Quantitative Weakest Hyper Pre: Unifying Correctness and Incorrectness Hyperproperties via Predicate Transformers

LINPENG ZHANG, University College London, United Kingdom
NOAM ZILBERSTEIN, Cornell University, USA
BENJAMIN LUCIEN KAMINSKI, Saarland University, Germany and University College London, United Kingdom
ALEXANDRA SILVA, Cornell University, USA

We present a novel *weakest pre calculus* for *reasoning about quantitative hyperproperties* over *nondeterministic and probabilistic* programs. Whereas existing calculi allow reasoning about the expected value that a quantity assumes after program termination from a *single initial state*, we do so for *initial sets of states* or *initial probability distributions*. We thus (i) obtain a weakest pre calculus for hyper Hoare logic and (ii) enable reasoning about so-called *hyperquantities* which include expected values but also quantities (e.g. variance) out of scope of previous work. As a byproduct, we obtain a novel strongest post for weighted programs that extends both existing strongest and strongest liberal post calculi. Our framework reveals novel dualities between forward and backward transformers, correctness and incorrectness, as well as nontermination and unreachability.

CCS Concepts: • **Theory of computation** → *Logic and verification*; **Programming logic**; *Axiomatic semantics*; **Pre- and post-conditions**; *Program verification*; *Program analysis*; *Probabilistic computation*; **Hoare logic**.

Additional Key Words and Phrases: quantitative software verification, hyperproperties, strongest postcondition, weakest precondition, probabilistic verification, nondeterminism

ACM Reference Format:
Linpeng Zhang, Noam Zilberstein, Benjamin Lucien Kaminski, and Alexandra Silva. 2024. Quantitative Weakest Hyper Pre: Unifying Correctness and Incorrectness Hyperproperties via Predicate Transformers. *Proc. ACM Program. Lang.* 8, OOPSLA2, Article 300 (October 2024), 51 pages. https://doi.org/10.1145/3689740

1 Introduction

Hoare Logic (HL) [Hoare 1969] is a proof system for establishing *partial correctness* of programs—properties of *individual executions* that will always hold *if* the program terminates. However, certain properties—e.g., establishing that a system is secure via confidentiality, integrity, or authenticity—cannot be expressed in terms of *individual* executions and are therefore beyond the scope of classical Hoare Logic. This is because attackers may compare several different traces to infer hidden secrets. Clarkson and Schneider [2010] gave characterizations for this richer class of behaviors, calling them *hyperproperties*. To overcome this limitation of Hoare Logic, Benton [2004] proposed a *relational* extension of Hoare Logic for reasoning about multiple executions and verifying hyperproperties.

Authors' Contact Information: Linpeng Zhang, University College London, London, United Kingdom, linpeng.zhang.20@ucl.ac.uk; Noam Zilberstein, Cornell University, Ithaca, USA, noamz@cs.cornell.edu; Benjamin Lucien Kaminski, Saarland University, Saarbrücken, Germany and University College London, London, United Kingdom, kaminski@cs.uni-saarland.de; Alexandra Silva, Cornell University, Ithaca, USA, alexandra.silva@cornell.edu.

ACM 2475-1421/2024/10-ART300
https://doi.org/10.1145/3689740

The common element of Hoare Logic and its relational counterparts is that they apply only to properties over *all* executions (in the case of relational logics, all pairs of executions). O'Hearn [2020] refers to this class of logics as *overapproximate* and argues that it hinders their application in establishing the presence of bugs, advocating for the development of a new generation of program logics that focus on bug-finding, akin to the approaches in symbolic execution literature described by Godefroid et al. [2010]. O'Hearn [2020] proposed *Incorrectness Logic* (IL) (independently proposed by de Vries and Koutavas [2011] under the name *reverse Hoare logic*) as an analogue of Hoare Logic for developing the formal theory of bug-finding. Subsequently, other similar logics and extensions of IL were proposed [Möller et al. 2021; Raad et al. 2020]. IL can witness the reachability of particular bad outcomes but cannot make guarantees about all the possible outcomes.

The aforementioned theories of incorrectness diverge significantly from theories of correctness (such as HL), meaning that entirely separate analysis algorithms must be used for verification vs bug-finding. To overcome this limitation, new theories for *unified* reasoning about both correctness and incorrectness have been proposed [Bruni et al. 2021; Dardinier and Müller 2024; Maksimović et al. 2023; Zilberstein 2024; Zilberstein et al. 2023, 2024]. These include logics not only for individual program traces but also on hyperproperties [Dardinier and Müller 2024].

We build on two such developments—Outcome Logic (OL) [Zilberstein 2024; Zilberstein et al. 2023, 2024] and Hyper Hoare Logic (HHL) [Dardinier and Müller 2024]—which advocate that a single logic can be used to prove (or disprove) a wide variety of properties, including hyperproperties, and we present a novel *(quantitative) weakest pre calculus* perspective. Weakest precondition calculi date back to the 1970's when Dijkstra [1975, 1976] introduced them as predicate transformer semantics for imperative programs. Given a command C and a postcondition Q, the *weakest liberal precondition* is the weakest assertion P such that running C in any state satisfying P will terminate in a state satisfying Q or not terminate at all. Pratt [1976] observed that these calculi have a close connection to Hoare Logic and they were later used in a completeness proof for Hoare Logic [Clarke 1979].[1]

Weakest liberal preconditions have been generalized to probabilistic programs to allow for reasoning about expected values of random variables in a program that terminates from a *single initial state*. The core idea in these quantitative calculi [Kaminski 2019; Kozen 1985; McIver and Morgan 2005; Zhang and Kaminski 2022] is that one can replace predicates over states by real-valued functions. All these calculi, classical and quantitative, offer predicate transformers that have two key benefits over program logics: First, they discover the *most precise* assertions to make a triple valid. Second, they provide a calculus with a clear path towards mechanizability.

In this paper, we present a novel *weakest pre calculus* (whp) for *reasoning about quantitative hyperproperties* over *programs with effects* that cause the program execution to branch such as nondeterminism or probabilistic choice, in the style of weighted programming [Batz et al. 2022] or OL [Zilberstein 2024] (Section 3). We generalize existing work on quantitative weakest pre calculi [Zhang and Kaminski 2022] by considering program termination from *initial sets of states* or *initial probability distributions* rather than single initial states. We thus obtain weakest preconditions for HHL and enable reasoning about so-called *hyperquantities* (Section 4), which include expected values (considered in previous work), but also more general quantities that were not supported before, e.g. variance. Unlike HHL, our whp supports quantitative probabilistic reasoning, employing *hyperquantities* evaluated in probability distributions. Moreover, we show that many existing logics are subsumed by whp (Section 5), and how to prove (and disprove) properties in those logics. whp is hence a single calculus for correctness and incorrectness analysis, which enjoys expected

[1] Although the original relative completeness proof of Cook [1978] used the *strongest postcondition*, a later, simplified proof by Clarke [1979] used the weakest liberal precondition.

healthiness and duality properties (Section 6). whp can be applied in a variety of settings, which we illustrate through a range of examples (Section 7).

Similarly to how predicate transformers and Hoare-like logics empower programmers to demonstrate correctness, we contend that our framework offers researchers a deeper comprehension of existing logics. Our calculus reveals novel dualities between forward and backward transformers, correctness and incorrectness, as well as nontermination and unreachability.

1.1 Main challenges

While we observe parallels with existing wp calculi [Kaminski 2019; Morgan et al. 1996], HHL, and OL, extending these frameworks to our setting of (quantitative) hyperproperties involves several non-trivial steps, including lifting the calculus from initial states to hyperproperties and weighted sets of states, and completely revisiting the rules to handle our more expressive assertion language with hyperquantitites. For example, we will show that our loop rule involves a fixpoint over a higher-order function (Proposition 4.10), which is not considered in previous works. A summary of these key technical insights follows.

Weighted Strongest Postcondition. One of our key advances is to anticipate the strongest postcondition (sp) rather than use a standard operational semantics such as that of Batz et al. [2022, Section 3.3]. To achieve this, we developed a novel forward weighted sp transformer; it is interesting that within our framework (1) we subsume *both* sp and slp (arguably the main contributions of Zhang and Kaminski [2022]), and (2) the order of factors changes in some rules.

To demonstrate this, we introduce the $\odot$ operator, which represents multiplication in the context of semirings, and will be explained fully in Section 3.1. In our programming language, these semiring elements are used as weights for traces. For example, Boolean weights can be used to describe which traces are possible in a nondeterministic program, whereas real-valued weights quantify the likelihoods of probabilistic outcomes.

In commutative semirings, the order of multiplication does not matter; that is, $a \odot b = b \odot a$ for any elements a and b. However, in non-commutative semirings—which deal with sequences and order-sensitive operations—$a \odot b$ may not equal $b \odot a$. An example is the formal languages semiring in Example 4.6, where $\odot$ corresponds to word concatenation, which is clearly order dependent.

Now, we investigate the predicate transformer semantics of these weighting constructs. Below, we see that sp weights the result in the opposite order as compared to wp.

$$\mathsf{sp}\,[\![\odot\, e]\!]\,(f) = f \odot e \qquad\qquad \mathsf{wp}\,[\![\odot\, e]\!]\,(f) = e \odot f$$

Our loop rule, while similar to those in [Zhang and Kaminski 2022, Table 2], features a slightly different factor order as well. These differences, while subtle, are crucial, and the correctness of our rules is supported by the novel dualities presented in Theorem 4.5 and Example 4.6. This underscores that previous rules [Dijkstra and Scholten 1990; Zhang and Kaminski 2022] were accurate only because they used commutative semirings.

Quantitative Reasoning over Hyperproperties. Defining the meaning of quantitative reasoning in a hyperproperty setting was another challenge. We observed the similarity between hyperproperties and weighted distributions, which necessitated the development of new rules and interpretations to handle this complexity. Each of the whp rules are different compared to those of Zhang and Kaminski [2022]. In addition, the rule for nondeterministic choice is different from those of Hyper Hoare Logic (HHL) and Outcome Logic (OL), since we aim for completeness in a predicate transformer semantics, whereas HHL and OL both require additional *infinitary* rules.

Restrictions of Hyperquantities. We investigated why reasoning over quantities—as in Batz et al. [2022]; Kaminski [2019]; Kozen [1985]; McIver and Morgan [2005]; Zhang and Kaminski [2022]—is simpler, and studied the restrictions of hyperquantities to derive simpler rules similar to the existing ones (Section 6). This involved identifying and formalizing conditions under which our more general framework could simplify, bridging the complexity gap between hyperproperties and traditional properties while maintaining greater expressivity. In fact, even in restricted settings (e.g., the expected value hyperquantity), we can reason about initial probability distributions rather than single initial states.

2 Overview: Strategies for Reasoning about Hyperproperties

We begin our discussion by focusing on noninterference [Goguen and Meseguer 1982]—a hyperproperty commonly used in information security applications. More precisely, noninterference stipulates that any two executions of a program with the same *public* inputs (but potentially different *secret* inputs) must have the same public outputs. This guarantees that the program does not *leak* any secret information to unprivileged observers. As a demonstration, consider the following program, where the variable ℓ (for low) is publicly visible, but h (for high) is secret.

$$C_{\mathrm{ni}} \;=\; \texttt{assume}\ h > 0\,⨾\,\ell \coloneqq \ell + h$$

Suppose we aim to prove C_{ni} satisfies noninterference. Following the approach of logics such as Hyper Hoare Logic (HHL), one can define $\mathrm{low}(\ell)$ to mean that the value of ℓ is equal in any pair of executions, and then attempt to establish the validity of $\models_{\mathrm{hh}} \{\, \mathrm{low}(\ell)\,\}\ C_{\mathrm{ni}}\ \{\, \mathrm{low}(\ell)\,\}$, meaning that if C_{ni} is executed twice with the same initial ℓ, then ℓ will also have the same value in both executions when (and if) the program finishes—hence, the initial values of h cannot influence ℓ.

HHL is sound and complete, meaning that any true triples can be proven in it. However, doing so is not always straightforward. For example, although the specification of the triple above does not mention h, intermediary assertions required to complete the proof *must* mention h, and introducing this information cannot be done in a mechanical way, but rather requires inventiveness.

Furthermore, whereas HHL (analogously to OL) can disprove any of its triples [Dardinier and Müller 2024, Theorem 4], deriving either a positive or negative result—i.e., proving that a program is secure or not—requires one to know a priori which spec they wish to prove, or trying both.

The predicate transformer approach we advocate in this paper proves highly advantageous as it only requires a *single hyperpostcondition* to determine the most precise *hyperprecondition* that validates (or invalidates) a triple. In that sense, it solves the two aforementioned issues by mechanically working backward from the postcondition, *discovering* intermediary assertions along the way, and finding the *most precise* precondition with respect to the desired spec.

In this paper, we define a novel whp calculus, and the validity of $\mathrm{low}(\ell) \subseteq \mathrm{whp}\,[\![C_{\mathrm{ni}}]\!]\,(\mathrm{low}(\ell))$ is the answer to the noninterference problem, without the risk of attempting to prove an invalid triple. In the case of the above example, our calculus leads us to a simple counterexample; if we have $\ell = 0$ and $h = 1$ in the first execution and $\ell = 0$ and $h = 2$ in the second execution, then clearly $\mathrm{low}(\ell)$ holds, but the values of ℓ will be distinguishable at the end. This means that the program is insecure. In the remainder of this section, we will give an overview of the technical ideas underlying our whp calculus.

