopy programs that may diverge. Consider the following two programs:

$$P_{div}: \quad x \coloneqq 1;\ \texttt{while}\,(x{=}1)\,\{x \coloneqq 1\}$$
$$P_{andiv}: \quad x \coloneqq 1;\ \texttt{while}\,(x{=}1)\,\{\{x \coloneqq 1\}\ [1/2]\ \{x \coloneqq 0\};\ \texttt{observe}\ (x{=}1)\}$$

Program P_{div} diverges as x is set to one in every iteration. This yields a null *expected outcome*. Due to the conditioning on $x{=}1$, P_{andiv} has just a single (valid)—non–terminating—run, but this run *almost surely* never happens, i.e. it happens with probability zero. The conditional expected outcome of P_{andiv} can thus not be measured. Our semantics can distinguish these programs while programs with (probabilistic) assertions must be loop–free to avoid similar problems [26]. Other approaches insist on the absence of diverging loops [6]. Neither of these assumptions are realistic.

Non–determinism is a powerful means to deal with unknown information, as well as to specify abstractions in situations where implementation details are unimportant. This feature turns out to be intricate in combination with conditioning. [1] Consider the program P_{nondet}

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

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

Contributions. The above issues—non–termination, loops, 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 [22]), an elementary though foundational language that includes (amongst others) parametric probabilistic choice. We take several semantic viewpoints.

We first provide a conditional version of a *weakest pre–condition* (wp) semantics à la [22]. 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. 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. Moreover, our w(l)p semantics is backward compatible with the original pGCL semantics for programs without conditioning; this *does not* apply to alternative approaches such as R2 [23].

Furthermore, Markov Decision Processes (MDPs) [25] are used as the basis for an *operational* semantics. This semantics is simple and elegant while covering *all* aforementioned phenomena, including non–determinism. We show that *Conditional*

[1] As stated in [12], "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".

expected rewards in the MDP–semantics correspond to (conditional) wp in the denotational semantics, extending a similar result for pGCL [14].

Finally, we present a program transformation which entirely eliminates conditioning from any program and prove its correctness using our semantics.

Summarized, after introducing pGCL (Section 2), we give a denotational semantics for fully probabilistic programs (Section 3). We provide the first operational semantics for imperative probabilistic programming languages with conditioning and both probabilistic and non–deterministic choice (Section 4). Our semantics enables us to prove the correctness of a program transformation that eliminates `observe` statements (Section 5). Finally, we show that it is not possible to provide an inductive semantics for programs that include both conditioning and non–determinism (Section 6).

An extended version of this paper including more details, all proofs, and further program transformations is available at [13].

2 The Programming Language

In this section we briefly present the probabilistic programming language used for our development. The language is an extension of the *probabilistic guarded command language* (pGCL) of McIver and Morgan [22]. The original pGCL is given by syntax

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

and constitutes a plain extension of Dijkstra's guarded command language (GCL) [8] with a binary probabilistic choice operator. 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: $\{P\}\ [p]\ \{Q\}$ is a *probabilistic choice* where program P is executed with probability p and program Q with probability $1{-}p$; $\{P\} \,\square\, \{Q\}$ is a *non–deterministic choice* between P and Q; finally `abort` is syntactic sugar for the diverging program `while` (`true`) {`skip`}.

To model probabilistic programs with conditioning we extend pGCL with observations, leading to the *conditional* pGCL (cpGCL). At the syntactic level, an observation is introduced with the instruction `observe` (G), G being a Boolean expression over $\mathcal{V}$. The effect of such an instruction is to block all invalid program executions violating G and rescale the probability of the remaining executions so that they sum up to one.

As an illustrative example consider the following pair of programs:

$$\begin{aligned} P_1\colon &\quad \{x \coloneqq 0\}\ [p]\ \{x \coloneqq 1\};\ \{y \coloneqq 0\}\ [p]\ \{y \coloneqq -1\} \\ P_2\colon &\quad \{x \coloneqq 0\}\ [p]\ \{x \coloneqq 1\};\ \{y \coloneqq 0\}\ [p]\ \{y \coloneqq -1\};\ \texttt{observe}\ (x{+}y{=}0) \end{aligned}$$

Program P_1 admits all (four) runs, two of which satisfy $x{=}0$; for this program the probability of $x{=}0$ is p. Program P_2—due to the `observe` statement requiring $x{+}y{=}0$—admits only two runs, only one of them satisfying $x{=}0$; for this program

the probability of $x{=}0$ is $\frac{pq}{pq+(1-p)(1-q)}$.

Note that there exists a connection between the `observe` statement used in our work and the well–known assert statement. Both statements `observe` (G) and assert (G) block all runs violating G. The crucial difference, however, is that `observe` (G) normalizes the probability of the unblocked runs while assert (G) does not, yielding then a sub–distribution of total mass possibly less than one [21,4].

3 Denotational Semantics for Conditional pGCL

In this section we recall the expectation transformer semantics of pGCL and extend it to conditional programs in the fully probabilistic fragment of cpGCL.

3.1 Expectation Transformers in pGCL

Expectation transformers are a quantitative version of predicate transformers [8] used to endow probabilistic pGCL programs a formal semantics. Loosely speaking, they capture the average or expected outcome of a program, measured w.r.t. a utility or reward function over the set of final states. To make this more precise, 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. Intuitively, we can think of P as a mapping from an initial state $\sigma \in \mathbb{S}$ to a distribution of final states $[\![P]\!](\sigma)$; its formal semantics is captured by a transformer $\mathsf{wp}[P]$, which acts as follows: Given a random variable $f\colon \mathbb{S} \to \mathbb{R}_{\geq 0}$, $\mathsf{wp}[P](f)$ 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 predicate transformers introduced by Dijkstra [8].)

For a program P including non–deterministic choices, the execution of P yields a *set* of final distributions. To account for this, we assume that $\mathsf{wp}[P](f)(\sigma)$ gives the tightest lower bound $\inf_{\mu \in [\![P]\!](\sigma)} \mathbf{E}_\mu(f)$ for the expected value of f. This corresponds with the notion of a *demonic* adversary resolving the non–deterministic choices.

We follow McIver and Morgan [22] and 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 by induction on the structure of P, following the rules in Figure 1. The transformer wp also admits a liberal variant wlp, which differs from wp in 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 the remainder we make use of the following notation related to expectations. 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.

3.2 Conditional Expectation Transformers

We now study how to extend the notion of expectation transformers to conditioned probabilistic programs without non–determinism in cpGCL. To illustrate the intuition behind our solution, consider the following scenario: Assume we want to measure the probability that some event A occurs after the execution of a conditioned program P. Since P contains observations, its execution leads to a conditional distribution $\mu|_O$ of final states. Now the conditional probability that A occurs (given that O occurs) is given as the quotient of the probabilities $\Pr[\mu \in A \wedge O]$ and $\Pr[\mu \in O]$. Motivated by this observation, we introduce an expectation transformer $\mathsf{cwp}[\cdot] \colon \mathbb{E} \times \mathbb{E}_{\leq 1} \to \mathbb{E} \times \mathbb{E}_{\leq 1}$, whose application $\mathsf{cwp}[P](\chi_A, \mathbf{1})$ will yield the desired pair of probabilities $(\Pr[\mu \in A \wedge O], \Pr[\mu \in O])$. We are only left to define a transformer $\underline{\mathsf{cwp}}[P]$ that computes the corresponding quotient. Formally, we let

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

where $\mathsf{cwp}_1[P](f,g)$ (resp. $\mathsf{cwp}_2[P](f,g)$) denotes the first (resp. second) component of $\mathsf{cwp}[P](f,g)$. If $\mathsf{cwp}_2[P](f,\mathbf{1})(\sigma) = 0$, then $\underline{\mathsf{cwp}}[P](f)$ is not well–defined in σ (in the same way as the conditional probability $\Pr(A \mid B)$ is not well–defined [2] when $\Pr(B) = 0$) and we say that program P is *infeasible* from state σ, meaning that *all* its executions are blocked by observations.

As so defined, $\underline{\mathsf{cwp}}[P](f)$ represents the weakest *conditional* pre–expectation of P with respect to post–expectation f and $\underline{\mathsf{cwp}}[\cdot]$ generalizes the transformer $\mathsf{wp}[\cdot]$ to conditioned programs. The weakest liberal conditional pre–expectation $\underline{\mathsf{cwlp}}[P](f)$ is defined analogously, in terms of the transformer $\mathsf{cwlp}[P] \colon \mathbb{E}_{\leq 1} \times \mathbb{E}_{\leq 1} \to \mathbb{E}_{\leq 1} \times \mathbb{E}_{\leq 1}$.

We are only left to provide definitions for $\mathsf{cwp}[P]$ and $\mathsf{cwlp}[P]$. Both transformers are defined by induction on the structure of P, following the rules in Figure 1. Let us briefly explain these rules. $\mathsf{cwp}[\texttt{skip}]$ behaves as the identity since `skip` has no effect. $\mathsf{cwp}[\texttt{abort}]$ 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. $\mathsf{cwp}[x \coloneqq E]$ 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

[2] In the *continuous* setting we could define a conditional density even when conditioning on events with 0 measure using the Radon–Nikodym theorem. However, in our setting programs generate *discrete* distributions only.