2.1 Classical Weakest Pre

Dijkstra's original weakest precondition calculus employs *predicate transformers* of type

$$\mathrm{wp}[\![C]\!]\colon\quad \mathbb{B} \;\longrightarrow\; \mathbb{B}\,, \qquad \text{where} \quad \mathbb{B} \;=\; \Sigma \longrightarrow \{0,1\}\;.$$

The set $\mathbb{B}$ of maps from program states (Σ) to Booleans ($\{0,1\}$) can also be thought of as predicates or assertions over program states. The *angelic* weakest precondition transformer $\mathrm{wp}[\![C]\!]$ maps a postcondition ψ to a precondition wp $[\![C]\!]$ (ψ) such that executing C on an initial state in wp $[\![C]\!]$ (ψ) guarantees that C *can*[2] terminate in a final state in ψ. Given a semantics function $[\![C]\!]$ such that $[\![C]\!](\sigma, \tau) = 1$ iff executing C on initial state σ can terminate in τ, the angelic wp is so defined:

$$\mathrm{wp}\,[\![C]\!]\,(\psi) \quad = \quad \{\sigma \in \Sigma \mid \exists \tau\boldsymbol{.}\; [\![C]\!](\sigma, \tau) = 1 \;\wedge\; \tau \in \psi\}$$

This allows to check if an angelic total correctness triple holds via the well-known fact

$$\models_{\mathrm{atc}} \{\, G \,\}\; C \;\{\, F \,\} \text{ is valid for angelic total correctness} \qquad \text{iff} \qquad G \implies \mathrm{wp}\,[\![C]\!]\,(F)\ .$$

While the above is a set perspective on wp, an equivalent perspective on wp is a map perspective: the predicate wp $[\![C]\!]$ (ψ) is a map that takes as input an initial state σ, determines for each reachable final state τ the (truth) value $\psi(\tau)$, takes a disjunction over all these truth values, and finally returns the truth value of that disjunction. More symbolically, $\mathrm{wp}\,[\![C]\!]\,(\psi)\,(\sigma) \;=\; \bigvee_{\tau\colon [\![C]\!](\sigma,\tau)=1}\; \psi(\tau)$.

2.2 Weakest Pre over Hyperproperties

To reason about hyperproperties [Clarkson and Schneider 2010], we lift our domain of discourse from *sets of states* to *sets of sets of states*, i.e. we go

$$\text{from} \qquad \mathrm{wp}[\![C]\!]\colon \quad \mathbb{B} \;\to\; \mathbb{B} \qquad \text{to} \qquad \mathrm{whp}[\![C]\!]\colon \quad \mathbb{BB} \;\to\; \mathbb{BB}\,,$$

where $\mathbb{B} \;=\; \Sigma \longrightarrow \{0, 1\}$, as before, and $\mathbb{BB} \;=\; \mathcal{P}(\Sigma) \longrightarrow \{0, 1\}$.

Given a postcondition $\psi \in \mathbb{B}$ (i.e. a predicate ranging over states), classical angelic wp $[\![C]\!]$ (ψ) anticipates for a *single* initial state σ whether running C on σ can reach ψ. Given a hyper-postcondition $\psi\!\!\!/ \;\in\; \mathbb{BB}$ (a predicate ranging over *sets* of states), the weakest hyperprecondition whp $[\![C]\!]$ $(\psi\!\!\!/)$ anticipates for a given *set* of initial states ϕ (a precondition), whether the set of states reachable from executing C on every state in ϕ satisfies $\psi\!\!\!/$. From a set perspective, we have:

$$\mathrm{whp}\,[\![C]\!]\,(\psi\!\!\!/) \quad = \quad \{\phi \in \mathcal{P}(\Sigma) \mid \mathrm{sp}\,[\![C]\!]\,(\phi) \in \psi\!\!\!/\}\,,$$

where sp $[\![C]\!]$ (ϕ) is the classical *strongest postcondition* [Dijkstra and Scholten 1990] of C with respect to precondition ϕ; in other words: the set of all final states *reachable* by executing C on any initial state in ϕ. From a map perspective, whp $[\![C]\!]$ $(\psi\!\!\!/)$ maps a hyperproperty $\psi\!\!\!/$ over postconditions to a hyperproperty whp $[\![C]\!]$ $(\psi\!\!\!/)$ over preconditions. In other words, we are anticipating whether the strongest postcondition of ϕ satisfies the hyperpostcondition $\psi\!\!\!/$:

$$\mathrm{whp}\,[\![C]\!]\,(\psi\!\!\!/)\,(\phi) \quad = \quad \psi\!\!\!/(\mathrm{sp}\,[\![C]\!]\,(\phi))\ .$$

In particular, executing C on a *precondition* ϕ satisfying whp $[\![C]\!]$ $(\psi\!\!\!/)$ guarantees that the set of reachable states sp $[\![C]\!]$ (ϕ) will satisfy $\psi\!\!\!/$. Reasoning about hyperproperties is strictly more expressive as it relates multiple executions. We showcase this in the following examples.

Example 2.1 (Weakest Hyperpreconditions). Given some precondition ϕ, if ϕ satisfies

(1) $\mathrm{whp}\,[\![C]\!]\,(\lambda\rho.\ |\rho| = 2)$, then the number of states reachable from ϕ by executing C is 2.
(2) $\mathrm{whp}\,[\![C]\!]\,(\lambda\rho.\ \mathrm{Bugs} \subseteq \rho)$, where $\mathrm{Bugs} \subseteq \Sigma$, then *all* states in the set Bugs are reachable by running C on *some* state in ϕ (this amounts to Incorrectness Logic [O'Hearn 2020]).
(3) $\mathrm{whp}\,[\![C]\!]\,(\lambda\rho.\ \rho \subseteq \mathrm{Good})$, where $\mathrm{Good} \subseteq \Sigma$, then starting from ϕ only Good can be reached or C does not terminate (this amounts to partial correctness [Hoare 1969]).

We refer to Clarkson and Schneider [2010] for more examples of hyperproperties. ◁

Remark 2.2. Outcome Logic [Zilberstein et al. 2023] and Hyper Hoare Logic [Dardinier and Müller 2024] can handle all of Example 2.1 via $\models \{\, \phi\!\!\!/ \,\}\; C \;\{\, \psi\!\!\!/ \,\}$ triples, but are agnostic of preconditions not satisfying $\phi\!\!\!/$ since $\phi \notin \phi\!\!\!/$ does not imply $\mathrm{sp}\,[\![C]\!]\,(\phi) \notin \psi\!\!\!/$. Predicate transformers, on the other hand, yield the most precise assertions in the sense that $\phi \in \mathrm{whp}\,[\![F]\!]\,(\psi\!\!\!/)$ iff $\mathrm{sp}\,[\![C]\!]\,(\phi) \in \psi\!\!\!/$. ◁

[2] C is a *nondeterministic* program. For the *demonic* setting and for deterministic programs, we can replace "can" by "will".

2.3 Quantitative Reasoning over Hyperproperties

As shown in [Kaminski 2019; Kozen 1985; McIver and Morgan 2005], one can replace predicates over states by real-valued functions, also known as quantities [Zhang and Kaminski 2022, Section 3]. These quantitative calculi subsume the classical ones by mimicking predicates through the use of *Iverson brackets* [Knuth 1992]. To design a calculus for quantitative reasoning over hyperproperties, we lift quantities in $\mathbb{A} = \{ f \mid f\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0} \}$, i.e. functions of type $\Sigma \to \mathbb{R}^{\infty}_{\geq 0}$, to hyperquantities.

Definition 2.3 (Hyperquantities). The set of all *hyper*quantities $\mathbb{AA} = \{ ff \mid ff\colon (\Sigma \to \mathbb{R}^{\infty}_{\geq 0}) \to \mathbb{R}^{\infty}_{\geq 0} \}$ is the set of all functions $ff\colon \mathbb{A} \to \mathbb{R}^{\infty}_{\geq 0}$ associating an *extended real* (i.e. either a non-negative real number or $+\infty$) to each quantity in $\mathbb{A}$. The point-wise order $ff \preceq gg \iff \forall f \in \mathbb{A}\colon ff(f) \leq gg(f)$ renders $\langle \mathbb{AA}, \preceq \rangle$ a complete lattice with join $\curlyvee$ and meet $\curlywedge$, given point-wise by

$$ff \curlyvee gg \;=\; \lambda f.\, \max\{ff(f), gg(f)\} \qquad \text{and} \qquad ff \curlywedge gg \;=\; \lambda f.\, \min\{ff(f), gg(f)\}\,.$$

Joins and meets over arbitrary subsets exist. For $a \curlyvee b \curlywedge c$, we assume that $\curlywedge$ binds stronger. ◁

Hyperquantities enable *quantitative* reasoning, e.g., measures over probability distributions.

Example 2.4 (Hyperquantities over Distributions). Given a quantity $f\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0} \in \mathbb{A}$ (think: random variable f), we define hyperquantities

$$\mathbb{E}[f] \triangleq \lambda\mu. \sum_{\sigma} f(\sigma) \cdot \mu(\sigma) \quad \mathsf{Cov}[f,g] \triangleq \lambda\mu.\, \mathbb{E}[fg](\mu) - \mathbb{E}[f](\mu) \cdot \mathbb{E}[g](\mu) \quad \mathsf{Var}[f] \triangleq \mathsf{Cov}[f,f]$$

that take as input quantities (interpreted as probability distributions) $\mu\colon \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$. The above hyperquantities are then respectively *expected value*, *variance* and *covariance* of f (and g) over μ.◁

We now present as an example an adaptation of [Dardinier and Müller 2024, Example 3] – showcasing how Boolean Hyper Hoare Logic (HHL) would deal with statistical properties.

Example 2.5 (Mean Number of Requests). Consider a program C_{db} where after termination the variable n represents the number of database requests performed. For a final set of states $\rho \subseteq \Sigma$, we define its mean number of requests by $\text{mean}_n(\rho) = \sum_{\sigma \in \rho} \frac{\sigma(n)}{|\rho|}$.

HHL allows to bound mean_n by a *specific number*, say 2, by taking as hyperpostcondition $Q = \lambda\rho.\text{mean}_n(\rho) \leq 2$. Proving the HHL triple $\models_{\text{hh}} \{\,\mathsf{true}\,\}\; C_{\text{db}}\; \{\,Q\,\}$ then ensures that for every initial set of states, the *mean number* of performed requests after the execution of C_{db} is at most 2.◁

Example 2.6 (Quantitative Information Flow). Consider a program, C_{qif} containing lowly and highly sensitive variables. As outlined in [Zhang and Kaminski 2022, Section 8.1], we will demonstrate in Section 7.3.1, how our framework also enables to determine, for instance, the maximum initial value allowable for the secret variable h based on observing a specific final value for l. HHL allows reasoning only about the existence of some information flow or about a bound over h.

Using instead quantitative weakest hyper pre has two main advantages over using HHL:

Beyond Decision Problems. While HHL and Outcome Logic (OL) are capable of statistical reasoning, our quantitative calculus can directly *measure* quantities of interest, such as the information flow.

Probability Distributions. Reasoning about means is restrictive, especially for infinite sets. As shown in Example 2.4, hyperquantities assign numerical values such as expected values to distributions. For example, whp $[\![C_{\text{db}}]\!]\,(\mathbb{E}[n])\,(\mu)$ maps *every distribution* μ to the *expected number of requests* after executing C_{db} on some initial state drawn from μ.

2.4 Limitations

Hyperproperties over probability distributions. We can only reason about properties over probability distributions *or* hyperproperties over single states (i.e., properties over sets of states) in our framework. In other words, we cannot reason about hyperproperties over probability distributions, such as probabilistic non-interference [O'Neill et al. 2006]. An attempt to do so would start by defining *probabilistic non-interference* for *observational* (i.e., input/output) programs (as opposed to Definition 3 of O'Neill et al. [2006], which focuses on interactive programs):

$$\lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(l) = \sigma_2(l) \implies \forall v.\ \mathsf{sp}\,[\![C]\!]\left(\mathbf{1}_{\sigma_1}\right)([l = v]) = \mathsf{sp}\,[\![C]\!]\left(\mathbf{1}_{\sigma_2}\right)([l = v]).$$

In other words, we require that if the program is run starting from two states with the same initial values of l, then the probability of observing each possible value of l is equal in both runs. For example, consider the program in Figure 1, which is the non-interactive analog of Program 4 from O'Neill et al. [2006]. If the probabilistic choices were replaced with non-deterministic ones, then the program would satisfy generalized non-interference, since we cannot infer the value of h by observing the value of l. However, with *probabilistic* choices, the situation changes; observing $l = 0$ means that it is more *likely* that the first path has been chosen, i.e., that h is even. We can address this situation with the above definition, and show that the program above does not satisfy probabilistic non-interference. Unfortunately, such property is a hyperproperty over probability distributions, and goes beyond our framework. Extending whp to support probabilistic non-interference is an interesting future direction.

```
if ( h is even ) {
    { l := 0 } [ 0.99 ] { l := 1 }
} else {
    { l := 0 } [ 0.01 ] { l := 1 }
}
```

Fig. 1. A program that does not satisfy probabilistic non-interference.

Demonic total correctness & angelic partial correctness. Similarly to Ascari et al. [2023]; Dardinier and Müller [2024]; Zhang and Kaminski [2022]; Zilberstein et al. [2023], we subsume neither demonic Hoare logic for total correctness, nor angelic Hoare logic for partial correctness, which are subsumed respectively by existing demonic wp and angelic wlp [Kaminski 2019]. This limitation is due to how our whp anticipates an angelic sp (as usual in literature), which only considers terminating states, and not the existence of divergent ones. We stress that this limitation holds for Hyper Hoare Logic and Outcome Logic, and that our initial objective was to establish a weakest precondition calculus for them.

3 Syntax and Semantics

We introduce a language of commands wReg, which encompasses nondeterministic imperative constructs similar to those found in the Guarded Command Language [Dijkstra 1976]. Furthermore, we adopt the weighting assertion as in [Batz et al. 2022; Zilberstein 2024], which enables representation of general weights over states. This includes reasoning of expected values over probability distributions, as studied in [Kaminski 2019; McIver and Morgan 2005].

3.1 Algebraic Preliminaries for Weights

We begin by reviewing some algebraic structures, starting with the weights of computation traces.

Definition 3.1 (Naturally Ordered Semirings). A *monoid* $\langle U, \oplus, \mathbb{0}\rangle$ consists of a set U, an associative binary operation $\oplus\colon U \times U \to U$, and an identity element $\mathbb{0} \in U$ (with $u \oplus \mathbb{0} = \mathbb{0} \oplus u = u$). The monoid is *partial* if $\oplus\colon U \times U \rightharpoonup U$ is partial, and *commutative* if $\oplus$ is commutative (i.e. $u \oplus v = v \oplus u$).

A *semiring* $\langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$ is an algebraic structure such that $\langle U, \oplus, \mathbb{0}\rangle$ is a commutative monoid, $\langle U, \odot, \mathbb{1}\rangle$ is a monoid, and the following additional properties hold:

(1) Distributivity: $u \odot (v \oplus w) = u \odot v \oplus u \odot w$ and $(u \oplus v) \odot w = u \odot w \oplus v \odot w$
(2) Annihilation: $\mathbb{0} \odot u = u \odot \mathbb{0} = \mathbb{0}$

The semiring is *partial* if $\langle U, \oplus, \mathbb{0}\rangle$ is a partial monoid (but $\odot$ is total).

On a (partial) semiring $\langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$, we define a relation $\leq$ by $u \leq v$ iff $\exists w.\, u \oplus w = v$. The semiring is called *naturally ordered* if $\leq$ is a complete partial order. ◁

As shown later in Figure 2, semirings will serve as the structure from which we draw weights of computation traces in our semantics. To this end, we extend the definition of quantities [Zhang and Kaminski 2022, Definition 3.1] to any semiring, similar to Zilberstein [2024, Definition 2.3].

Definition 3.2 (Quantities). Given a partial semiring $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$, the set $\mathbb{A}_{\mathcal{A}}(X)$ of all *quantities* is defined as the set of all functions $f\colon X \to U$, i.e. $\mathbb{A}_{\mathcal{A}}(X) = \{ f \mid f\colon X \to U \}$. ◁

We will write $\mathbb{A}$ instead of $\mathbb{A}_{\mathcal{A}}(X)$ when $\mathcal{A}$ and X are clear from context. Semiring addition, scalar multiplication, and constants are lifted pointwise to quantities as follows:

$$(m_1 \oplus m_2)(x) \triangleq m_1(x) \oplus m_2(x), \qquad (u \odot m)(x) \triangleq u \odot m(x), \qquad \text{and} \qquad u(x) \triangleq u$$

For example, by taking X as the set of program states Σ and the semiring $\langle \mathbb{R}^{\pm\infty}, \max, \min, -\infty, +\infty\rangle$ one can represent the quantities of Zhang and Kaminski [2022, Definition 3.1]. Other instances of semirings encode other computations. For example:

- Nondeterministic computation employs the Boolean semiring $\mathsf{Bool} = \langle \{0, 1\}, \vee, \wedge, 0, 1\rangle$.
- Randomization adopts probabilities in the partial semiring $\mathsf{Prob} = \langle [0, 1], +, \cdot, 0, 1\rangle$, where $x + y$ is undefined if $x + y > 1$.
- Expectations in quantitative weakest pre [Hark et al. 2019; McIver and Morgan 2005] adopts non-negative values in the semiring $\mathsf{PosReals} = \langle \mathbb{R}_{\geq 0}, +, \cdot, 0, +\infty\rangle$
- Optimization problems (e.g., the path with minimum weight) can be encoded via the tropical semiring $\mathsf{Tropical} = \langle [0, +\infty], \min, +, +\infty, 0\rangle$ which utilises non-negative real-valued weights with minimum and addition operations.

We refer to Batz et al. [2022, Table 1] and Zilberstein [2024, Section 2] for more examples and details. In the rest of the paper we will employ a more general definition of hyperquantities (Definition 2.3) that is parametrised to arbitrary semi-rings.

Definition 3.3 ((Weighted) Hyperquantities). Given a partial semiring $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$, the set $\mathbb{AA}_{\mathcal{A}}$ of all *hyperquantities* is defined as the set of all functions $ff\colon (\Sigma \to U) \to \mathbb{R}^{\infty}_{\geq 0}$, i.e. $\mathbb{AA}_{\mathcal{A}} = \mathbb{A}_{\mathsf{PosReals}}(\mathbb{A}_{\mathcal{A}}(\Sigma))$.

Similarly to quantities, when $\mathcal{A}$ is clear from the context, we will write $\mathbb{AA}$. We point out that we are purposely mapping quantities of type $\Sigma \to U$ to hyperquantities of type $(\Sigma \to U) \to \mathbb{R}^{\infty}_{\geq 0}$ and not $(\Sigma \to U) \to U'$. While we argue that one can develop a similar calculus via two possibly different semi-rings, our aim is to enable quantitative reasoning within the very well-understood real numbers, as done by [Kaminski 2019]. This allows us to generalize beyond hyperproperties, while still keeping our framework concrete enough to immediately build further tools on top of it.

3.2 Program States and Quantities

A state σ is a function that assigns a natural-numbered value to each variable. To ensure that the set of states is countable, we restrict to a finite set of program variables Vars. The set of program states is given by $\Sigma = \{ \sigma \mid \sigma\colon \mathsf{Vars} \to \mathbb{N} \}$. The semantics of an arithmetic, boolean or weight expression e is denoted by $\llbracket e \rrbracket\colon \Sigma \to \mathbb{N} \cup U$ and is obtained in a state σ, by evaluating e after replacing all occurrences of variables x by $\sigma(x)$. Moreover, we denote by $\sigma[x/v]$ a new state obtained from σ by setting the valuation of $x \in \mathsf{Vars}$ to $v \in \mathbb{N}$. Formally: $\sigma[x/v](y) = v$, if $y = x$; and $\sigma(y)$, otherwise.

A particular useful quantity is the Iverson bracket [Knuth 1992]: denoted as $[\varphi]$ for a given predicate φ, it takes as input a state σ and evaluates to 1 if the statement is true and 0 if the statement is false. We generalise it to arbitrary semirings, subsuming other quantitative generalisations such as [Zhang and Kaminski 2022, Definition 3.5].

$$[\![x := e]\!](\sigma,\tau) \triangleq [\sigma[x/\sigma(e)] = \tau] \qquad \text{(assignment)}$$

$$[\![x := \mathtt{nondet}()]\!](\sigma,\tau) \triangleq \bigoplus_{\alpha\in\mathbb{N}} [\sigma[x/\alpha] = \tau] \qquad \text{(nondeterministic assignment)}$$

$$[\![\odot\, e]\!](\sigma,\tau) \triangleq [\![e]\!](\sigma) \odot [\sigma = \tau] \qquad \text{(weighting)}$$

$$[\![C_1 \mathbin{⨾} C_2]\!](\sigma,\tau) \triangleq \bigoplus_{\iota\in\Sigma} [\![C_1]\!](\sigma,\iota) \odot [\![C_2]\!](\iota,\tau) \qquad \text{(sequential composition)}$$

$$[\![\{\, C_1 \,\} \mathbin{\square} \{\, C_2 \,\}]\!](\sigma,\tau) \triangleq [\![C_1]\!](\sigma,\tau) \oplus [\![C_2]\!](\sigma,\tau) \qquad \text{(nondeterministic choice)}$$

$$[\![C^{\langle e,e'\rangle}]\!](\sigma,\tau) \triangleq (\mathsf{lfp}\, X\text{.}\quad \Phi_{C,e,e'}(X))(\sigma,\tau) \qquad \text{(iteration)}$$

$$\text{where}\quad \Phi_{C,e,e'}(X)(\sigma,\tau) = [\![e]\!](\sigma) \odot \left(\bigoplus_{\iota\in\Sigma} [\![C]\!](\sigma,\iota) \odot X(\iota,\tau)\right) \oplus [\![e']\!](\sigma) \odot [\sigma = \tau] \quad \triangleleft$$

Fig. 2. Denotational semantics $[\![C]\!]\colon (\Sigma\times\Sigma) \longrightarrow U$ of wReg programs, where $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$ is a semiring and the least fixed point is defined via point-wise extension of the natural order $\leq$ such that $f \leq f'$ iff $f(\sigma_1,\sigma_2) \leq f'(\sigma_1,\sigma_2)$ for all $\sigma,\sigma' \in \Sigma$.

Definition 3.4 (Iverson Brackets). For any semiring $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$ and a predicate φ over program states Σ, the Iverson bracket $[\varphi] : \Sigma \longrightarrow U$ is defined as

$$[\varphi]\,(\sigma) \triangleq \begin{cases}\mathbb{1}, \quad \text{if } \sigma \models \varphi; \qquad \text{and} \qquad \mathbb{0}, \quad \text{otherwise}\,.\end{cases} \qquad \triangleleft$$

3.3 Weighted Programs

Throughout the paper, we denote $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$ as a naturally ordered, complete, Scott continuous, partial semiring with a top element $\top \in U$ such that $\top \geq u$ for all $u \in U$. We assign meaning to wReg-statements in terms of a denotational semantics, taking as input an *initial state* σ and a *final state* τ, and returning the sum of the weights of all paths starting from σ and terminating in τ after the execution of C. The syntax of the *weighted regular command language* (wReg) is below:

$$\begin{array}{llllllll} C & ::= & x := e & \text{(assignment)} & \mid\; x := \mathtt{nondet}() & \text{(nondet. assign.)} & \mid\; \odot\, e & \text{(weighting)} \\ & \mid & C \mathbin{⨾} C & \text{(sequencing)} & \mid\; \{\, C \,\} \mathbin{\square} \{\, C \,\} & \text{(nondet. choice)} & \mid\; C^{\langle e,e'\rangle} & \text{(iteration)} \end{array}$$

where $\odot\, e$ weights the current computation branch. Similarly to [Batz et al. 2022; Zhang and Kaminski 2022], we do not provide an explicit syntax for weights because we focus on semantic assertions. Our weighting construct is more expressive than Batz et al. [2022]; Zilberstein [2024]: not only we can represent values $u \in U$ and Boolean tests (via Iverson brackets), but we also reason about intensional properties of the computation. The iteration $C^{\langle e,e'\rangle}$, introduced in [Zilberstein 2024], terminates with weight e' or executes the body C with weight e. This construct simplifies the representation of while loops with $\mathtt{while}\,(\,\varphi\,)\,\{\,C\,\}$, probabilistic iterations using $C^{\langle p,1-p\rangle}$, and Kleene's star as $C^{\langle \mathbb{1},\mathbb{1}\rangle}$. Its usefulness is evident, especially in partial semirings where loops via Kleene star may not be well-defined due to its nondeterministic nature [Zilberstein 2024, Footnote 2]. Many common constructs, such as tests, branchings and loops are syntactic sugar, for instance:

$$\mathtt{assume}\ \varphi \triangleq \odot\,\varphi \qquad \mathtt{diverge} \triangleq \odot\,\mathbb{0}$$

$$\mathtt{if}\ (\,\varphi\,)\ \{\,C_1\,\}\ \mathtt{else}\ \{\,C_2\,\} \triangleq \{\,\mathtt{assume}\ \varphi \mathbin{⨾} C_1\,\} \mathbin{\square} \{\,\mathtt{assume}\ \neg\varphi \mathbin{⨾} C_2\,\}$$

$$\{\,C_1\,\}\ [\,p\,]\ \{\,C_2\,\} \triangleq \{\,\odot\, p \mathbin{⨾} C_1\,\} \mathbin{\square} \{\,\odot\, 1-p \mathbin{⨾} C_1\,\}$$

$$\mathtt{while}\,(\,\varphi\,)\,\{\,C\,\} \triangleq C^{\langle \varphi,\neg\varphi\rangle} \qquad C^{\star} \triangleq C^{\langle \mathbb{1},\mathbb{1}\rangle}$$

The semantics is shown in Figure 2 and is described below.

Assignment: The semantics for assignment asserts that the weight of transitioning from σ to τ after executing $x := e$ is $\mathbb{1}$ if τ is equal to σ with the value of x updated to $\sigma(e)$, or $\mathbb{0}$ otherwise.

Nondeterministic Assignment: The denotational semantics for $x \coloneqq \texttt{nondet()}$, indicates that the weight of transitioning from initial state σ to final state τ after executing $x \coloneqq \texttt{nondet()}$ is $\mathbb{1}$ if σ and τ differ only in the value of x, and $\mathbb{0}$ otherwise. This is achieved by treating $\bigoplus$ akin to an existential quantifier. Specifically, given σ, we consider all possible values that x may take after the execution of $x \coloneqq \texttt{nondet()}$.

Assume/Weighting: The semantics for $\texttt{assume}\ \varphi$ indicates that the weight of transitioning from σ to σ is determined by the evaluation of φ in σ. If $\tau \neq \sigma$, then the weight of the transition is $\mathbb{0}$.