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

P	$\mathsf{wlp}[P](f)$	$\mathsf{cwlp}[P](f,\, g)$
`abort`	$\mathbf{1}$	$(\mathbf{1},\, \mathbf{1})$
`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. 1. Definitions for the wp/wlp and cwp/cwlp operators. The wlp (cwlp) operator differs from wp (cwp) only for `abort` and the `while`–loop. Multiplication $h \cdot (f,\, g)$ is meant componentwise yielding $(h \cdot f,\, h \cdot g)$. Likewise, addition $(f,\, g) + (f',\, g')$ is meant componentwise yielding $(f + f',\, g + g')$.

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.[3] 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 $(\mathbf{1}, \mathbf{1})$; $\mathsf{cwlp}[\texttt{while}\,(G)\,\{P\}]$ is defined in terms of a greatest rather than a least fixed point.

Observe that Figure 1 presents no rule for the non–deterministic choice operator. Therefore our conditional expectation transformers cwp/cwlp can only handle fully probabilistic cpGCL programs. In Section 6 we elaborate on this limitation.

Example 3.1 Assume we want to compute the expected value of the expression $10{+}x$ after executing program P' given as:

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

The computation of $\mathsf{cwp}[P'](10{+}x, \mathbf{1})$ goes as follows:

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

[3] We define $\mathsf{cwp}[\texttt{while}\,(G)\,\{P\}]$ as 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 $\sqsupseteq$.

$$= \tfrac{1}{2} \cdot \tfrac{4}{5} \cdot (\mathbf{10}+\mathbf{0},\, \mathbf{1}) + \tfrac{1}{2} \cdot \tfrac{1}{2} \cdot (\mathbf{10}+\mathbf{1},\, \mathbf{1}) = \left(\tfrac{\mathbf{27}}{\mathbf{4}}, \tfrac{\mathbf{13}}{\mathbf{20}}\right) \ .$$

The expected value of $10{+}x$ is then given by $\underline{\mathsf{cwp}}[P'](10{+}x) = \frac{\mathbf{27}}{\mathbf{4}} / \frac{\mathbf{13}}{\mathbf{20}} = \frac{\mathbf{135}}{\mathbf{13}} \approx 10.38$.

In the rest of this section we investigate some properties of the expectation transformer semantics (of the fully probabilistic fragment) of cpGCL. As every fully probabilistic pGCL program is contained in cpGCL, we begin by studying the relation between the w(l)p–semantics of pGCL and the cw(l)p–semantics of cpGCL. To that end, we extend the w(l)p operator to cpGCL by the clauses $\mathsf{wp}[\texttt{observe}\ (G)](f) = \chi_G \cdot f$ and $\mathsf{wlp}[\texttt{observe}\ (G)](f) = \chi_G \cdot f$. Our first result says that cwp (resp. cwlp) can be decoupled as the product wp $\times$ wlp (resp. wlp $\times$ wlp).

Lemma 3.2 (Decoupling of cw(l)p) *Let P be a fully probabilistic* cpGCL *program, $f \in \mathbb{E}$ and $f', g \in \mathbb{E}_{\leq 1}$. Then* $\mathsf{cwp}[P](f,\, g) = \big(\mathsf{wp}[P](f),\, \mathsf{wlp}[P](g)\big)$ *and* $\mathsf{cwlp}[P](f',\, g) = \big(\mathsf{wlp}[P](f'),\, \mathsf{wlp}[P](g)\big)$.

Our next result shows that the $\underline{\mathsf{cwp}}$–semantics is a conservative extension of the wp–semantics for the fully probabilistic fragment of pGCL. The same applies to the weakest liberal pre–expectation semantics.

Theorem 3.3 (Compatibility with the w(l)p–semantics) *Let P be a fully probabilistic* pGCL *program, $f \in \mathbb{E}$, and $g \in \mathbb{E}_{\leq 1}$. Then* $\mathsf{wp}[P](f) = \underline{\mathsf{cwp}}[P](f)$ *and* $\mathsf{wlp}[P](g) = \underline{\mathsf{cwlp}}[P](g)$.

Proof. By Lemma 3.2 and the fact that $\mathsf{wlp}[P](\mathbf{1}) = \mathbf{1}$ (see Lemma 3.4). □

We now show that $\underline{\mathsf{cwp}}$ and $\underline{\mathsf{cwlp}}$ preserve the so–called healthiness conditions of wp and wlp.

Lemma 3.4 (Healthiness conditions of $\underline{\mathsf{cwp}}$ and $\underline{\mathsf{cwlp}}$) *For every fully probabilistic* cpGCL *program P with at least one feasible execution (from every initial state), every $f, g \in \mathbb{E}$ and non–negative real constants α, β:*

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

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

iii) $\underline{\mathsf{cwp}}[P](\mathbf{0}) = \mathbf{0}$ *and* $\underline{\mathsf{cwlp}}[P](\mathbf{1}) = \mathbf{1}$.

Proof. Using Lemma 3.2 one can show that the transformers $\underline{\mathsf{cwp}}$ and $\underline{\mathsf{cwlp}}$ inherit these properties from wp and wlp. For details we refer to Appendix A.4. □

We conclude this section by discussing alternative approaches for providing an expectation transformer semantics for $P \in$ cpGCL. By Lemma 3.2, the transformers $\underline{\mathsf{cwp}}[P]$ and $\underline{\mathsf{cwlp}}[P]$ can be recast as

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

respectively. An alternative is to normalize using wp instead of wlp in the denominator, yielding the two transformers

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

Transformer *ii)* 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 might lead to probabilities exceeding one. Transformer *i)* normalizes w.r.t. the terminating executions. This interpretation corresponds to the semantics of the probabilistic programming language R2 [23,17] and is *only meaningful if programs terminate almost surely* (i.e. with probability one). A noteworthy consequence of adopting transformer *i)* is that `observe` (G) is equivalent to `while` $(\neg G)$ $\{$`skip`$\}$ [17], see the discussion in Section 5.

Let us briefly compare the four alternatives by means of a concrete program P:

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

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:

$$\frac{\mathsf{wp}[P](\chi_{y=0})}{\mathsf{wlp}[P](\mathbf{1})} = \frac{2}{7} \qquad \frac{\mathsf{wlp}[P](\chi_{y=0})}{\mathsf{wlp}[P](\mathbf{1})} = \frac{6}{7} \qquad \frac{\mathsf{wp}[P](\chi_{y=0})}{\mathsf{wp}[P](\mathbf{1})} = \frac{2}{3} \qquad \frac{\mathsf{wlp}[P](\chi_{y=0})}{\mathsf{wp}[P](\mathbf{1})} = 2$$

As mentioned before, the transformer *ii)* is not significant as it yields a "probability" exceeding one. Note that our cwp–semantics yields that the probability of $y{=}0$ after the execution of P while passing all `observe`–statements is $\frac{2}{7}$. 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`.

4 Operational Semantics for Conditional pGCL

This section presents an operational semantics for cpGCL using Markov decision processes (MDPs) as underlying model. We begin by recalling the notion of MDPs. For that, let $\mathit{Distr}(S)$ denote the set of distributions $\mu\colon S \to \mathbb{R}$ over S with $\sum_{s\in S} \mu(s) = 1$.

Definition 4.1 An *MDP* is a tuple $\mathfrak{R} = (S,\, s_I,\, \mathit{Act},\, \mathcal{P},\, L)$ with a countable set of states S, an initial state $s_I \in S$, a finite set of actions Act, a transition probability function $\mathcal{P}\colon S \times \mathit{Act} \to \mathit{Distr}(S)$ with $\sum_{s'\in S} \mathcal{P}(s,\,\alpha)(s') = 1$ for all $(s,\,\alpha) \in S \times \mathit{Act}$ and a labeling function $L\colon S \to 2^{AP}$ for a set of atomic propositions AP.

A function $r\colon S \to \mathbb{R}_{\geq 0}$ is used to add *rewards* to an MDP. A *path* of $\mathfrak{R}$ is a finite or infinite sequence $\pi = s_0\alpha_0 s_1\alpha_1 \ldots$ such that $s_i \in S$, $\alpha_i \in \mathit{Act}$, $s_0 = s_I$, and $\mathcal{P}(s_i,\,\alpha_i)(s_{i+1}) > 0$ for all $i \geq 0$. The i-th state s_i of π is denoted $\pi(i)$. The set of all paths of $\mathfrak{R}$ is denoted by $\mathsf{Paths}^{\mathfrak{R}}$. $\mathsf{Paths}^{\mathfrak{R}}(s)$ 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. We sometimes omit superscript $\mathfrak{R}$ in $\mathsf{Paths}^{\mathfrak{R}}$.

An MDP operates by a non–deterministic choice of an action $\alpha \in \mathit{Act}$ that is *enabled* at state s and a subsequent probabilistic determination of a successor state according to $\mathcal{P}(s, \alpha)$. For resolving the non–deterministic choices, so–called *schedulers* are used. Let $\mathit{Sched}^{\mathfrak{R}}$ denote the class of all schedulers for $\mathfrak{R}$.

For MDP $\mathfrak{R}$, the fully probabilistic system ${}^{\mathfrak{S}}\mathfrak{R}$ induced by a scheduler $\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}}$ is called the *induced Markov Chain (MC)* on which a *probability measure*

over paths is defined. The measure for MC $\mathcal{R}$ is given by $\Pr^{\mathcal{R}}\colon \mathsf{Paths}^{\mathcal{R}} \to [0,1] \subseteq \mathbb{R}$ with $\Pr^{\mathcal{R}}(\hat{\pi}) = \prod_{i=0}^{n-1} \mathcal{P}(s_i, s_{i+1})$, for a finite path $\hat{\pi} = s_0 \dots s_n$. For sets of infinite paths the standard cylinder set construction is used, see [2, Ch. 10]. The *cumulated reward* of a finite path $\hat{\pi} = s_0 \dots s_n$ is given by $r(\hat{\pi}) = \sum_{i=0}^{n-1} r(s_i)$.

We consider *reachability properties* $\Diamond T$ for a set of target states $T \subseteq S$ where $\Diamond T$ also denotes all paths that reach T from the initial state s_I. Analogously, the set $\neg\Diamond T$ contains all paths that never reach a state in T.

First, consider reward objectives for MCs. The *expected reward* for a countable set of paths $\Diamond T$ is given by $\mathsf{ExpRew}^{\mathcal{R}}(\Diamond T) = \sum_{\hat{\pi}\in\Diamond T} \Pr^{\mathcal{R}}(\hat{\pi}) \cdot r(\hat{\pi})$. 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: $\mathsf{LExpRew}^{\mathcal{R}}(\Diamond T) = \mathsf{ExpRew}^{\mathcal{R}}(\Diamond T) + \Pr^{\mathcal{R}}(\neg\Diamond T)$. To 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]

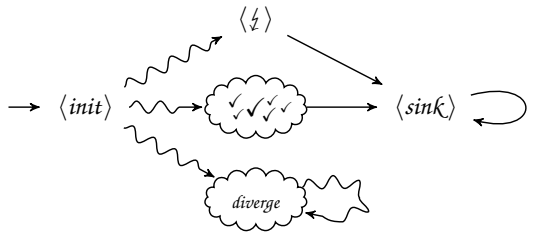
$$\mathsf{CExpRew}^{\mathcal{R}}(\Diamond T \,|\, \neg\Diamond U) \;\triangleq\; \frac{\mathsf{ExpRew}^{\mathcal{R}}(\Diamond T \cap \neg\Diamond U)}{\Pr^{\mathcal{R}}(\neg\Diamond U)} \,.$$

Reward objectives for MDPs are now defined using a *demonic* scheduler $\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}}$ minimizing probabilities and expected rewards for the induced MC ${}^{\mathfrak{S}}\mathfrak{R}$. For the expected reward this yields $\mathsf{ExpRew}^{\mathfrak{R}}(\Diamond T) = \inf_{\mathfrak{S}\in\mathit{Sched}^{\mathfrak{R}}} \mathsf{ExpRew}^{{}^{\mathfrak{S}}\mathfrak{R}}(\Diamond T)$. For conditional expected reward properties, the value of the quotient is minimized:

$$\mathsf{CExpRew}^{\mathfrak{R}}(\Diamond T \,|\, \neg\Diamond U) \;\triangleq\; \inf_{\mathfrak{S}\in\mathit{Sched}^{\mathfrak{R}}} \frac{\mathsf{ExpRew}^{{}^{\mathfrak{S}}\mathfrak{R}}(\Diamond T \cap \neg\Diamond U)}{\Pr^{{}^{\mathfrak{S}}\mathfrak{R}}(\neg\Diamond U)} \,.$$

The liberal reward notions for MDPs 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. For details about conditional probabilities and expected rewards see [3].

The structure of the operational MDP of a cpGCL program is depicted on the right. 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 ✓–labelled state, or terminates due to a violation of an 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 ✓–labelled states are the *only ones* with positive reward. Note that the sets of paths that eventually reach $\langle \lightning \rangle$, or ✓, or diverge are pairwise disjoint.



Definition 4.2 [Operational cpGCL semantics] The *operational semantics* of $P \in \mathsf{cpGCL}$ for $\sigma \in \mathbb{S}$ and $f \in \mathbb{E}$ is the MDP $\mathfrak{R}^f_\sigma[\![P]\!] = (S, \langle P, \sigma\rangle, \mathit{Act}, \mathcal{P}, L, r)$, such

[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.

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. A state of the form $\langle \downarrow, \tau \rangle$ denotes a terminal state in which no program is left to be executed. $\mathcal{P}$ is formed according to SOS rules given in Appendix A.5. For some $\tau \in \mathbb{S}$, the labelling and the reward function is given by:

$$L(s) \triangleq \begin{cases} \{\checkmark\}, & \text{if } s = \langle \downarrow, \tau \rangle \\ \{\mathit{sink}\}, & \text{if } s = \langle \mathit{sink} \rangle \\ \{\lightning\}, & \text{if } s = \langle \lightning \rangle \\ \emptyset, & \text{otherwise,} \end{cases} \qquad r(s) \triangleq \begin{cases} f(\tau), & \text{if } s = \langle \downarrow, \tau \rangle \\ 0, & \text{otherwise .} \end{cases}$$

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 $\mathsf{CExpRew}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink} \,|\, \neg\Diamond\lightning)$. Recall for the condition $\neg\Diamond\lightning$ 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:

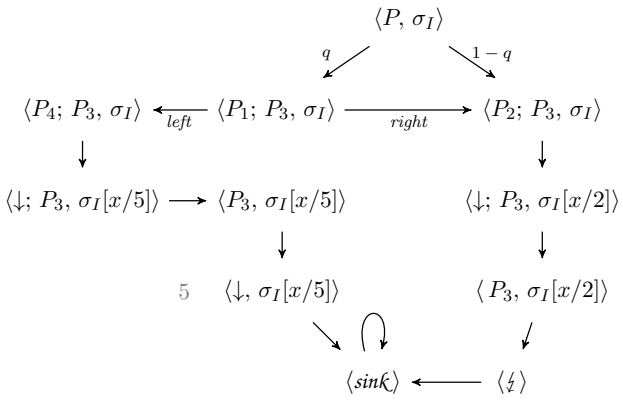
$$\begin{aligned} \mathsf{CExpRew}^{\mathfrak{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink} \,|\, \neg\Diamond\lightning) &= \inf_{\mathfrak{S} \in \mathit{Sched}^{\mathfrak{R}^f_\sigma[\![P]\!]}} \frac{\mathsf{ExpRew}^{\mathfrak{S}\mathfrak{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink} \cap \neg\Diamond\lightning)}{\Pr^{\mathfrak{S}\mathfrak{R}^f_\sigma[\![P]\!]}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

The liberal version $\mathsf{CLExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink} \,|\, \neg\Diamond\lightning)$ is defined analogously.

Example 4.3 Consider the program $P \in \mathsf{cpGCL}$:

$$\{\{x \coloneqq 5\} \,\square\, \{x \coloneqq 2\}\} \;[q]\; \{x \coloneqq 2\}; \texttt{observe}\ (x{>}3)$$

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 $P_1 = \{x \coloneqq 5\} \,\square\, \{x \coloneqq 2\}$, $P_2 = x \coloneqq 2$, $P_3 = \texttt{observe}\ (x{>}3)$, and $P_4 = x \coloneqq 5$. The operational MDP $\mathfrak{R}^x_{\sigma_I}[\![P]\!]$ for an arbitrary initial variable valuation σ_I and post–expectation x is depicted below:



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 MC 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 *demonic* 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 MC, 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 ($0/0$) and thus the *right* branch is preferred over the *left* branch. In general, the operational MDP need not be finite, even if the program terminates almost–surely (i.e. with probability 1).

We now investigate the connection to the denotational semantics of Section 3, starting with some auxiliary results. First, we establish a relation between (liberal) expected rewards and weakest (liberal) pre–expectations.

Lemma 4.4 *For any fully probabilistic $P \in \mathsf{cpGCL}$, $f \in \mathbb{E}, g \in \mathbb{E}_{\leq 1}$, and $\sigma \in \mathbb{S}$:*

$$\mathsf{ExpRew}^{\mathcal{R}_\sigma^f[\![P]\!]}\left(\Diamond \langle \mathit{sink} \rangle\right) = \mathsf{wp}[P](f)(\sigma) \tag{i}$$

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

Moreover, the probability of never reaching $\langle \lightning \rangle$ in the MC of program P coincides with the weakest liberal pre–expectation of P w.r.t. post–expectation $\mathbf{1}$:

Lemma 4.5 *For any fully probabilistic $P \in \mathsf{cpGCL}$, $g \in \mathbb{E}_{\leq 1}$, and $\sigma \in \mathbb{S}$ we have* $\Pr^{\mathcal{R}_\sigma^g[\![P]\!]}(\neg \Diamond \lightning) = \mathsf{wlp}[P](\mathbf{1})(\sigma)$.

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 fully probabilistic cpGCL programs. It turns out that the weakest (liberal) pre–expectation $\underline{\mathsf{cwlp}}[P](f)(\sigma)$ (resp. $\underline{\mathsf{cwlp}}[P](f)(\sigma)$) coincides with the conditional (liberal) expected reward in the RMC $\mathcal{R}_\sigma^f[\![P]\!]$ of terminating while never violating an `observe`–statement, i.e., avoiding the $\langle \lightning \rangle$ states.