The intuition of the weighting statement in Batz et al. [2022] is to weight arbitrary constant values $u \in U$, which does not generalize $\texttt{assume}\ \varphi$ (but only $\texttt{assume true}$ and $\texttt{assume false}$). In our setting, weight can be any expression, so $\odot\, e$ is a proper generalization of the assume rule and is defined as $[\![\odot\, e]\!](\sigma, \tau) \; = \; [\![e]\!](\sigma) \odot [\sigma = \tau]$. Here, the weighting rule expresses that the weight of transitioning from σ to itself after a weighting operation is determined by the weight $[\![e]\!](\sigma)$.

Sequential Composition: The semantics for $C_1 \fatsemi C_2$ calculates the weight of transitioning from σ to τ after executing a sequence of C_1 followed by C_2, considering all possible intermediate states σ'.

Nondeterministic Choice: The semantics for $\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}$ captures the weight of transitioning from σ to τ after executing either C_1 or C_2, with the weight being the sum of the individual weights.

Iteration: The intended meaning of $C^{\langle e,e' \rangle}$ is to be equal to $\left\{ \odot\, e \fatsemi C \fatsemi C^{\langle e,e' \rangle} \right\} \square \{\, \odot\, e' \,\}$. Replacing the recursive instance of $C^{\langle e,e' \rangle}$ with X, we get $\Phi_{C,e,e'}(X)$, and so by Kleene's fixpoint theorem, the least fixed point corresponds to iterating on the least element of the complete partial order $\mathbb{0}$, which yields an ascending chain of unrollings. This process can be demonstrated through the following sequence:

$$\begin{aligned}
\Phi_{C,e,e'}(\mathbb{0})(\sigma,\tau) \;&=\; [\![\{\, \odot\, e \fatsemi \texttt{diverge} \,\} \,\square\, \{\, \odot\, e' \,\}]\!](\sigma,\tau) \\
\Phi^2_{C,e,e'}(\mathbb{0})(\sigma,\tau) \;&=\; [\![\{\, \odot\, e \fatsemi C \fatsemi \{\, \odot\, e \fatsemi \texttt{diverge} \,\} \,\square\, \{\, \odot\, e' \,\} \,\} \,\square\, \{\, \odot\, e' \,\}]\!](\sigma,\tau)
\end{aligned}$$

and so on, whose supremum is the least fixed point of $\Phi_{C,e,e'}$.

Well-definedness of the Denotational Semantics

The semantics of iteration loops is well-defined if $\Phi_{C,e,e'}(X)$ is a total function. This is always the case for any total semirings (such as Bool,Tropical), rendering our semantics more general than several others [Batz et al. 2022; Dardinier and Müller 2024; Zhang and Kaminski 2022]. For partial semi-rings, extra caution is necessary as $\oplus$ may not always be well-defined. Hence:

(1) We restrict the assignment $x \coloneqq \texttt{nondet()}$, Kleene's star $C^\star$ and nondeterministic choices $\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}$ to total semi-rings only.
(2) We allow only nondeterministic choices of the form $\{\, e \fatsemi C_1 \,\} \,\square\, \{\, e \fatsemi C_2 \,\}$ and loops $C^{\langle e,e' \rangle}$ where the expressions are compatible [Zilberstein 2024, Section A.3], that is, $[\![e_1]\!](\sigma) \oplus [\![e_2]\!](\sigma)$ is defined for any $\sigma \in \Sigma$.

Restricting to compatible expressions allows the use of $\texttt{if}\ (\,\varphi\,)\ \{\, C_1 \,\}\ \texttt{else}\ \{\, C_2 \,\}$ and the guarded loop $\texttt{while}\,(\,\varphi\,)\,\{\, C \,\}$ for *every* semiring. Additionally, the probabilistic choice $\{\, C_1 \,\}\ [\, p \,]\ \{\, C_2 \,\}$ remains well-defined for the partial semiring Prob. For the remainder of the paper, we assume that programs are constructed in this manner, ensuring they are always well-defined.

4 Quantitative Weakest Hyper Pre

4.1 A Quantitative Strongest Post for Weighted Programs

As hinted in Section 2.2, we want our calculus to anticipate the so-called strongest post. Therefore, we define a novel *quantitative strongest post* transformer for wReg.

Definition 4.1 (Quantitative Strongest Post). The *strongest post transformer* $\mathsf{sp}\colon \mathsf{wReg} \longrightarrow (\mathbb{A} \longrightarrow \mathbb{A})$ is defined inductively according to the rules in Table 3 on p. 13, middle column. ◁

Let us show what sp computes semantically, before providing some intuitions on the rules.

THEOREM 4.2 (CHARACTERIZATION OF sp). *For all programs $C \in \mathsf{wReg}$ and final states $\tau \in \Sigma$,*

$$\mathsf{sp}\,[\![C]\!]\,(\mu)\,(\tau) \quad = \quad \bigoplus_{\sigma\in\Sigma} \mu(\sigma) \odot [\![C]\!](\sigma,\tau)\ .$$

Theorem 4.2 guarantees the correct behavior of sp[3] by asserting that it appropriately maps initial quantities to final quantities, including probability distributions and program sets of states. In particular, Table 1 shows that by instantiating our calculus with different semirings we subsume several existing strongest post calculi. Additionally, similarly to [Batz et al. 2022, Table 1], weighted strongest post can handle optimization and combinatorial problems as well, with the main difference to be our calculus moving forward instead of backward.

We contend that our definition of sp is inherently intuitive, extending the classical concept of "reachable sets" to final distributions where the binary notion of reachability is substituted with real values. This inherent intuitiveness is additionally justified by the close connection between weakest pre and strongest post in our framework. To underscore this point, we extend the weighted wp of [Batz et al. 2022, Table 2] to our language wReg, as shown in Table 2 and we revisit Kozen's duality between forward transformers and wp.

C	**wp** $[\![C]\!]$ **(** f **)**
$x \coloneqq e$	$f\,[x/e]$
$x \coloneqq \mathtt{nondet()}$	$\bigoplus_\alpha f\,[x/\alpha]$
$\odot\, w$	$w \odot f$
$C_1 \fatsemi C_2$	$\mathsf{wp}\,[\![C_1]\!]\,(\mathsf{wp}\,[\![C_2]\!]\,(f))$
$\{\,C_1\,\} \mathbin{\square} \{\,C_2\,\}$	$\mathsf{wp}\,[\![C_1]\!]\,(f) \oplus \mathsf{wp}\,[\![C_2]\!]\,(f)$
$C^{\langle e,e'\rangle}$	$\mathsf{lfp}\,X\boldsymbol{.}\ [\![e']\!] \odot f \oplus [\![e]\!] \odot \mathsf{wp}\,[\![C]\!]\,(X)$

Table 2. Rules for weighted wp [Batz et al. 2022, Table 2], extended to wReg.

THEOREM 4.3 (KOZEN [1985] DUALITY). *For all programs C, probability distributions $\mu\colon \Sigma \longrightarrow [0,1]$, and all functions $f \in \mathbb{A}$, we have* $\mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma) = \sum_{\tau\in\Sigma} [\![C]\!](\sigma,\tau) \cdot f(\tau)$.

We now prove a more general version of the duality above for weighted programming.

THEOREM 4.4 (EXTENDED KOZEN DUALITY FOR WEIGHTED PROGRAMMING). *For all programs $C \in \mathsf{wReg}$ and final states $\tau \in \Sigma$, the following equality holds:*

$$\mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma) \quad = \quad \bigoplus_{\tau\in\Sigma} [\![C]\!](\sigma,\tau) \odot f(\tau)\ .$$

We can also prove that the following more symmetrical duality between our sp and wp holds:

THEOREM 4.5 (WEIGHTED sp-wp DUALITY). *For all programs C and all functions $\mu, g \in \mathbb{A}$, we have*

$$\bigoplus_{\tau\in\Sigma} \mathsf{sp}\,[\![C]\!]\,(\mu)\,(\tau) \odot g(\tau) \quad = \quad \bigoplus_{\sigma\in\Sigma} \mu(\sigma) \odot \mathsf{wp}\,[\![C]\!]\,(g)\,(\sigma)\ .$$

[3]It is essential to note that our formulation of sp differs from the one disproven by [Jones 1990, p. 135]. The latter focuses on identifying the most precise assertion for the triples defined in [Jones 1990, p. 124].

Calculus	**Semiring**
Strongest Postcondition [Dijkstra and Scholten 1990]	$\langle\{0,1\}, \vee, \wedge, 0, 1\rangle$
Strongest Liberal Postcondition [Zhang and Kaminski 2022]	$\langle\{0,1\}, \wedge, \vee, 1, 0\rangle$
Quantitative Strongest Post [Zhang and Kaminski 2022]	$\langle\mathbb{R}^{\pm\infty}, \max, \min, -\infty, +\infty\rangle$
Quantitative Strongest Liberal Post [Zhang and Kaminski 2022]	$\langle\mathbb{R}^{\pm\infty}, \min, \max, +\infty, -\infty\rangle$

Table 1. Existing strongest post calculi subsumed via our quantitative strongest post.

In essence, Theorem 4.5 establishes a novel equivalence between forward and backward transformers. An intuition for the probabilistic semiring Prob is that computing the expectation of a quantity g after the program execution—captured in the final distribution $\mathsf{sp}\,[\![C]\!]\,(\mu)$—is analogous to calculating the expected value through $\mathsf{wp}\,[\![C]\!]\,(g)\,(\sigma)$ but with the added nuance of being weighted by the initial distribution μ. In the case of other semirings, the idea is that on the left-hand side all terminating traces originating from μ are aggregated and then g appended. Conversely, on the right-hand side, the process is reversed: we initiate from g and move backward until we reach μ.

Example 4.6. Consider the semiring of formal languages $\mathcal{A} = \langle \mathcal{P}(\{a,b\}^*), \cup, \odot, \emptyset, \{\epsilon\}\rangle$ and the program $C = \{\odot\{a\}\}\,\Box\,\{\odot\{b\}\}$. Let $\mu = \lambda\sigma.\,\{a\}$ and $g = \lambda\sigma.\,\{b\}$ represent the prequantity we aim to prepend and the postquantity we intend to append at the end of the execution, respectively. This results in the following language:

$$
\begin{aligned}
\bigoplus_{\sigma\in\Sigma} \mu(\sigma)\odot \mathsf{wp}\,[\![C]\!]\,(g)\,(\sigma) &= \bigoplus_{\sigma\in\Sigma} \{a\}\odot(\mathsf{wp}\,[\![\odot\{a\}]\!]\,(g)\,(\sigma)\oplus \mathsf{wp}\,[\![\odot\{b\}]\!]\,(g)\,(\sigma)) \\
&= \{a\}\odot(\{ab\}\oplus\{bb\}) \;=\; \{aab, abb\}
\end{aligned}
$$

which is exactly

$$
\begin{aligned}
\bigoplus_{\tau\in\Sigma} \mathsf{sp}\,[\![C]\!]\,(\mu)\,(\tau)\odot g(\tau) &= \bigoplus_{\tau\in\Sigma} (\mathsf{sp}\,[\![\odot\{a\}]\!]\,(\mu)\,(\sigma)\oplus \mathsf{sp}\,[\![\odot\{b\}]\!]\,(\mu)\,(\sigma))\odot\{b\} \\
&= (\{aa\}\oplus\{ab\})\odot\{b\} \;=\; \{aab, abb\}
\end{aligned}
$$

◁

Let us explain the rules in Table 3 individually.

Assignment: The quantitative strongest post $\mathsf{sp}\,[\![x \coloneqq e]\!]\,(f)$ is calculated by considering all possible values α that x could have had before the assignment and summing all evaluations of quantity f under those possible α.

Nondeterministic Assignment: The statement $x \coloneqq \mathtt{nondet()}$ is analogous to $x \coloneqq e$, but without any restriction on the initial value of x, since the assignment is entirely nondeterministic and hence the original value of x cannot be retrieved.

Assume/Weighting: In the assume statement, the strongest post is given by $[\varphi]\cdot f$, where $[\varphi]$ acts as a filter, nullifying states for which the predicate does not hold.

The weighting statement $\odot\, a$ extends the assume rule by allowing any weighting function a. The strongest post for weighting involves scaling the initial quantity f by the weight a.

Sequential Composition: The quantitative strongest post for sequential composition $C_1 \fatsemi C_2$ is obtained by evaluating the second program C_2 starting from the strongest post of the first program C_1. The quantity $\mathsf{sp}\,[\![C_1]\!]\,(f)$ represents the possible states reached with associated weights after executing C_1, and C_2 is then executed from these states.

Nondeterministic Choice: For the nondeterministic choice $\{C_1\}\,\Box\,\{C_2\}$, the strongest post is the sum of the strongest posts of C_1 and C_2. This accounts for the possibility of either program being executed, resulting in a combination of the quantities reached by each.