Theorem 4.6 (Correspondence theorem) *For any fully probabilistic $P \in \mathsf{cpGCL}$, $f \in \mathbb{E}$, $g \in \mathbb{E}_{\leq 1}$ and $\sigma \in \mathbb{S}$,*

$$\begin{aligned} \mathsf{CExpRew}^{\mathcal{R}_\sigma^f[\![P]\!]}\left(\Diamond \mathit{sink} \,\middle|\, \neg \Diamond \lightning\right) &= \underline{\mathsf{cwp}}[P](f)(\sigma) \\ \mathsf{CLExpRew}^{\mathcal{R}_\sigma^g[\![P]\!]}\left(\Diamond \mathit{sink} \,\middle|\, \neg \Diamond \lightning\right) &= \underline{\mathsf{cwlp}}[P](g)(\sigma)\ . \end{aligned}$$

Proof. The proof makes use of Lemmas 4.4, 4.5, and Lemma 3.2 which themselves are proven by induction on the structure of P. For details see Appendix A.9. □

5 Program Transformation

In this section we present a program transformation for removing observations from fully probabilistic cpGCL programs and use the expectation transformer semantics

from Section 3 to prove the transformation correct. Intuitively, the presented program transformation "hoists" the `observe` statements while updating the probabilities of the probabilistic choices. Given a fully probabilistic program $P \in \mathsf{cpGCL}$, the transformation delivers a semantically equivalent `observe`–free program $\hat{P} \in \mathsf{pGCL}$ and—as a side product—an expectation $\hat{h} \in \mathbb{E}_{\leq 1}$ that captures the probability that the original program establishes all `observe` statements. For an intuition, reconsider the program from Example 3.1. The transformation yields program

$$\{x \coloneqq 0\}\ [8/13]\ \{x \coloneqq 1\}; \texttt{ite}\,(x{=}1)\,\{\{y \coloneqq 0\}\ [1]\ \{y \coloneqq 2\}\}\{\{y \coloneqq 0\}\ [1]\ \{y \coloneqq 3\}\}$$

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

$$\{x \coloneqq 0\}\ [8/13]\ \{x \coloneqq 1\};\ y \coloneqq 0$$

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 by analyzing the original program. Formally, the program transformation is given by a function

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

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 2 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 5.1 (Program Transformation Correctness) *Let P be a fully probabilistic* cpGCL *program that admits at least one valid run for every initial state and let $\mathcal{T}(P, \mathbf{1}) = (\hat{P}, \hat{h})$. Then for any $f \in \mathbb{E}$ and $g \in \mathbb{E}_{\leq 1}$, we have $\mathsf{wp}[\hat{P}](f) = \underline{\mathsf{cwp}}[P](f)$ and $\mathsf{wlp}[\hat{P}](g) = \underline{\mathsf{cwlp}}[P](g)$.*

A similar program transformation has been given by Nori *et al.* [23]. 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 over probabilistic choices and completely remove them from programs. Another difference is that the semantics of Nori *et al.* only accounts for terminating program behaviors and thus they can guarantee the correctness of the program transformation for almost–surely terminating programs only. Our semantics is more expressive and enables establishing the correctness for non–terminating program behavior, too.

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

Fig. 2. Program transformation for eliminating `observe` statements in fully probabilistic cpGCL programs.

6 Denotational Semantics for Full cpGCL

In this section we argue why (under mild assumptions) it is not possible to provide a denotational semantics in the style of conditional pre–expectation transformers (CPETs for short) for full cpGCL, i.e. including non–determinism. 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 6.1 [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 exist two functions $\mathcal{K} \colon \mathsf{cpGCL}^- \times [0,1] \times \mathsf{cpGCL}^- \to D$ and $\mathcal{N} \colon \mathsf{cpGCL}^- \times \mathsf{cpGCL}^- \to D$, such that for any $P_1, P_2 \in \mathsf{cpGCL}^-$ we have $\mathsf{cwp}^*[\{P_1\}\ [p]\ \{P_2\}] = \mathcal{K}(\mathsf{cwp}^*[P_1], p, \mathsf{cwp}^*[P_2])$ and $\mathsf{cwp}^*[\{P_1\} \,\square\, \{P_2\}] = \mathcal{N}(\mathsf{cwp}^*[P_1], \mathsf{cwp}^*[P_2])$, where $\forall d_1, d_2 \in D_\bullet\ \mathcal{N}(d_1, d_2) \in \{d_1, d_2\}$.

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 and 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 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 implementation 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 6.2 *There exists no inductive CPET.*

Proof Sketch (for details, see Appendix A.11)**.** By contradiction: Consider the program $P = \{P_1\}\ [1/2]\ \{P_5\}$ with

$$
\begin{aligned}
P_1 &: \quad x \coloneqq 1 \\
P_5 &: \quad \{P_2\} \,\square\, \{P_4\} \\
P_2 &: \quad x \coloneqq 2 \\
P_4 &: \quad \{\texttt{observe false}\}\ [1/2]\ \{P_3\} \\
P_3 &: \quad x \coloneqq 2.2
\end{aligned}
$$

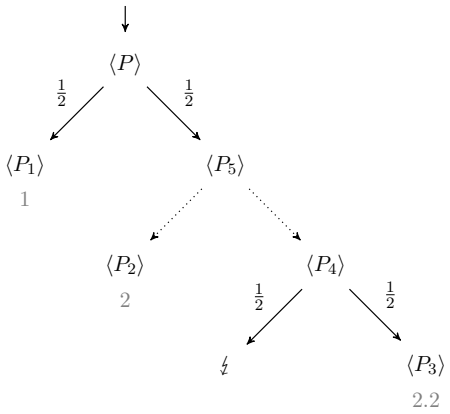


Fig. 3: Schematic depiction of the RMDP $\mathfrak{R}^{x}_{\sigma_0}[\![P]\!]$

A schematic depiction of the $\mathfrak{R}^{x}_{\sigma_0}[\![P]\!]$ is given in Figure 3. Assume there exists an inductive CPET cwp^* over some appropriate domain D. With the program given above, one can get to the contradiction $[\![\mathsf{cwp}^*[P_5]]\!] = [\![\mathsf{cwp}^*[P_4]]\!] > [\![\mathsf{cwp}^*[P_2]]\!] = [\![\mathsf{cwp}^*[P_5]]\!]$. □

As an immediate corollary of Theorem 6.2 we obtain the following result:

Corollary 6.3 *We cannot extend the* cwp *or* cwlp *rules in Figure 1 for non–deterministic programs such that Theorem 4.6 extends to full* cpGCL*.*

This result is related to Varacca and Winskel's work [28], who have already noticed the difficulties that arise when trying to integrate non–determinism and probabilities, even in the absence of conditioning. When conditioning is taken into account, Andrés and van Rossum [1] have also observed that positional schedulers—i.e. the kind of schedulers implicitly considered in the expectation transformer semantics—are not sufficient for minimizing probabilities. In contrast to our work, their development is done in the context of temporal logics.

7 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 transformation. 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 [6,19], program analysis [7] and model checking [3] 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.

Acknowledgments. We would like to thank Pedro d'Argenio and Tahiry Rabehaja for the valuable discussions preceding this paper.

References

[1] Andrés, M. E. and P. van Rossum, *Conditional probabilities over probabilistic and nondeterministic systems*, in: *Proc. of TACAS*, LNCS **4963** (2008), pp. 157–172.

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

[3] Baier, C., J. Klein, S. Klüppelholz and S. Märcker, *Computing conditional probabilities in Markovian models efficiently*, in: *Proc. of TACAS*, LNCS **8413** (2014), pp. 515–530.

[4] Barthe, G., B. Köpf, F. Olmedo and S. Z. Béguelin, *Probabilistic relational reasoning for differential privacy*, ACM Trans. Program. Lang. Syst. **35** (2013), p. 9.

[5] Bekic, H., *Definable operation in general algebras, and the theory of automata and flowcharts*, in: *Programming Languages and Their Definition* (1984), pp. 30–55.

[6] Chakarov, A. and S. Sankaranarayanan, *Expectation invariants for probabilistic program loops as fixed points*, in: *Proc. of SAS*, LNCS **8723** (2014), pp. 85–100.

[7] Cousot, P. and M. Monerau, *Probabilistic abstract interpretation*, in: H. Seidl, editor, *Proc. of ESOP*, LNCS **7211** (2012), pp. 169–193.

[8] Dijkstra, E. W., "A Discipline of Programming," Prentice Hall, 1976.

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

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

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

[12] Gordon, A. D., T. A. Henzinger, A. V. Nori and S. K. Rajamani, *Probabilistic programming*, in: *Proc. of FOSE* (2014), pp. 167–181.

[13] Gretz, F., N. Jansen, B. L. Kaminski, J.-P. Katoen, A. McIver and F. Olmedo, *Conditioning in probabilistic programming*, CoRR (2015).

[14] Gretz, F., J.-P. Katoen and A. McIver, *Operational versus weakest pre-expectation semantics for the probabilistic guarded command language*, Perform. Eval. **73** (2014), pp. 110–132.

[15] Gupta, V., R. Jagadeesan and P. Panangaden, *Stochastic processes as concurrent constraint programs*, in: A. W. Appel and A. Aiken, editors, *POPL '99, Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, January 20-22, 1999* (1999), pp. 189–202.
URL http://doi.acm.org/10.1145/292540.292558

[16] Gupta, V., R. Jagadeesan and V. A. Saraswat, *Probabilistic concurrent constraint programming*, in: *Concurrency Theory*, LNCS **1243** (1997), pp. 243–257.

[17] Hur, C.-K., A. V. Nori, S. K. Rajamani and S. Samuel, *Slicing probabilistic programs*, in: *Proc. of PLDI* (2014), pp. 133–144.

[18] Jones, C. and G. D. Plotkin, *A probabilistic powerdomain of evaluations*, in: *Logic in Computer Science* (1989), pp. 186–195.

[19] Katoen, J.-P., A. McIver, L. Meinicke and C. C. Morgan, *Linear-invariant generation for probabilistic programs*, in: *Proc. of SAS*, LNCS **6337** (2010), pp. 390–406.

[20] Kozen, D., *Semantics of probabilistic programs*, J. Comput. Syst. Sci. **22** (1981), pp. 328–350.

[21] Kozen, D., *A probabilistic {PDL}*, Journal of Computer and System Sciences **30** (1985), pp. 162 – 178.

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

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

[24] Paige, B. and F. Wood, *A compilation target for probabilistic programming languages*, in: *Proc. of ICML*, JMLR Proceedings **32** (2014), pp. 1935–1943.

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

[26] Sampson, A., P. Panchekha, T. Mytkowicz, K. S. McKinley, D. Grossman and L. Ceze, *Expressing and verifying probabilistic assertions*, in: *Proc. of PLDI* (2014), p. 14.

[27] Scott, D. S., *Stochastic λ-calculi: An extended abstract*, J. Applied Logic **12** (2014), pp. 369–376.

[28] Varacca, D. and G. Winskel, *Distributing probability over non-determinism*, Mathematical. Structures in Comp. Sci. **16** (2006), pp. 87–113.

A Appendix

Throughout the appendix we will use $\mathsf{pGCL}^{\boxtimes}$ (resp. $\mathsf{cpGCL}^{\boxtimes}$) to denote the fully probabilistic fragment of pGCL (resp. cpGCL).

A.1 Continuity of wp *and* wlp

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

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

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

$$\sup_{f\in D} \mathsf{wp}[P](f) \;=\; \mathsf{wp}[P]\left(\sup_{f\in D} f\right)\ . \tag{A.1}$$

This can be shown by structural induction on P. All cases except for the `observe` statement have been covered in [14]. It remains to show that Equality (A.1) holds for $P = \texttt{observe}\ G$:

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

The proof for the liberal transformer wlp is analogous. □

A.2 Proof of Lemma 3.2

Lemma 3.2 (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 Lemma 3.2 goes by induction over all $\mathsf{cpGCL}^{\boxtimes}$ programs. For the induction base we have:

The Effectless Program `skip`.

For cwp we have:

$$\mathsf{cwp}[\texttt{skip}](f,\, g) \;=\; (f,\, g)$$

$$= \big(\mathsf{wp}[\mathtt{skip}](f),\, \mathsf{wlp}[\mathtt{skip}](g)\big)$$

The argument for cwlp is completely analogous.

The Faulty Program `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:

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

The Assignment $x \coloneqq E$.
For cwp we have:

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

The argument for cwlp is completely analogous.

The Observation `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.

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:

The Concatenation $P; Q$.
For cwp we have:

$$\begin{aligned}&\mathsf{cwp}[P;\, Q](f,\, g)\\ =\ &\mathsf{cwp}[P](\mathsf{cwp}[Q](f,\, g)\end{aligned}$$

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

The argument for cwlp is completely analogous.

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

For cwp we have:

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

The argument for cwlp is completely analogous.

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.

The Loop $\texttt{while}\,(G)\,\{P\}$.

For cwp we have:

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

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

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

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

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

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 [5] which tells us that the least fixed point of H is given as $\big(\widehat{X_1},\ \widehat{X_2}\big)$ 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

$$\begin{aligned}
\widehat{X_2} \;&=\; \boldsymbol{\mu}_{\sqsupseteq}\, X_{2\bullet}\ H_2\big(\,\boldsymbol{\mu}_{\sqsupseteq}\, X_{1\bullet}\ H_1(X_1, X_2),\ X_2\big) \\
&=\; \boldsymbol{\mu}_{\sqsupseteq}\, X_{2\bullet}\ H_2\big(_,\ X_2\big) \\
&=\; \boldsymbol{\mu}_{\sqsupseteq}\, X_{2\bullet}\ \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g \\
&=\; \boldsymbol{\nu}_{\sqsubseteq}\, X_{2\bullet}\ \chi_G \cdot \mathsf{wlp}[P](X_2) + \chi_{\neg G} \cdot g \\
&=\; \mathsf{wlp}[\texttt{while}\,(G)\,\{P\}](g) \ ,
\end{aligned}$$

which gives us in total

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

The argument for cwlp is completely analogous. □

A.3 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,*

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

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

The Effectless Program `skip`.

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

The Faulty Program `abort`.

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

The Assignment $x \coloneqq E$.

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

The Observation `observe` G.

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

The Concatenation $P; Q$.

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

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

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

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

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

The Loop $\texttt{while}\,(G)\,\{P\}$.

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

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

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

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

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

□

A.4 Proof of Lemma 3.4

Lemma 3.4 (Elementary properties of $\underline{\mathsf{cwp}}$ and $\underline{\mathsf{cwlp}}$) *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 $\underline{\mathsf{cwp}}[P](f) \sqsubseteq \underline{\mathsf{cwp}}[P](g)$ and likewise for $\underline{\mathsf{cwlp}}$ (monotonicity).

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

iii) $\underline{\mathsf{cwp}}[P](\mathbf{0}) = \mathbf{0}$ and $\underline{\mathsf{cwlp}}[P](\mathbf{1}) = \mathbf{1}$.

Proof of i)

We do the proof for transformer $\underline{\mathsf{cwp}}$; the proof for $\underline{\mathsf{cwp}}$ is analogous. On view of Lemma 3.2, the monotonicity of $\underline{\mathsf{cwp}}$ reduces to the monotonicity of wp which follows immediately from its continuity (see Lemma A.1).

Proof of ii)

Once again, on view of Lemma 3.2, the linearity of $\underline{\mathsf{cwp}}$ follows from the linearity of wp, which we prove in Lemma A.2. [5]

[5] We cannot adopt the results from the original work [22] because their analyses is restricted to bounded expectations.

Proof of iii)

Let us begin by proving that $\underline{\mathsf{cwp}}[P](\mathbf{0}) = \mathbf{0}$. On account of Lemma 3.2 this assertion reduces to $\mathsf{wp}[P](\mathbf{0}) = \mathbf{0}$, which has already been proved for pGCL programs (see e.g. [22]). Therefore we only have to deal with the case of `observe` statements and the claim holds since $\mathsf{wp}[\texttt{observe}\,G](\mathbf{0}) = \chi_G \cdot \mathbf{0} = \mathbf{0}$. Finally formula $\underline{\mathsf{cwlp}}[P](\mathbf{1}) = \mathbf{1}$ follows immediately from Lemma 3.2. □

A.5 SOS Rules for Operational Semantics

Here we state the complete rules for constructing the operational RMDP. 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.

$$\textbf{(terminal)}\ \frac{}{\langle\downarrow,\sigma\rangle \longrightarrow \langle \mathit{sink}\rangle}$$

$$\textbf{(skip)}\ \frac{}{\langle \texttt{skip},\sigma\rangle \longrightarrow \langle\downarrow,\sigma\rangle}$$

$$\textbf{(abort)}\ \frac{}{\langle \texttt{abort},\sigma\rangle \longrightarrow \langle \texttt{abort},\sigma\rangle}$$

$$\textbf{(undesired)}\ \frac{}{\langle \lightning\rangle \longrightarrow \langle \mathit{sink}\rangle}$$

$$\textbf{(assign)}\ \frac{}{\langle x \coloneqq E,\sigma\rangle \longrightarrow \langle\downarrow, \sigma[x \leftarrow [\![E]\!]\sigma]\rangle}$$

$$\textbf{(observe)}\ \frac{\sigma \models G}{\langle \texttt{observe}\,G,\sigma\rangle \longrightarrow \langle\downarrow,\sigma\rangle} \qquad \frac{\sigma \not\models G}{\langle \texttt{observe}\,G,\sigma\rangle \longrightarrow \langle \lightning\rangle}$$

$$\textbf{(concatenate)}\ \frac{}{\langle\downarrow;Q,\sigma\rangle \longrightarrow \langle Q,\sigma\rangle} \qquad \frac{\langle P,\sigma\rangle \longrightarrow \langle \lightning\rangle}{\langle P;Q,\sigma\rangle \longrightarrow \langle \lightning\rangle}$$

$$\frac{\langle P,\sigma\rangle \longrightarrow \mu}{\langle P;Q,\sigma\rangle \longrightarrow \nu}, \text{where } \forall P'.\, \nu(\langle P';Q,\sigma'\rangle) \coloneqq \mu(\langle P',\sigma'\rangle)$$

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

$$\textbf{(while)}\ \frac{\sigma \models G}{\langle \texttt{while}\,(G)\,\{P\},\sigma\rangle \longrightarrow \langle P;\texttt{while}\,(G)\,\{P\},\sigma\rangle}$$

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

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

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

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. $x \coloneqq E$ 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.