Iteration: The begin post for the iteration $C^{\langle e,e'\rangle}$ is an extension to the one in [Zhang and Kaminski 2022, Definition 4.1], but generalised to arbitrary weights e, e' instead of predicates. It is thus obtained via loop unrollings

$$
\begin{aligned}
\Psi_f(\mathbb{0})\odot[\![e']\!] &= \mathsf{sp}\,[\![\{\odot\, e \fatsemi \mathtt{diverge}\}\,\Box\,\{\odot\, e'\}]\!]\,(f) \\
\Psi_f^2(\mathbb{0})\odot[\![e']\!] &= \mathsf{sp}\,[\![\{\odot\, e \fatsemi C \fatsemi \{\odot\, e \fatsemi \mathtt{diverge}\}\,\Box\,\{\odot\, e'\}\}\,\Box\,\{\odot\, e'\}]\!]\,(f)
\end{aligned}
$$

which converge to the least fixed point of $\Psi_f(X) = f \oplus \mathsf{sp}\,[\![C]\!]\,\big(X\odot[\![e]\!]\big)$, yielding the rule $\mathsf{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(f) \;=\; \big(\mathsf{lfp}\, X.\, f\oplus \mathsf{sp}\,[\![C]\!]\,\big(X\odot[\![e]\!]\big)\big)\odot[\![e']\!]$

C	$\mathsf{sp}\,[\![C]\!]\,(f)$	$\mathsf{whp}\,[\![C]\!]\,(\mathit{ff})$
$x \coloneqq e$	$\bigoplus_\alpha f\,[x/\alpha] \odot [x = e\,[x/\alpha]]$	$\mathit{ff}\,[x/e]$
$x \coloneqq \mathtt{nondet()}$	$\bigoplus_\alpha f\,[x/\alpha]$	$\lambda f.\, \mathit{ff}(\bigoplus_\alpha f\,[x/\alpha])$
$\odot\, w$	$f \odot w$	$\mathit{ff} \odot w$
$C_1 \,⨟\, C_2$	$\mathsf{sp}\,[\![C_2]\!]\,(\mathsf{sp}\,[\![C_1]\!]\,(f))$	$\mathsf{whp}\,[\![C_1]\!]\,(\mathsf{whp}\,[\![C_2]\!]\,(\mathit{ff}))$
$\{\, C_1 \,\} \square \{\, C_2 \,\}$	$\mathsf{sp}\,[\![C_1]\!]\,(f) \oplus \mathsf{sp}\,[\![C_2]\!]\,(f)$	$\bigoplus_{\nu_1,\nu_2} \mathit{ff}(\nu_1 \oplus \nu_2) \odot \mathsf{whp}\,[\![C_1]\!]\,([\nu_1]) \odot \mathsf{whp}\,[\![C_2]\!]\,([\nu_2])$
$C^{\langle e,e' \rangle}$	$(\mathsf{lfp}\, X.\, f \oplus \mathsf{sp}\,[\![C]\!]\,(X \odot [\![e]\!])\,) \odot [\![e']\!]$	$\lambda f.\, \mathit{ff}((\mathsf{lfp}\, X.\, f \oplus \mathsf{sp}\,[\![C]\!]\,(X \odot [\![e]\!])\,) \odot [\![e']\!])$

Table 3. Rules for defining the quantitative strongest post and weakest hyper pre transformers.

4.2 Quantitative Weakest Hyper Pre

First of all, we show in which sense we can represent hyperproperties via functions. We have already seen that predicates can be encoded via Iverson brackets (Definition 3.4), and decoded by the support set, since every quantity $f : \Sigma \to U$ can be seen as a set of states via $\mathrm{supp}\,(f) = \{\sigma \colon f(\sigma) \neq \mathbb{0}\}$. For example, the set of reachable states starting from $\phi \subseteq \Sigma$ is $\mathrm{supp}\,(\mathsf{sp}\,[\![C]\!]\,([\phi]))$. To encode and decode hyperpredicates, we need to introduce hyper Iverson brackets.

Definition 4.7 (Hyper Iverson Brackets). Given a semiring $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1} \rangle$, for a hyperpredicate $\phi\!\!\!\phi \colon \mathcal{P}(\mathcal{P}(\Sigma))$ we define the hyper Iverson bracket $[\phi\!\!\!\phi] : (\Sigma \to U) \to \mathbb{R}^\infty_{\geq 0}$ by

$$[\phi\!\!\!\phi]\,(f) \;=\; \begin{cases} +\infty & \text{if } \mathrm{supp}\,(f) \in \phi\!\!\!\phi; \qquad \text{and} \qquad 0 \quad \text{otherwise.} \end{cases} \qquad \triangleleft$$

For a hyperquantity ff, its corresponding hyperpredicate is defined by $\mathrm{supp}\,(\mathit{ff}) = \{f \colon \mathit{ff}(f) > 0\}$. We shall remark that hyperpredicates in our setting can represent predicates over quantities, including hyperproperties and predicates over probability distributions.

Definition 4.8 (Quantitative Weakest Hyper Pre). The *quantitative weakest hyper pre transformer* $\mathsf{whp} \colon \mathrm{Reg} \to (\mathbb{AA} \to \mathbb{AA})$ is defined inductively according to the rules in Table 3, right column.

Let us show for some of the rules how the quantitative weakest hyper pre semantics can be developed and understood analogously to Dijkstra's classical weakest preconditions.

Assignment. The weakest pre*condition* of an assignment is given by $\mathsf{wp}\,[\![x \coloneqq e]\!]\,(\psi) \;=\; \psi\,[x/e]$, where $\psi\,[x/e]$ denotes the substitution of the variable x in ψ with the expression e. From a semantic perspective, this replacement can be expressed as $\psi\,[x/e] \;\coloneqq\; \lambda\sigma.\, \psi\Big(\sigma\,[x \mapsto \sigma(e)]\Big)$. In simpler terms, the weakest precondition operates by predicting the operational semantics: it examines whether, given an initial state σ, the final state $\sigma\,[x \mapsto \sigma(e)]$ adheres to the condition ψ.

For *quantitative weakest hyper pre*, a similar approach is taken, but we anticipate the strongest post rather than the operational semantics. Therefore, the value of ff in the resulting distribution (or set of states) after the execution of $x \coloneqq e$ on the initial distribution (or set) f corresponds to ff, but evaluated at the final distribution $\mathsf{sp}\,[\![x \coloneqq e]\!]\,(f) = \bigoplus_\alpha f\,[x/\alpha] \odot [x = e\,[x/\alpha]]$. We thus define the syntactic replacement of the variable x in a hyperquantity ff by $\mathit{ff}\,[x/e] \coloneqq \lambda f.\, \mathit{ff}(\mathsf{sp}\,[\![x \coloneqq e]\!]\,(f))$, yielding the rule $\mathsf{whp}\,[\![x \coloneqq e]\!]\,(\mathit{ff}) \;=\; \mathit{ff}\,[x/e]$

Nondeterministic Assignment: The nondeterministic assignment is analogous to the standard assignment, but now with x ranging over any possible value.

Assume/Weighting. We have $\mathsf{wp}\,[\![\mathtt{assume}\ \varphi]\!]\,(\psi) \;=\; \varphi \wedge \psi$. Indeed, if the initial state σ satisfies the combined precondition $\varphi \wedge \psi$, the execution of $\mathtt{assume}\ \varphi$ entails progression through the assumption of φ. Since the assumption itself does not alter the program state, the process concludes in state σ, which also satisfies the post ψ. Conversely, if σ fails to meet $\varphi \wedge \psi$, the execution of $\mathtt{assume}\ \varphi$ results in either not progressing through the assumption of φ or passing through the assumption but σ not satisfying the post ψ. The *quantitative weakest hyper pre* on an initial distribution (set) f anticipates the strongest post, yielding the rule $\mathsf{whp}\,[\![\texttt{assume}\ \varphi]\!]\,(ff) = \lambda f.\, ff([\varphi] \odot f)$.

To simplify the notation, we introduce the product $\odot$ between quantities and hyperquantities as:

$$ff \odot w \;=\; \lambda f.\, ff(f \odot w) \qquad w \odot ff \;=\; \lambda f.\, ff(w \odot f)\,,$$

leading to the syntactically simpler rule $\mathsf{whp}\,[\![\texttt{assume}\ \varphi]\!]\,(ff) \;=\; ff \odot [\varphi]$. For the more general weighting statement, $\mathsf{whp}\,[\![\odot\, w]\!]\,(ff) \;=\; ff \odot w$ is a generalization, where w can be any quantity.

Nondeterministic Choice. When executing nondeterministic choice $\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}$ on some initial state σ, operationally *either* C_1 *or* C_2 will be executed. Hence, the execution will reach either a final state in which executing C_1 on σ terminates or a final state in which executing C_2 on σ terminates (or no final state if both computations diverge).

The *angelic* weakest pre*condition* of $\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}$ is given by $\mathsf{wp}\,[\![\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}]\!]\,(\psi) \;=\; \mathsf{wp}\,[\![C_1]\!]\,(\psi) \vee \mathsf{wp}\,[\![C_2]\!]\,(\psi)$. Indeed, whenever an initial state σ satisfies the precondition $\mathsf{wp}\,[\![C_1]\!]\,(\psi)$ or $\mathsf{wp}\,[\![C_2]\!]\,(\psi)$, then — either by executing C_1 or C_2 — it is possible that the computation will terminate in some final state satisfying the postcondition ψ.

Moving to hyperquantities, the elimination of nondeterminism occurs because the strongest post $\mathsf{sp}[\![\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}]\!]$ is deterministic. Consequently, the value of ff in the resulting distribution (or set of states) after executing either C_1 or C_2 on the initial distribution (or set) f is

$$\mathsf{whp}\,[\![\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}]\!]\,(ff) \;=\; \bigoplus_{\nu_1, \nu_2 \colon \Sigma \to U} ff(\nu_1 \oplus \nu_2) \odot \mathsf{whp}\,[\![C_1]\!]\,([\nu_1]) \odot \mathsf{whp}\,[\![C_2]\!]\,([\nu_2])\ .$$

Recalling that the final distribution is the combination of $\mathsf{sp}\,[\![C_1]\!]\,(f)$ and $\mathsf{sp}\,[\![C_2]\!]\,(f)$, identifying ν_i such that $\nu_i = \mathsf{sp}\,[\![C_i]\!]\,(f)$ makes computing $ff(\nu_1 \oplus \nu_2)$ sufficient. By aggregating over every ν_i for which $\mathsf{whp}\,[\![C_i]\!]\,([\nu_1])\,(f)$ holds, we ensure that only those ν_i where $\nu_i = \mathsf{sp}\,[\![C_i]\!]\,(f)$ will contribute, making the sum non-zero. Consequently, $ff(\nu_1 \oplus \nu_2)$ precisely equals $ff(\mathsf{sp}\,[\![\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}]\!]\,(f))$.

Remark 4.9. In the case of $\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}$, OL and HHL exhibit forward-style rules that are simpler but not comprehensive. While these rules maintain soundness, completeness necessitates the inclusion of an existential rule. As our approach adopts a weakest pre style calculus aiming for both soundness and completeness, the introduction of the $\bigoplus$ quantification becomes imperative. This quantification mirrors the existential rule utilized in OL and HHL, encompassing all relevant cases. Our rule shares similarities with den Hartog [2002, Definition 6.5.2], although they provide multiple rules depending on the structure of the hyperquantity. Since our paper focuses on semantic assertions, we refrain from analyzing the syntactic structure of hyperquantities. However, we later introduce simpler rules for the class of linear hyperquantities, as outlined in Definition 6.5.

Sequential Composition. What is the anticipated value of ff after executing $C_1 \fatsemi C_2$, i.e. the value of ff after first executing C_1 and then C_2? To answer this, we first anticipate the value of ff after execution of C_2 which gives $\mathsf{whp}\,[\![C_2]\!]\,(ff)$. Then, we anticipate the value of the intermediate quantity $\mathsf{whp}\,[\![C_2]\!]\,(ff)$ after execution of C_1, yielding $\mathsf{whp}\,[\![C_1 \fatsemi C_2]\!]\,(ff) = \mathsf{whp}\,[\![C_1]\!]\,\big(\mathsf{whp}\,[\![C_2]\!]\,(ff)\big)$.

Iteration. The rule for $C^{\langle e,e' \rangle}$ is obtained by anticipating the execution of $C^{\langle e,e' \rangle}$. It is consistent in the sense that it is a solution of the equation:

$$\begin{aligned}\mathsf{whp}[\![C^{\langle e,e' \rangle}]\!] \;&=\; \mathsf{whp}[\![\Big\{ \odot\, e \fatsemi C \fatsemi C^{\langle e,e' \rangle} \Big\} \,\square\, \{\, \odot\, e' \,\}]\!] \\ &=\; \lambda ff\, \lambda f.\, \bigoplus_{\nu} ff(\nu \oplus f \odot [\![e']\!]) \odot \mathsf{whp}\,[\![C]\!]\,\Big(\mathsf{whp}[\![C^{\langle e,e' \rangle}]\!]([\nu])\Big)\,(f \odot [\![e]\!])\end{aligned}$$

Indeed one can show the following.

Proposition 4.10 (Consistency of iteration rule). *Let*

$$\Phi(\mathit{trnsf}) = \lambda \hbar\, \lambda f .\, \bigoplus_{\nu} \hbar(\nu \oplus f \odot [\![e']\!]) \odot \mathsf{whp}\, [\![C]\!]\, (\mathit{trnsf}([\nu]))\, (f \odot [\![e]\!])$$

Then, $\mathsf{whp}[\![C^{\langle e,e'\rangle}]\!]$ *is a fixpoint of the higher order function* $\Phi(\mathit{trnsf})$*, that is:*

$$\Phi(\lambda ff\, \lambda\mu .\, ff(\mathsf{sp}\, [\![C^{\langle e,e'\rangle}]\!]\, (\mu))) = \lambda ff\, \lambda\mu .\, ff(\mathsf{sp}\, [\![C^{\langle e,e'\rangle}]\!]\, (\mu))$$

Remark 4.11. One might attempt a rule for $C^{\langle e,e'\rangle}$ by defining $F(X) = \lambda f .\, X(f \oplus \mathsf{sp}\, [\![C]\!]\, (f \odot [\![e]\!]))$. Intuitively, F takes as input a hyperquantity X, but instead of applying it on a distribution f, it computes one iteration of the loop $\mathsf{sp}\, [\![C]\!]\, (f \odot [\![e]\!])$ and then pass all as argument of X. Recalling that $\Psi_f(X) = f \oplus \mathsf{sp}\, [\![C]\!]\, (X \odot [\![e]\!])$, one can then observe that for every $n \in \mathbb{N}$:

$$\begin{aligned} \lambda f .\, ff(f \odot [\![e']\!]) &= \lambda f .\, ff(\Psi_f(\mathbb{0}) \odot [\![e']\!]) \\ F(\lambda f .\, ff(f \odot [\![e']\!])) &= \lambda f .\, ff(\Psi_f^2(\mathbb{0}) \odot [\![e']\!]) \\ &\vdots \\ F^n(\lambda f .\, ff(f \odot [\![e']\!])) &= \lambda f .\, ff(\Psi_f^{n+1}(\mathbb{0}) \odot [\![e']\!]) \end{aligned}$$

However, it's important to note that in general, $F^n(\lambda f .\, ff(f \odot [\![e']\!]))$ does not form an ascending or descending chain. For example, take $ff = \mathbf{1}_\nu$, where ν is a probability distribution. It's very well possible that $\mathbf{1}_\nu(\Psi_f^k(\mathbb{0}) \odot [\![e']\!]) = \mathbb{1}$ for some k, μ: that is, we anticipate an incomplete proability distribution and find out that it is equal ν. However, at the $k+1$ iteration, the anticipated probability distribution is refined, so that it could be $\Psi_\mu^{k+1}(\mathbb{0}) \odot [\![e']\!] \neq \nu$, leading to a decreasing iterate. Additionally, it's not always desirable to stop at the first fixpoint - as multiple extra iterations might be needed to compute the correct anticipated probability distribution. That said, it is entirely possible that simpler rules exist when restricting ff, see e.g. Table 7. ◁

After having provided an intuition on the rules, let us show that whp does actually anticipate sp.

Theorem 4.12 (Characterization of whp). *For all programs C, hyperquantities $ff \in \mathbb{AA}$ and quantities $f \in \mathbb{A}$:* $\mathsf{whp}\, [\![C]\!]\, (ff)\, (f) = ff(\mathsf{sp}\, [\![C]\!]\, (f))$.

For a given hyperquantity ff and initial quantity μ, $\mathsf{whp}\, [\![C]\!]\, (ff)\, (\mu)$ represents the value assumed by ff in the final quantity reached after the termination of C on μ. Unlike standard wp, which distinguishes between terminating and nonterminating states, whp does not make this distinction. When there are no terminating states, i.e., $\mathsf{sp}\, [\![C]\!]\, (\mu) = \mathbb{0}$, the value of $\mathsf{whp}\, [\![C]\!]\, (ff)\, (\mu)$ is determined by $ff(\mathbb{0})$. The assignment of any desired value to the empty set of states 0 by the hyperquantity ff allows us to express both weakest preconditions and weakest liberal ones.

5 Expressivity

In the preceding sections, we characterized our quantitative weakest hyper pre calculus. In this section, we aim to illustrate the expressive capabilities of the calculus by demonstrating that it subsumes several other logics and calculi.

5.1 An Overview of Several Hoare-Like Logics

We subsume Hyper Hoare Logic for non-probabilistic programs (since HHL is non-probabilistic).

Theorem 5.1 (Subsumption of HHL). *For hyperpredicates $\psi\!\!\!/$, $\phi\!\!\!/$ and non-probabilistic program C:*

$$\models_{\mathrm{hh}} \{\, \psi\!\!\!/ \,\}\; C\; \{\, \phi\!\!\!/ \,\} \quad \text{iff} \quad \mathsf{supp}\,([\psi\!\!\!/]) \subseteq \mathsf{supp}\,(\mathsf{whp}\, [\![C]\!]\, ([\phi\!\!\!/]))$$

As a byproduct, whp subsumes demonic partial correctness, angelic total correctness, partial incorrectness, and total incorrectness (according to the terminology in [Zhang and Kaminski 2022]).

Logic	Syntax	Semantics	Semantics via whp
Hoare Logic (partial correctness)	$\models_{\text{pc}} \{\, P \,\}\; C \;\{\, Q \,\}$	$P \subseteq \text{wlp}[\![C]\!]\,(Q)$	$\Box P \subseteq \text{whp}\,[\![C]\!]\,(\Box Q)$
Lisbon Logic (angelic total correctness)	$\models_{\text{atc}} \{\, P \,\}\; C \;\{\, Q \,\}$	$P \subseteq \text{wp}\,[\![C]\!]\,(Q)$	$\Diamond P \subseteq \text{whp}\,[\![C]\!]\,(\Diamond Q)$
Partial Incorrectness Logic	$\models_{\text{pi}} [\, P \,]\; C \;[\, Q \,]$	$Q \subseteq \text{slp}[\![C]\!]\,(P)$	$\{\neg P\} \subseteq \text{whp}\,[\![C]\!]\,(\Box(\neg Q))$
Incorrectness Logic/Reverse Hoare Logic	$\models_{\text{ti}} [\, P \,]\; C \;[\, Q \,]$	$Q \subseteq \text{sp}\,[\![C]\!]\,(P)$	$\{P\} \subseteq \text{whp}\,[\![C]\!]\,(\lambda\rho.\, Q \subseteq \rho)$

Table 4. Partial and total (in)correctness using classical predicate transformers and whp.

Syntax	Semantics	Semantics via whp
$\not\models_{\text{pc}} \{\, P \,\}\; C \;\{\, Q \,\}$	$P \cap \text{wp}\,[\![C]\!]\,(\neg Q) \neq \emptyset$	$\{P\} \subseteq \text{whp}\,[\![C]\!]\,(\Diamond(\neg Q))$
$\not\models_{\text{atc}} \{\, P \,\}\; C \;\{\, Q \,\}$	$P \cap \text{wlp}[\![C]\!]\,(\neg Q) \neq \emptyset$	$\exists \sigma \in P\boldsymbol{.}\; \{\{\sigma\}\} \subseteq \text{whp}\,[\![C]\!]\,(\Box\neg Q)$
$\not\models_{\text{pi}} [\, P \,]\; C \;[\, Q \,]$	$Q \cap \text{sp}\,[\![C]\!]\,(\neg P) \neq \emptyset$	$\{\neg P\} \subseteq \text{whp}\,[\![C]\!]\,(\Diamond Q)$
$\not\models_{\text{ti}} [\, P \,]\; C \;[\, Q \,]$	$Q \cap \text{slp}[\![C]\!]\,(\neg P) \neq \emptyset$	$\{P\} \subseteq \text{whp}\,[\![C]\!]\,(\lambda\rho.\, Q \cap \neg\rho \neq \emptyset)$

Table 5. Disproving partial and total (in)correctness using classical predicate transformers and whp.

To highlight this, we will utilize the following modality syntax introduced in [Zilberstein 2024]:

$$\Box P \;=\; \lambda\rho\boldsymbol{.}\; [\rho \subseteq P] \qquad \text{and} \qquad \Diamond P \;=\; \lambda\rho\boldsymbol{.}\; [P \cap \rho \neq \emptyset]$$

When reasoning about hyperproperties, we may omit Iverson brackets and write $\psi \subseteq \text{whp}\,[\![C]\!]\,(\phi)$ instead of $\text{supp}\,([\psi]) \subseteq \text{supp}\,\big(\text{whp}\,[\![C]\!]\,([\phi])\big)$. We obtain the relationships in Table 4.

Arguably, Hoare-like logics are designed to be accessible to programmers to prove correctness, whereas reasoning about whp (and HHL, OL) enables better understanding of relationships between different program logics, leading to definitions of new logics, as we will show in the following.

5.2 Disproving Hoare-Like Triples

For example, we can semantically define new triples by falsifying the triples of Table 4, see Table 5.

- $\not\models_{\text{pc}} \{\, P \,\}\; C \;\{\, Q \,\}$: there is some state in P that can terminate in $\neg Q$, and hence it is false that every state in P terminates only in Q (if it terminates at all)
- $\not\models_{\text{atc}} \{\, P \,\}\; C \;\{\, Q \,\}$: there is some state in P that terminates only in $\neg Q$ (if it terminates at all), and hence it is false that every state in P can terminate in Q
- $\not\models_{\text{pi}} [\, P \,]\; C \;[\, Q \,]$: there is some state in Q that is reachable from $\neg P$, and hence it is false that every state in Q is reachable only from P
- $\not\models_{\text{ti}} [\, P \,]\; C \;[\, Q \,]$: there is some state in Q that is reachable only from $\neg P$ (if it is reachable at all), and hence it is false that every state in Q is reachable from P

It remains to define program logics for the newly defined falsifying triples. To this end, one can prove that the existing program logics are actually falsifying program logics. More precisely:

Theorem 5.2 (Falsifying correctness triples via correctness triples).

$$\begin{aligned}
\models_{\text{pc}} \{\, P \,\}\; C \;\{\, Q \,\} \quad &\text{iff} \quad \forall \sigma \in P\boldsymbol{.}\; \not\models_{\text{atc}} \{\, \{\sigma\} \,\}\; C \;\{\, \neg Q \,\} \\
\models_{\text{atc}} \{\, P \,\}\; C \;\{\, Q \,\} \quad &\text{iff} \quad \forall \sigma \in P\boldsymbol{.}\; \not\models_{\text{pc}} \{\, \{\sigma\} \,\}\; C \;\{\, \neg Q \,\} \\
\models_{\text{pi}} [\, P \,]\; C \;[\, Q \,] \quad &\text{iff} \quad \forall \sigma \in Q\boldsymbol{.}\; \not\models_{\text{ti}} [\, \neg P \,]\; C \;[\, \{\sigma\} \,] \\
\models_{\text{ti}} [\, P \,]\; C \;[\, Q \,] \quad &\text{iff} \quad \forall \sigma \in Q\boldsymbol{.}\; \not\models_{\text{pi}} [\, \neg P \,]\; C \;[\, \{\sigma\} \,]
\end{aligned}$$

- $\models_{\text{pc}} \{\, P \,\}\; C \;\{\, Q \,\}$: every state in P can only terminate in Q (if it terminates at all), and hence by starting on any of those state it is false that it can terminate in $\neg Q$
- $\models_{\text{atc}} \{\, P \,\}\; C \;\{\, Q \,\}$: every state in P can terminate in Q, and hence by starting on any of those states it is false that it can terminates only in $\neg Q$ (if it terminates at all)
- $\models_{\text{pi}} [\, P \,]\; C \;[\, Q \,]$: every state in Q is reachable only from P, and hence from any of those states it is false that it is reachable from $\neg P$

- $\models_{\text{ti}} [\, P \,]\ C\ [\, Q \,]$: every state in Q is reachable from P, and hence from any of those states it is false that it is reachable only from $\neg P$

Theorem 5.2 not only demonstrates that existing program logics can generate proofs to falsify other triples but also establishes a crucial "if and only if" relationship. This indicates that not only the current logics are sound, but they are *complete* as well: the existence of an invalid triple implies the presence of a corresponding valid triple that renders the original one invalid. Restating Theorem 5.2 from a negative perspective as below might make it more clear how to practically falsify triples.

Corollary 5.3.
$$\begin{aligned}
\not\models_{\text{pc}} \{\, P \,\}\ C\ \{\, Q \,\} &\quad \text{iff} \quad \exists \sigma \in P.\ \models_{\text{atc}} \{\, \{\sigma\} \,\}\ C\ \{\, \neg Q \,\} \\
\not\models_{\text{atc}} \{\, P \,\}\ C\ \{\, Q \,\} &\quad \text{iff} \quad \exists \sigma \in P.\ \models_{\text{pc}} \{\, \{\sigma\} \,\}\ C\ \{\, \neg Q \,\} \\
\not\models_{\text{pi}} [\, P \,]\ C\ [\, Q \,] &\quad \text{iff} \quad \exists \sigma \in Q.\ \models_{\text{ti}} [\, \neg P \,]\ C\ [\, \{\sigma\} \,] \\
\not\models_{\text{ti}} [\, P \,]\ C\ [\, Q \,] &\quad \text{iff} \quad \exists \sigma \in Q.\ \models_{\text{pi}} [\, \neg P \,]\ C\ [\, \{\sigma\} \,]
\end{aligned}$$

As highlighted by Zhang and Kaminski [2022, p. 20, "Other Triples"], the use of the terms "correctness" and "incorrectness" in naming conventions may be imprecise. Correctness triples can be seen as ∀-properties over preconditions, whereas incorrectness triples exhibit characteristics of ∀-properties over postconditions. Furthermore, it is noteworthy that the falsification of such ∀-triples can be interpreted as ∃-triples, a result that aligns with the expectation that disproving these properties involves finding at least one counterexample. This perspective concurs with the observation made by Cousot [2024, Logic 23] that Incorrectness Logic provides sufficient (though not necessary) conditions to falsify partial correctness triples, thereby demonstrating its greater-than-needed power. Let us show how to practically falsify triples.

Example 5.4 (Backward-Moving Assignment Rule for (Total) Incorrectness Logic). Consider the triple $\models_{\text{ti}} [\, y = 42 \,]\ x \coloneqq 42\ [\, y = x \,]$, obtained by taking as precondition the syntactic replacement of $x = 42$ from the post. As shown in [O'Hearn 2020] with a counterexample, this is not valid. We can prove it by computing a partial incorrectness triple with precondition $y \neq 42$.