A.6 Proof of Lemma 4.4 (i)

For proving Lemma 4.4 (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.4 [Bounded `while`–Loops] Let $P \in \mathsf{pGCL}$. Then we define:

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

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

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

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

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

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

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

As the induction hypothesis assume now that

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

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

$$\begin{aligned} & \mathsf{wp}[\texttt{while}^{<k+1}\,(G)\,\{P\}](f) \\ =\;& \mathsf{wp}[P;\ \texttt{ite}\,(G)\,\{\texttt{while}^{<k}\,(G)\,\{P\}\}\,\{\texttt{skip}\}](f) \\ =\;& (\chi_G \cdot \mathsf{wp}[P] \circ \mathsf{wp}[\texttt{while}^{<k}\,(G)\,\{P\}] \\ & \qquad + \chi_{\neg G} \cdot \mathsf{wp}[\texttt{skip}])(f) \\ =\;& \chi_G \cdot \mathsf{wp}[P](\mathsf{wp}[\texttt{while}^{<k}\,(G)\,\{P\}](f)) \\ & \qquad + \chi_{\neg G} \cdot \mathsf{wp}[\texttt{skip}](f) \end{aligned}$$

$$= \chi_G \cdot \mathsf{wp}[P](F^k(\mathbf{0})) + \chi_{\neg G} \cdot f \qquad \text{(I.H.)}$$
$$= F^{k+1}(\mathbf{0})(f) \ .$$

We have by now established that

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

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

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

□

With Lemma A.5 in mind, we can now restate and prove Lemma 4.4 (i):

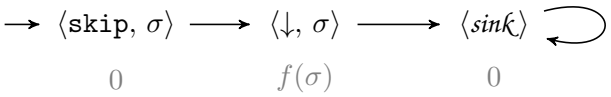
Lemma 4.4 (i) *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $f \in \mathbb{E}, g \in \mathbb{E}_{\leq 1}$, and $\sigma \in \mathbb{S}$:*

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

Proof.

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

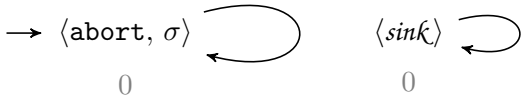
The Effectless Program `skip`. The RMC for this program is of the following form: [6]



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

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

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

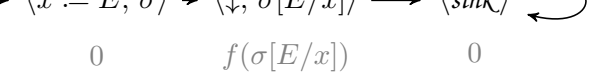


[6] If transitions have probability 1, we omit this in our figures. Moreover, all states—with the exception of $\langle \textit{sink}\rangle$—are left out if they are not reachable from the initial state.

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

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

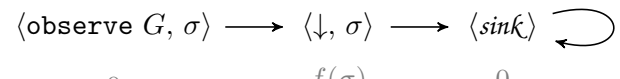
The Assignment $x \coloneqq E$. The RMC for this program is of the following form:



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 $\texttt{observe}\ G$. 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:

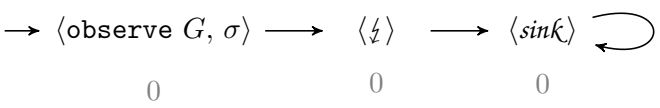


In this RMC we have $\Pi := \mathsf{Paths}(\langle \texttt{observe}\ G, \sigma\rangle, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \texttt{observe}\ G, \sigma\rangle \to \langle \downarrow, \sigma\rangle \to \langle \mathit{sink}\rangle$. Then we have for the expected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

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

$$= \mathsf{wp}[\texttt{observe } G](f)(\sigma)$$

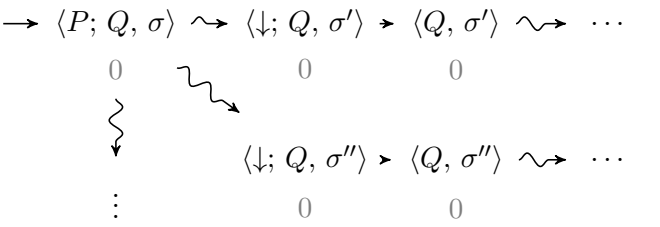
In Case 2 we have $\sigma \not\models G$, so we have $\chi_G(\sigma) = 0$. The RMC in this case is of the following form:



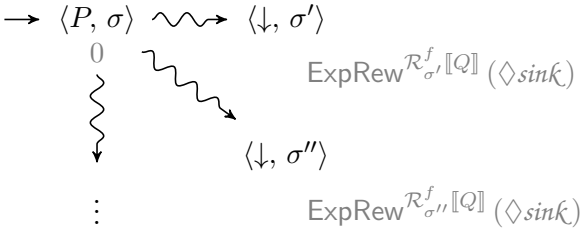
In this RMC we have $\Pi := \mathsf{Paths}(\langle \texttt{observe } G, \sigma\rangle, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \texttt{observe } G, \sigma\rangle \to \langle \lightning \rangle \to \langle \mathit{sink}\rangle$. Then for the expected reward we also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

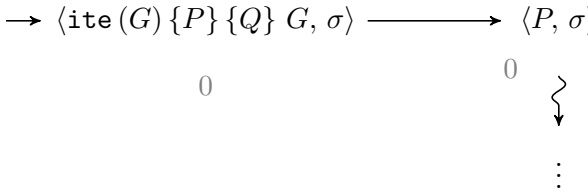


But the above RMC is exactly $\mathcal{R}^{\lambda\tau.\mathsf{ExpRew}^{\mathcal{R}^f_\tau[\![Q]\!]}(\Diamond \mathit{sink})}_\sigma[\![P]\!]$ for which the expected reward is also known by the induction hypothesis. So we have

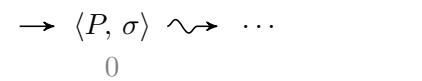
$$\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P;\,Q]\!]}(\Diamond \mathit{sink})$$

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

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:



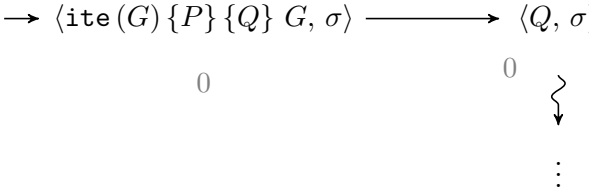
In this RMC every path in $\mathsf{Paths}(\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle, \langle sink\rangle)$ starts with $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle \to \langle P,\, \sigma\rangle \to \cdots$. As the state $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle$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



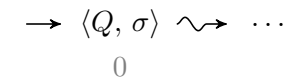
But the above RMC is exactly $\mathcal{R}_\sigma^f[\![P]\!]$ for which the expected reward is known by the induction hypothesis. So we have

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

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:



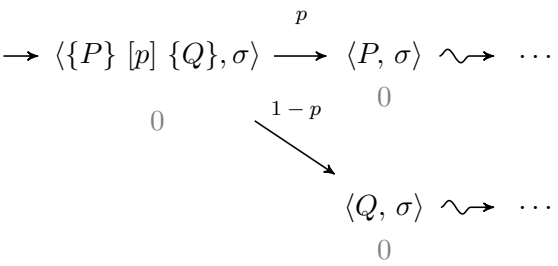
In this RMC every path in $\mathsf{Paths}(\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle, \langle sink\rangle)$ starts with $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle \to \langle Q,\, \sigma\rangle \to \cdots$. As the state $\langle \texttt{ite}\,(G)\,\{P\}\,\{Q\},\, \sigma\rangle$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



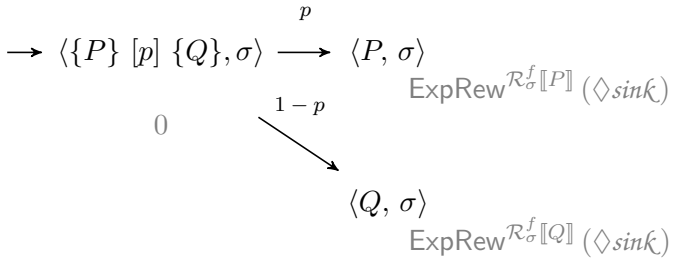
But the above RMC is exactly $\mathcal{R}^f_\sigma[\![Q]\!]$ for which the expected reward is known by the induction hypothesis. So we also have

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

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



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:



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:

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

The Loop $\texttt{while}\,(G)\,\{Q\}$. By Lemma A.5 we have

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

and as $\texttt{while}^{<k}\,(G)\,\{P\}$ 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

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

It is now left to show that

$$\sup_{k\in\mathbb{N}} \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\texttt{while}^{<k}\,(G)\,\{P\}]\!]}\,(\Diamond \mathit{sink}) \tag{A.2}$$

$$=\; \mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![\texttt{while}\,(G)\,\{P\}]\!]}\,(\Diamond \mathit{sink})\;. \tag{A.3}$$

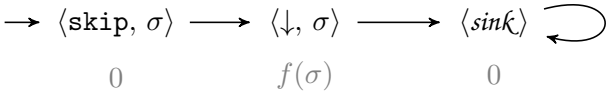
While the above is intuitively evident, it is a tedious and technically involved task to prove it. Herefore we just provide an intuition thereof: For showing (A.2) $\leq$ (A.3), we know that every path in the RMDP $\mathcal{R}^f_\sigma[\![\texttt{while}^{<k}\,(G)\,\{P\}]\!]$ either terminates properly or is prematurely aborted (yielding 0 reward) due to the fact that the bound of less than k loop iterations was reached. The RMDP $\mathcal{R}^f_\sigma[\![\texttt{while}\,(G)\,\{P\}]\!]$ 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 (A.2) $\geq$ (A.3), we know that a path that collects positive reward is necessarily finite. Therefore there exists some $k \in \mathbb{N}$ such that $\mathcal{R}^f_\sigma[\![\texttt{while}^{<k}\,(G)\,\{P\}]\!]$ includes this path. Taking the supremum over k we eventually include every path in $\mathcal{R}^f_\sigma[\![\texttt{while}\,(G)\,\{P\}]\!]$ that collects positive reward. □