Using the rules defined in [Zhang and Kaminski 2022, Table 2, Column 2], we have:

$$\models_{\text{pi}} [\, y \neq 42 \,]\ x \coloneqq 42\ [\, y \neq 42 \vee x \neq 42 \,]$$

This post clearly contains at least one state with $y = x$ (e.g., take a state where $\sigma(x) = \sigma(y) = 0$), which implies $\not\models_{\text{ti}} [\, y = 42 \,]\ x \coloneqq 42\ [\, y = x \,]$ (by Corollary 5.3). ◁

We conclude the section by observing that we have the following connection.

Proposition 5.5 (wp / sp Connection). $P \cap \mathrm{wp}\, [\![C]\!]\, (Q) \neq \emptyset \quad \text{iff} \quad Q \cap \mathrm{sp}\, [\![C]\!]\, (P) \neq \emptyset.$

A simple consequence of the above is the duality $\not\models_{\text{pc}} \{\, P \,\}\ C\ \{\, \neg Q \,\}$ iff $\not\models_{\text{pi}} [\, \neg P \,]\ C\ [\, Q \,]$, which is not surprising, as the duality $\models_{\text{pc}} \{\, P \,\}\ C\ \{\, Q \,\}$ iff $\models_{\text{pi}} [\, \neg P \,]\ C\ [\, \neg Q \,]$ has already been explored in [Zhang and Kaminski 2022, p.22, "Duality"] and again in [Ascari et al. 2023].

5.3 Designing (Falsifying) Hoare-Like Logics via Hyperpredicate Transformers

The observations above indicate that there is no advantage for new program logics to falsify triples from an expressivity point of view, as they can be converted into existing triples via Theorem 5.2. However, one may wonder whether it is possible to design triples that are more useful in practice. In this regard, we emphasize that the design of program logics should follow predicate transformer reasoning. We provide an intuition on how whp aids in reasoning about designing logics (rather than triples). We illustrate this with an example of partial correctness.

Partial Correctness as Classical Predicate Transformers. Partial correctness amounts to a logic that takes $Q \subseteq \mathcal{P}(\Sigma)$ and proves every P such that $P \subseteq \mathrm{wlp}[\![C]\!]\, (Q)$.

Partial Correctness as a Hyperproperty. We observe that partial correctness, as a logic, is a hyperproperty. Indeed, $P \subseteq \mathsf{wlp}[\![C]\!]\,(Q)$ iff $P \in \{S \mid S \subseteq \mathsf{wlp}[\![C]\!]\,(Q)\}$, and this is a predicate over sets of states. Also, by Galois connection, this is equivalent to proving $\mathsf{sp}\,[\![C]\!]\,(P) \subseteq Q$ iff $\mathsf{sp}\,[\![C]\!]\,(P) \in \{S \mid S \subseteq Q\}$, explaining why our whp captures partial correctness (via $P \in \mathsf{whp}\,[\![C]\!]\,(\lambda\rho\boldsymbol{.}\ \rho \subseteq Q)$).

(Dis)proving Partial Correctness, Practically. One may wonder why partial correctness is much easier than our whp calculus. At first glance, it seems that, for a given post Q, one may want to find $\{S \mid S \subseteq \mathsf{wlp}[\![C]\!]\,(Q)\}$. However, the actual logic aims to find just $\mathsf{wlp}[\![C]\!]\,(Q)$ since $\mathsf{wlp}[\![C]\!]\,(Q)$ fully characterizes the original hyperproperty. Even if $\mathsf{wlp}[\![C]\!]\,(Q)$ itself is not found, any $S \subseteq \mathsf{wlp}[\![C]\!]\,(Q)$ allows soundly proving $\models_{\mathrm{pc}} \{\, P \,\}\ C\ \{\, Q \,\}$ by checking $P \subseteq S$. The same reasoning applies to falsify partial correctness triples. Our key insight is that it is enough to find any $\mathsf{wlp}[\![C]\!]\,(Q) \subseteq S$ and then prove $\not\models_{\mathrm{pc}} \{\, P \,\}\ C\ \{\, Q \,\}$ by checking $P \not\subseteq S$. With this in mind, we argue that the most sensible proof system to falsify partial correctness should aim for $\mathsf{wlp}[\![C]\!]\,(Q) \subseteq P$.

So we obtain the following sound and complete falsifying partial correctness logic, which is the same as partial correctness except for the following different rules:

$$\frac{G \Longleftarrow G' \quad \models \{\, G' \,\}\ C\ \{\, F' \,\} \quad F' \Longleftarrow F}{\models \{\, G \,\}\ C\ \{\, F \,\}}\ \text{Antecedence}^{4} \qquad \frac{\forall n\boldsymbol{.}\ \models \{\, p(n+1) \,\}\ C\ \{\, p(n) \,\}}{\models \{\, \forall n.p(n) \,\}\ C^{\star}\ \{\, p(0) \,\}}\ \text{Kleene}$$

We argue that by similar reasoning, it is easy to find falsifying logics for the other triples.

Do we need falsifying logics? It is known from [Zhang and Kaminski 2022, p.22] that $\mathsf{wlp}[\![C]\!]\,(Q) \subseteq S$ corresponds to the contrapositive of Lisbon Logic, i.e., amounts to $\neg S \subseteq \mathsf{wp}\,[\![C]\!]\,(\neg Q)$. This means that, to prove $\not\models_{\mathrm{pc}} \{\, P \,\}\ C\ \{\, Q \,\}$, one should prove $\models_{\mathrm{atc}} \{\, \neg S \,\}\ C\ \{\, \neg Q \,\}$ (possibly keeping $\neg S$ large) and then check $P \not\subseteq S$. Similar reasoning applies if we want to apply Theorem 5.2, and so we argue that reasoning via contrapositive is a lot harder to do for the average programmer.

5.4 Semantics of Nontermination and Unreachability

As discussed in Ascari et al. [2023, Section 5.4], we also show how existing triples capture properties such as must-nontermination, may-termination, unreachability, and reachability, but within our setting. Our initial focus is on illustrating ∀-properties, see Table 6. It is noteworthy that the transition from partial to total involves the negation of the properties under consideration. Specifically, the negation of may-termination corresponds to must-nontermination, and unreachability is the negation of reachability. A useful perspective is to view reachability as the may-termination of backward semantics, while unreachability can be conceptualized as its must-nontermination. By examining their falsification, we derive their dual counterparts, characterized as ∃-properties, see Table 6.

Triple	Semantics	Property
$\models_{\mathrm{pc}} \{\, P \,\}\ C\ \{\, \mathsf{false} \,\}$	$\forall\sigma \in P\boldsymbol{.}\ \nexists\tau\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	Must-Nontermination
$\models_{\mathrm{atc}} \{\, P \,\}\ C\ \{\, \mathsf{true} \,\}$	$\forall\sigma \in P\boldsymbol{.}\ \exists\tau\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	May-Termination
$\models_{\mathrm{pi}} [\, \mathsf{false} \,]\ C\ [\, Q \,]$	$\forall\tau \in Q\boldsymbol{.}\ \nexists\sigma\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	Unreachability
$\models_{\mathrm{ti}} [\, \mathsf{true} \,]\ C\ [\, Q \,]$	$\forall\tau \in Q\boldsymbol{.}\ \exists\sigma\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	Reachability
$\not\models_{\mathrm{pc}} \{\, P \,\}\ C\ \{\, \mathsf{false} \,\}$	$\exists\sigma \in P\boldsymbol{.}\ \exists\tau\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	May-Termination
$\not\models_{\mathrm{atc}} \{\, P \,\}\ C\ \{\, \mathsf{true} \,\}$	$\exists\sigma \in P\boldsymbol{.}\ \nexists\tau\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	Must-Nontermination
$\not\models_{\mathrm{pi}} [\, \mathsf{false} \,]\ C\ [\, Q \,]$	$\exists\tau \in Q\boldsymbol{.}\ \exists\sigma\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	Reachability
$\not\models_{\mathrm{ti}} [\, \mathsf{true} \,]\ C\ [\, Q \,]$	$\exists\tau \in Q\boldsymbol{.}\ \nexists\sigma\boldsymbol{.}\ \tau \in [\![C]\!](\sigma)$	Unreachability

Table 6. ∀-properties on nontermination and unreachability.

We conclude by noting that we are unable to prove must-termination (which is related to demonic total correctness) within our framework. However, we can prove may-nontermination by observing that our calculus subsumes angelic total correctness, and to prove that `while` $(\,\varphi\,)\,\{\,C\,\}$ may not terminate when run from any state in P, it is sufficient to prove $\models_{\mathrm{atc}} \{\, P \wedge \varphi \,\}\ C\ \{\, P \wedge \varphi \,\}$, which is analogous to the Div-While rule of Raad et al. [2024, Fig. 3].

[4] Which replaces the rule of *consequence*.

5.5 Expressing Quantitative Weakest Pre

In this section we show that our calculus subsumes several existing calculi. We define $\mathbf{1}_\sigma(\tau) = \mathbb{1}$ if $\tau = \sigma$ and $\mathbf{1}_\sigma(\tau) = \mathbb{0}$ otherwise.

Nondeterministic Programs. We start by defining hyperquantities subsuming existing angelic weakest pre and demonic weakest liberal pre [Zhang and Kaminski 2022].

Definition 5.6 (Hyper Suprema and Infima). For a given semiring $\mathcal{A} = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1} \rangle$ and a quantity $f\colon \Sigma \longrightarrow U$, we define hyperquantities

$$\curlyvee[f] \triangleq \lambda\mu. \curlyvee_{\sigma \in \mathsf{supp}(\mu)} f(\sigma) \qquad\qquad \curlywedge[f] \triangleq \lambda\mu. \curlywedge_{\sigma \in \mathsf{supp}(\mu)} f(\sigma)\,,$$

that take as input quantities $\mu\colon \Sigma \longrightarrow U$. Intuitively, $\curlyvee[f]$ and $\curlywedge[f]$ map a given μ to the maximum (minimum) value of $f(\sigma)$ where σ is drawn from the support set $\mathsf{supp}\,(\mu)$. ◁

Theorem 5.7 (Subsumption of Quantitative wp, wlp for Nondeterministic Programs [Zhang and Kaminski 2022]). *Let* $\mathcal{A} = \langle \mathbb{R}^{\pm\infty}, \max, \min, \mathbb{0}, \mathbb{1} \rangle$. *For any quantities* g, f *and any program* C *satisfying the syntax of* [Zhang and Kaminski 2022, Section 2]*:*

$$\mathsf{whp}\,[\![C]\!]\left(\curlywedge[f]\right)(\mathbf{1}_\sigma) \;=\; \mathsf{wlp}[\![C]\!]\,(f)\,(\sigma) \qquad \text{and} \qquad \mathsf{whp}\,[\![C]\!]\left(\curlyvee[f]\right)(\mathbf{1}_\sigma) \;=\; \mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma)$$

The result follows from the fact that $\mathsf{whp}\,[\![C]\!]\,(\curlywedge[f])\,(\mathbf{1}_\sigma)$ and $\mathsf{whp}\,[\![C]\!]\,(\curlyvee[f])\,(\mathbf{1}_\sigma)$ compute respectively the maximum and the minimum value of f in the support of $\mathsf{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)$, which is the set of reachable states starting from σ. Our calculus is strictly more expressive than [Zhang and Kaminski 2022] as our syntax is richer and allows to reason about weighted programs as well.

Probabilistic Programs. By employing the expected value hyperquantity, we show how whp subsumes wp and wlp for deterministic and probabilistic programs [Kaminski 2019] as well.

Theorem 5.8 (Subsumption of Quantitative wp, wlp for probabilistic programs [Kaminski 2019]). *Let* $Prob = \langle [0,1], +, \cdot, 0, 1 \rangle$. *For any quantities* g, f *and any* <u>non-non</u>*deterministic program* C*:*

$$\mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[f])\,(\mathbf{1}_\sigma) = \mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma) \quad \textit{and} \quad \mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[f] + 1 - \mathbb{E}[1])\,(\mathbf{1}_\sigma) = \mathsf{wlp}[\![C]\!]\,(f)\,(\sigma).$$

The results stem from our calculus, which computes $\mathbb{E}[f]$ on the final distribution $\mathsf{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)$ using the expected values hyperquantity, which precisely yields $\mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma)$. Additionally, it is known [Kaminski 2019, Theorem 4.25] that for nondeterministic programs $\mathsf{wlp}[\![C]\!]\,(f)\,(\sigma)$ calculates the expected value of f in the final distribution $\mathsf{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)$, but adjusted for the probability of nontermination. This latter probability is in our setting the hyperquantity $1 - \mathbb{E}[1]$.

Probabilistic termination. Since our calculus subsumes many existing quantitative wp calculi such as those of McIver and Morgan [2005]; Zhang and Kaminski [2022], we know that is can also prove probabilistic termination (see Kaminski [2019, Section 6] for a comprehensive overview). For example, almost-sure termination amounts to proving that $\mathsf{wp}\,[\![C]\!]\,(1)\,(\sigma) = 1$, which in our setting is just $\mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[1])\,(\mathbf{1}_\sigma) = \mathbf{1}$. Bounds over expected values, such as those in Hark et al. [2019], are easily handled as well; for example, $\mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[f])\,(\mu) < k$ checks whether the expected value of f after execution of the program is less than k. While at first one may argue that this expressiveness comes at the cost of more complex rules, we will show in Section 6 that when using *linear hyperquantities* (Section 6.3), reasoning via whp is indeed very similar to reasoning via wp.