A.7 Proof of Lemma 4.4 (ii)

Lemma 4.4 (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:

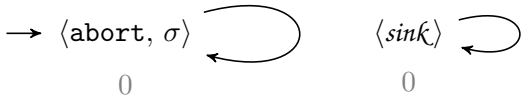


In this RMC we have $\Pi := \mathsf{Paths}(\langle\texttt{skip}, \sigma\rangle, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle\texttt{skip}, \sigma\rangle \to \langle\downarrow, \sigma\rangle \to \langle \mathit{sink}\rangle$. Then we have for the liberal expected reward:

$$\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![\texttt{skip}]\!]}\,(\Diamond \mathit{sink})$$

$$
\begin{aligned}
&= \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \textit{sink}\rangle) \\
&= \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + 0 \\
&= 1 \cdot g(\sigma) \\
&= g(\sigma) \\
&= \mathsf{wlp}[\texttt{skip}](g)(\sigma)
\end{aligned}
$$

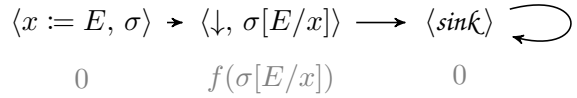
The Faulty Program `abort`. The RMC for this program is of the following form:



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

$$
\begin{aligned}
&\mathsf{ExpRew}^{\mathcal{R}^g_\sigma[\![\texttt{abort}]\!]}(\Diamond \textit{sink}) \\
&= \sum_{\hat{\pi}\in\Pi} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + \Pr(\neg\Diamond\langle \textit{sink}\rangle) \\
&= \sum_{\hat{\pi}\in\emptyset} \Pr(\hat{\pi}) \cdot r(\hat{\pi}) + 1 \\
&= 0 + 1 \\
&= 1 \\
&= \mathbf{1}(\sigma) \\
&= \mathsf{wlp}[\texttt{abort}](g)(\sigma)
\end{aligned}
$$

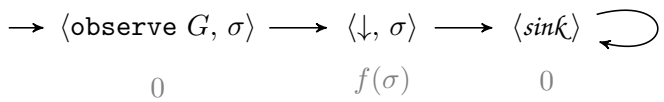
The Assignment $x \coloneqq E$. The RMC for this program is of the following form:



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

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

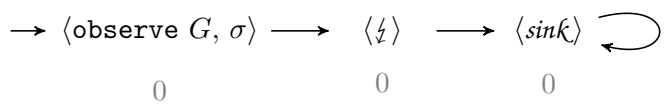
The Observation `observe` G. For this program there are two cases: In Case 1 we have $\sigma \models G$, so we have $\chi_G(\sigma) = 1$. The RMC in this case is of the following form:



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

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

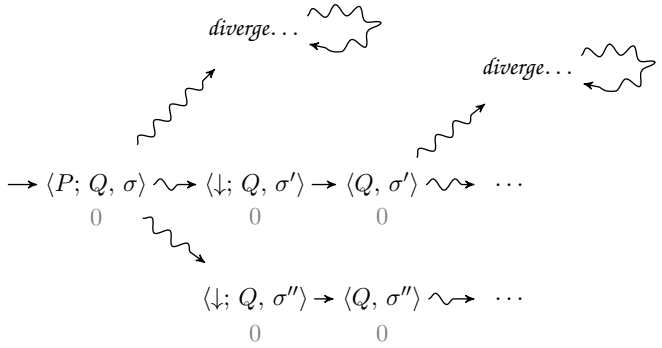
In Case 2 we have $\sigma \not\models G$, so we have $\chi_G(\sigma) = 0$. The RMC in this case is of the following form:



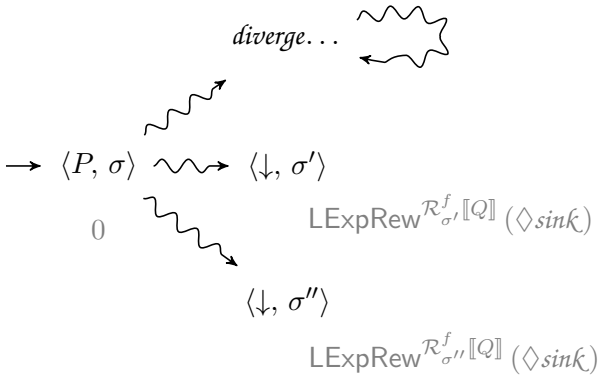
In this RMC we have $\Pi := \mathsf{Paths}(\langle \texttt{observe}\ G, \sigma\rangle, \langle \mathit{sink}\rangle) = \{\hat{\pi}_1\}$ with $\hat{\pi}_1 = \langle \texttt{observe}\ G, \sigma\rangle \to \langle \lightning\rangle \to \langle \mathit{sink}\rangle$. Then we have for the liberal expected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

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

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



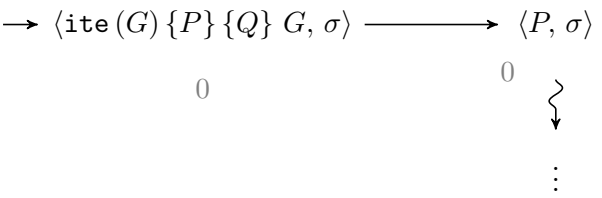
In this RMC every path in $\mathsf{Paths}(\langle P;\, Q,\, \sigma\rangle, \langle \mathit{sink}\rangle)$ starts with $\langle P;\, Q,\, \sigma\rangle$, eventually reaches $\langle \downarrow;\, Q,\, \sigma\rangle$, and then immediately after that reaches $\langle Q,\, \sigma\rangle$ which is the initial state of $\mathcal{R}^g_\sigma[\![Q]\!]$. 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:



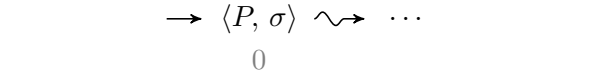
But the above RMC is exactly $\mathcal{R}^{\mathsf{LExpRew}^{\mathcal{R}^g_\sigma[\![Q]\!]}(\Diamond \mathit{sink})}_\sigma[\![P]\!]$ for which the liberal expected reward is known by the induction hypothesis. So we have for the liberal expected reward:

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

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



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

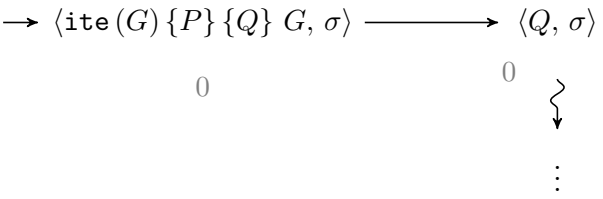


But the above RMC is exactly $\mathcal{R}^g_\sigma[\![P]\!]$ for which the expected reward is known by Lemma . A similar argument can be applied to the probability of not eventually

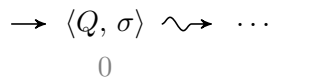
reaching $\langle \mathit{sink} \rangle$. So we have for the liberal expected reward:

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

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:



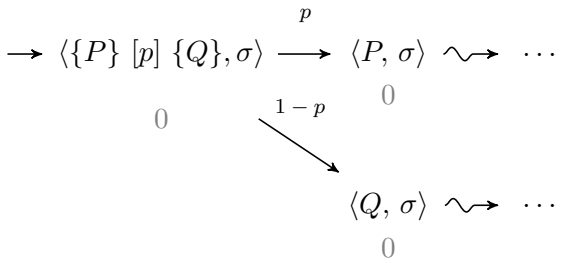
In this RMC every path in $\mathsf{Paths}(\langle \mathtt{ite}\,(G)\,\{P\}\,\{Q\}, \sigma\rangle, \langle \mathit{sink} \rangle)$ starts with $\langle \mathtt{ite}\,(G)\,\{P\}\,\{Q\}, \sigma\rangle \to \langle Q, \sigma\rangle \to \cdots$. As the state $\langle \mathtt{ite}\,(G)\,\{P\}\,\{Q\}, \sigma\rangle$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



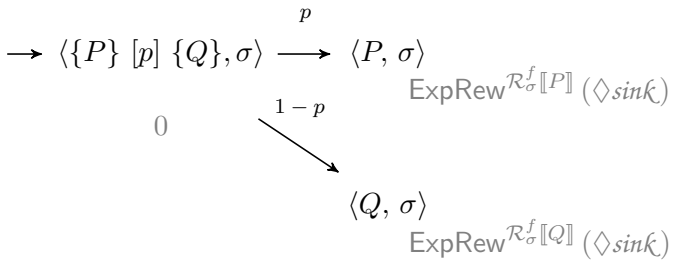
But the above RMC is exactly $\mathcal{R}^g_\sigma[\![Q]\!]$ for which the expected reward is known by the induction hypothesis. A similar argument can be applied to the probability of not eventually reaching $\langle \mathit{sink} \rangle$. So we also have for the liberal expected reward:

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

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



In this RMC every path in $\mathsf{Paths}(\langle\{P\}\ [p]\ \{Q\}, \sigma\rangle, \langle \textit{sink}\rangle)$ 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 $\langle \textit{sink}\rangle$. By this insight we can transform the above RMC into the RMC with equal liberal expected reward below:



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

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

The Loop $\texttt{while}\,(G)\,\{Q\}$.
The argument is dual to the case for the (non–liberal) expected reward. □

A.8 Proof of Lemma 4.5

Lemma 4.5 *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. First, observe that paths on reaching ✓or ↯immediately move to the state $\langle \textit{sink}\rangle$. 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 $\langle \textit{sink}\rangle$ and therefore neither reach ✓nor ↯. Furthermore the set of

"(a)–paths" and the set of "(b)–paths" are disjoint. Thus:

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

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

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

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

$$\begin{aligned} & = \ \mathsf{ExpRew}^{\mathcal{R}^{\mathbf{1}}_\sigma[\![P]\!]}(\Diamond \mathit{sink}) + \Pr^{\mathcal{R}^g_\sigma[\![P]\!]}(\neg\Diamond \mathit{sink}) \\ & = \ \mathsf{LExpRew}^{\mathcal{R}^{\mathbf{1}}_\sigma[\![P]\!]}(\Diamond \mathit{sink}) \\ & = \ \mathsf{wlp}[P](\mathbf{1}) && \text{(Lemma 4.4)} \end{aligned}$$

□

A.9 Proof of Theorem 4.6

Theorem 4.6 (Correspondence theorem) *For $P \in \mathsf{cpGCL}^{\boxtimes}$, $f \in \mathbb{E}$, $g \in \mathbb{E}_{\leq 1}$ and $\sigma \in \mathbb{S}$,*

$$\begin{aligned} \mathsf{CExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink} \,|\, \neg\Diamond \lightning) \ &= \ \underline{\mathsf{cwp}}[P](f)(\sigma) \\ \mathsf{CLExpRew}^{\mathcal{R}^g_\sigma[\![P]\!]}(\Diamond \mathit{sink} \,|\, \neg\Diamond \lightning) \ &= \ \underline{\mathsf{cwlp}}[P](g)(\sigma) \ . \end{aligned}$$

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

$$\begin{aligned} & \mathsf{CExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink} \,|\, \neg\Diamond \lightning) \\ = \ & \frac{\mathsf{ExpRew}^{\mathcal{R}^f_\sigma[\![P]\!]}(\Diamond \mathit{sink})}{\Pr^{\mathcal{R}^f_\sigma[\![P]\!]}(\neg\Diamond \lightning)} \\ = \ & \frac{\mathsf{wp}[P](f)}{\mathsf{wlp}[P](\mathbf{1})} && \text{(Lemmas 4.4, 4.5)} \\ = \ & \frac{\mathsf{cwp}_1[P](f,\,\mathbf{1})}{\mathsf{cwp}_2[P](f,\,\mathbf{1})} && \text{(Lemma 3.2)} \\ = \ & \underline{\mathsf{cwp}}[P](f) && \square \end{aligned}$$

A.10 Proof of Theorem 5.1

Theorem 5.1 (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 $g \in \mathbb{E}_{\leq 1}$,*

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

In view of Lemma 3.2, 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.6 below by taking $h = \mathbf{1}$.

Lemma A.6 *Let $P \in \mathsf{cpGCL}^{\boxtimes}$. Then for all expectations $f \in \mathbb{E}$ and $g, h \in \mathbb{E}_{\leq 1}$, it holds*

$$\hat{h} \cdot \mathsf{wp}[\hat{P}](f) = \mathsf{wp}[P](h \cdot f) \tag{A.4}$$
$$\hat{h} \cdot \mathsf{wlp}[\hat{P}](g) = \mathsf{wlp}[P](h \cdot g) \tag{A.5}$$
$$\hat{h} = \mathsf{wlp}[P](h), \tag{A.6}$$

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

Proof. We prove only equations (A.4) and (A.6) since (A.5) follows a reasoning similar to (A.4). The proof proceeds by induction on the structure of P. In the remainder we will refer to the inductive hypothesis about (A.4) as to IH_1 and to the inductive hypothesis about (A.6) as to IH_2.

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

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

and

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

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

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

and

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

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 $\mathcal{T}(\texttt{observe}\, G, h) = (\texttt{skip}, \chi_G \cdot h)$ and the statement follows immediately since

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

$$= \mathsf{wp}[\texttt{observe}\, G](h \cdot f)$$

and

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

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

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

and

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

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

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

Now

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

and

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

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

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:

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

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:

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

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

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

The first claim of the lemma says that

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

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 $\forall n_\bullet\ \hat{h} \cdot H^n(\mathbf{0}) = I^n(\mathbf{0})$. We prove the formula by induction on n. The case $n = 0$ is trivial. For the inductive case we reason as follows:

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

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.)

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

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

□

A.11 Proof of Theorem 6.2

Theorem 6.2 *There exists no inductive CPET.*

Proof. By contradiction: Consider the program $P = \{P_1\}\ [1/2]\ \{P_5\}$ with

$$P_1 \;=\; x \coloneqq 1 \qquad P_5 \;=\; \{P_2\} \,\square\, \{P_4\} \qquad P_2 \;=\; x \coloneqq 2$$
$$P_4 \;=\; \{\texttt{observe false}\}\ [1/2]\ \{P_3\} \qquad P_3 \;=\; x \coloneqq 2.2\ ,$$

A schematic depiction of the $\mathfrak{R}^x_{\sigma_0}[\![P]\!]$ is given in Figure A.1. Assume there exists an inductive CPET cwp^* over some appropriate domain D. Then,

$$\begin{aligned} \mathsf{cwp}^*[P_1] &= d_1, \text{ with } [\![d_1]\!] = 1 \\ \mathsf{cwp}^*[P_2] &= d_2, \text{ with } [\![d_2]\!] = 2 \\ \mathsf{cwp}^*[P_3] &= d_{2.2}, \text{ with } [\![d_{2.2}]\!] = 2.2 \\ \mathsf{cwp}^*[\texttt{observe false}] &= \mathfrak{of}, \text{ with } [\![\mathfrak{of}]\!] = \bot \end{aligned}$$

for some appropriate $d_1, d_2, d_{2.2}, \mathfrak{of} \in D$. By Definition 6.1, we have $\mathsf{cwp}^*[P_4] = \mathcal{K}\,(\mathsf{cwp}^*[\texttt{observe false}],\, 1/2,\, \mathsf{cwp}^*[P_3]) = \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2.2})$ and furthermore $\mathsf{cwp}^*[P_5] = \mathcal{N}\,(\mathsf{cwp}^*[P_2],\, \mathsf{cwp}^*[P_4]) = \mathcal{N}(d_2,\, \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2.2}))$. Since P_4 is a probabilistic choice between an infeasible branch and P_3, the expected value for x has to be rescaled to the feasible branch. Hence P_4 yields $[\![\mathsf{cwp}^*[P_4]]\!] = 2.2$, whereas $[\![\mathsf{cwp}^*[P_2]]\!] = 2$. Thus:

$$[\![d_2]\!] \;\lneq\; [\![\mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2.2})]\!] \tag{A.7}$$

As non–deterministic choice is demonic, we have:

$$\mathsf{cwp}^*[P_5] \;=\; \mathcal{N}(d_2,\, \mathcal{K}(\mathfrak{of},\, 1/2,\, d_{2.2})) \;=\; d_2 \tag{A.8}$$

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 $[\![\mathsf{cwp}^*\{P_1\}\ [1/2]\ \{P_2\}]\!] = 3/2$ or we rewrite P to $\{P_1\}\ [1/2]\ \{P_4\}$, which gives $[\![\mathsf{cwp}^*\{P_1\}\ [1/2]\ \{P_4\}]\!] = 7/5$. The

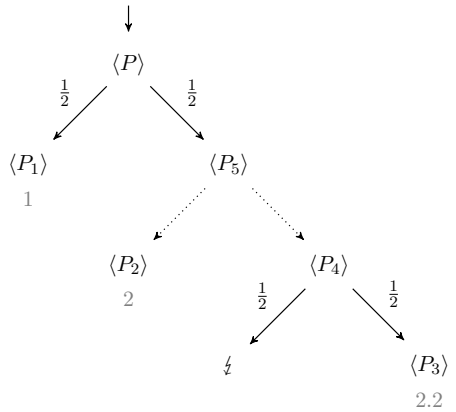


Fig. A.1. Schematic depiction of the RMDP $\mathfrak{R}^x_{\sigma_0}[\![P]\!]$

second option should be preferred by a demonic scheduler. This, however, suggests $\mathsf{cwp}^*[P_5] = \mathcal{N}(d_2, \mathcal{K}(\mathfrak{of}, 1/2, d_{2.2})) = \mathcal{K}(\mathfrak{of}, 1/2, d_{2.2})$. Together with Equality (A.8) we get $d_2 = \mathcal{K}(\mathfrak{of}, 1/2, d_{2.2})$, which implies $[\![d_2]\!] = [\![\mathcal{K}(\mathfrak{of}, 1/2, d_{2.2})]\!]$. This is a contradiction to Inequality (A.7). □