Nondeterminism, Regular Languages, and Schedulers. While the results above highlight that many existing wp are mere specializations of whp for single initial pre-states, we claim that there are some limitations as well, particularly in how nondeterminism is resolved. The main reason is that all of our transformers, being related to the strongest post sp, cannot detect whether a program C starting from σ diverges for at least one possible execution. Therefore we cannot express demonic wp and angelic wlp. The closest attempt is to define the following hyperquantities.

Definition 5.9 (Demonic Weakest Pre and Angelic Weakest Liberal Pre). Let the ambient semiring be $\mathcal{A} = \langle \mathbb{R}^{\pm\infty}, \max, \min, -\infty, +\infty \rangle$. Given a quantity $f\colon \Sigma \to \mathbb{R}^{\pm\infty}$, we define hyperquantities

$$\bigwedge[f]_{\Downarrow} \quad \triangleq \quad \lambda\mu.\ \bigwedge_{\sigma\in\mathrm{supp}(\mu)} f(\sigma) \ \wedge \bigvee_{\sigma\in\mathrm{supp}(\mu)} +\infty \qquad \text{and} \qquad \bigvee[f]_{\Uparrow} \quad \triangleq \quad \lambda\mu.\ \bigvee_{\sigma\in\mathrm{supp}(\mu)} f(\sigma) \ \vee \bigwedge_{\sigma\in\mathrm{supp}(\mu)} -\infty\ .$$

One can define two novel transformers:

$$\mathsf{wp_{inf}}[\![C]\!]\,(f)\,(\sigma) \triangleq \mathsf{whp}\,[\![C]\!]\left(\bigwedge[f]_{\Downarrow}\right)(\mathbf{1}_{\sigma}) \ \text{ and }\ \mathsf{wlp_{sup}}[\![C]\!]\,(f)\,(\sigma) \triangleq \mathsf{whp}\,[\![C]\!]\left(\bigvee[f]_{\Uparrow}\right)(\mathbf{1}_{\sigma}) \quad \triangleleft$$

Intuitively, $\mathsf{wp_{inf}}[\![C]\!]\,(f)\,(\sigma)$ operates akin to a demonic weakest pre calculus by determining the minimum value of f after the execution of program C starting from σ. However, unlike the demonic weakest pre calculus in [Kaminski 2019], we do not necessarily assign the value bottom $\mathbb{0}$ if the program has a single diverging trace; instead, we do so only when all traces are diverging. Similarly, for $\mathsf{wlp_{sup}}$, our calculus outputs $\mathbb{1}$ if all traces are diverging. In other words, both our $\mathsf{wp_{inf}}$ and angelic $\mathsf{wlp_{sup}}$ attempt to avoid termination whenever possible, mirroring the behavior of the angelic wp and demonic wlp as discussed in [Zhang and Kaminski 2022, Section 6.2].

To better illustrate, let us demonstrate that our demonic weakest pre ($\mathsf{wp_{inf}}$) and angelic weakest liberal pre ($\mathsf{wlp_{sup}}$) transformers differ from those in [Kaminski 2019] through an example.

Example 5.10 (Comparing Nondeterminism). Let dwp and awlp be the demonic weakest pre and angelic weakest liberal pre in [Kaminski 2019], and let $C = \{\,\mathtt{diverge}\,\} \,\square\, \{\,\mathtt{skip}\,\}$. Then:

- $\mathsf{dwp}[\![C]\!]\,([\mathsf{true}]) = [\mathsf{false}] \quad \neq \quad [\mathsf{true}] = \mathsf{wp_{inf}}[\![C]\!]\,([\mathsf{true}])$
- $\mathsf{awlp}[\![C]\!]\,([\mathsf{false}]) = [\mathsf{true}] \quad \neq \quad [\mathsf{false}] = \mathsf{wlp_{sup}}[\![C]\!]\,([\mathsf{false}])$ $\triangleleft$

Conventional treatment of nondeterministic programs in established weakest pre calculi inherently involve schedulers [Kaminski 2019, Definition 3.7] designed to resolve nondeterminism, seeking the maximum or minimum expected value across all possible schedulers. In contrast, our approach aligns with the Incorrectness Logic literature, using Kleene Algebra and strongest-post-style calculi as program semantics [Dardinier and Müller 2024; O'Hearn 2020; Zhang and Kaminski 2022; Zilberstein et al. 2023]: for nondeterministic programs, we treat all choices as if they were executed. To further highlight the differences, using a semantics involving schedulers and extending dwp in the sense of Kaminski [2019] would invalidate the synctactic sugar of branching and loops.

Example 5.11. Let dwp and awlp be the demonic weakest pre and angelic weakest liberal pre of Kaminski [2019]. We extend both for the assume statement, obtaining:

$$\mathsf{dwp}[\![\mathsf{assume}\ \varphi]\!]\,(f) = \varphi \wedge f \qquad \text{and} \qquad \mathsf{awlp}[\![\mathsf{assume}\ \varphi]\!]\,(f) = [\neg\varphi] \vee f$$

We have $\mathsf{dwp}[\![\mathtt{if}\ (\,\mathtt{true}\,)\ \{\,\mathtt{skip}\,\}\ \mathtt{else}\ \{\,\mathtt{skip}\,\}]\!]\,([\mathsf{true}]) = [\mathsf{true}]$, whereas for the seemingly equivalent $\{\,\mathsf{assume\ true} \fatsemi \mathsf{skip}\,\} \,\square\, \{\,\mathsf{assume\ false} \fatsemi \mathsf{skip}\,\}$ we have:

$$\mathsf{dwp}[\![\{\,\mathsf{assume\ true} \fatsemi \mathsf{skip}\,\} \,\square\, \{\,\mathsf{assume\ false} \fatsemi \mathsf{skip}\,\}]\!]\,([\mathsf{true}]) \;=\; [\mathsf{true}] \wedge [\mathsf{false}] \;=\; [\mathsf{false}]$$

Similarly, $\mathsf{awlp}[\![\mathtt{if}\ (\,\mathtt{true}\,)\ \{\,\mathtt{skip}\,\}\ \mathtt{else}\ \{\,\mathtt{skip}\,\}]\!]\,([\mathsf{false}]) = [\mathsf{false}]$ but:

$$\mathsf{awlp}[\![\{\,\mathsf{assume\ true} \fatsemi \mathsf{skip}\,\} \,\square\, \{\,\mathsf{assume\ false} \fatsemi \mathsf{skip}\,\}]\!]\,([\mathsf{false}]) = [\mathsf{false}] \vee [\mathsf{true}] = [\mathsf{true}] \quad \triangleleft$$

Whilst the fact that demonic total correctness is inexpressible in KAT [Kozen 1997] because it lacks a way of reasoning about nontermination [von Wright 2002], here we argue that also angelic partial correctness in the sense of [Kaminski 2019] is inexpressible. This highlights the fact that regular languages, such as KAT variants, are not equivalent to guarded imperative languages in general.

6 Properties

Our quantitative hyper transformers enjoy several *healthiness properties*, some of which are analogous to Dijkstra's, Kozen's, or McIver & Morgan's calculi. In this section, we argue that there exists only one backward hyper predicate transformer, as whp enjoys several properties and dualities that both liberal and non-liberal weakest pre style calculus have.

6.1 Healthiness Properties

Theorem 6.1 (Healthiness Properties of Quantitative Transformers). *For all programs C,* $\mathsf{whp}[\![C]\!]$ *satisfies the following properties:*

(1) Quantitative universal conjunctiveness and disjunctiveness: For any set of hyperquantities $S \subseteq \mathbb{AA}$,

$$\mathsf{whp}\,[\![C]\!]\left(\prod S\right) \;=\; \prod_{f\!f \in S} \mathsf{whp}\,[\![C]\!]\,(f\!f) \qquad \textit{and} \qquad \mathsf{whp}\,[\![C]\!]\left(\sum S\right) \;=\; \sum_{f\!f \in S} \mathsf{whp}\,[\![C]\!]\,(f\!f)$$

(2) k-Strictness: For any $k \in \mathbb{R}^{\infty}_{\geq 0}$, $\mathsf{whp}\,[\![C]\!]\,(\lambda f\boldsymbol{.}\, k) \;=\; \lambda f\boldsymbol{.}\, k$.

(3) Monotonicity: $f\!f \;\leq\; g\!g$ implies $\mathsf{whp}\,[\![C]\!]\,(f\!f) \;\leq\; \mathsf{whp}\,[\![C]\!]\,(g\!g)$.

Quantitative universal conjunctiveness and strictness in the context of wp, as well as the notions of disjunctiveness and co-strictness for wlp, serve as quantitative analogues of Dijkstra and Scholten's original calculi. These properties have been explored in [Zhang and Kaminski 2022, Section 5.1]. We demonstrate that whp exhibits all these characteristics, as the k-strictness of whp implies both strictness and co-strictness. This observation aligns with our intuition that whp functions as both a liberal and a non-liberal calculus.

Healthiness properties are mainly beneficial for conducting compositional proofs. Some of the key properties are outlines below:

Universal (con/dis)junctivenesses. This property allows a complex (hyper)property to be broken down into simpler ones, which can be proved separately. The results can then be soundly recombined to complete the proof of the original complex (hyper)property.

(K-)Strictness. In the context of classical wp, strictness (also known as the "Law of the Excluded Miracle" [Dijkstra 1975]) ensures that no initial state can terminate in a state satisfying "false". Quantitative generalisations of strictness [Kaminski 2019, Definition 4.13], defined as $\mathsf{wp}\,[\![C]\!]\,(0) = 0$, mean that the expected value of the constantly 0 random variable after executing a program C is 0.

In our setting, we can represent strictness by taking $k = 0$: for predicates, it means it is impossible to terminate in a set of states that satisfies the hyperpostcondition "false." Conversely, for $k = +\infty$, we have a generalisation of the so-called co-strictness: any initial precondition will terminate in a postcondition that satisfies the hyperpostcondition "true".

Monotonicity. Larger (hyper)quantities as inputs yield larger (hyper)quantity as results. Monotonicity is a fundamental property that allows compositional reasoning and, for classical weakest pre, it is closely related to the rule of consequence in Hoare logic. An in-depth treatment of this particular connection can be found in Kaminski [2019, p.95]. In our context, unsurprisingly, monotonicity enables the proof of the *Cons* rule from Dardinier and Müller [2024, Fig. 2].

Sub- and superlinearity, extensively studied by Kozen, McIver & Morgan, and Kaminski for probabilistic w(l)p transformers, also find applications in our whp. Notably, our calculus adheres to linearity and, additionally, exhibits multiplicativity.

Theorem 6.2 (Linearity). *For all programs C,* $\mathsf{whp}[\![C]\!]$ *is linear, i.e. for all $f\!f, g\!g \in \mathbb{AA}$ and* non-negative *constants $r \in \mathbb{R}_{\geq 0}$,* $\mathsf{whp}\,[\![C]\!]\,(r \cdot f\!f + g\!g) = r \cdot \mathsf{whp}\,[\![C]\!]\,(f\!f) + \mathsf{whp}\,[\![C]\!]\,(g\!g)$.

C	**whp** $[\![C]\!]$ (ff)
$x \coloneqq e$	$ff\,[x/e]$
$x \coloneqq$ `nondet()`	$\lambda f.\, ff(\bigoplus_{\alpha} f\,[x/\alpha])$
$\odot\, w$	$ff \odot w$
$C_1 \fatsemi C_2$	$\mathsf{whp}\,[\![C_1]\!]\,\big(\mathsf{whp}\,[\![C_2]\!]\,(ff)\big)$
$\{\, C_1 \,\} \,\square\, \{\, C_2 \,\}$	$\mathsf{whp}\,[\![C_1]\!]\,(ff) \oplus \mathsf{whp}\,[\![C_2]\!]\,(ff)$
$C^{\langle e, e' \rangle}$	$\bigoplus_{n \in \mathbb{N}} W_e^n(ff \odot [\![e']\!])$
`if` $(\,\varphi\,)\,\{\, C_1 \,\}$ `else` $\{\, C_2 \,\}$	$\mathsf{whp}\,[\![C_1]\!]\,(ff) \odot [\varphi] \oplus \mathsf{whp}\,[\![C_2]\!]\,(ff) \odot [\neg\varphi]$
$\{\, C_1 \,\}\,[\, p \,]\,\{\, C_2 \,\}$	$\mathsf{whp}\,[\![C_1]\!]\,(ff) \odot p \oplus \mathsf{whp}\,[\![C_2]\!]\,(ff) \odot (1 - p)$
`while` $(\,\varphi\,)\,\{\, C \,\}$	$\bigoplus_{n \in \mathbb{N}} W_\varphi^n(ff \odot [\neg\varphi])$

Table 7. Rules for the weakest hyper pre transformer for linear posts ff. Here, $W_e(X) = \mathsf{whp}\,[\![C]\!]\,(X) \odot [\![e]\!]$

Theorem 6.3 (Multiplicativity). *For all programs C, $\mathsf{whp}[\![C]\!]$ is multiplicative, i.e. for all $ff, gg \in \mathbb{AA}$ and* non-negative *constants $r \in \mathbb{R}_{\geq 0}$,* $\mathsf{whp}\,[\![C]\!]\,(r \cdot ff \cdot gg) = r \cdot \mathsf{whp}\,[\![C]\!]\,(ff) \cdot \mathsf{whp}\,[\![C]\!]\,(gg)$.

6.2 Relationship between Liberal and Non-liberal Transformers

Various dualities between wp and wlp have been explored extensively in the literature. In Dijkstra's classical calculus, the duality relationship is expressed as $\mathsf{wp}\,[\![C]\!]\,(\psi) = \neg\mathsf{wlp}[\![C]\!]\,(\neg\psi)$. In quantitative settings, particularly in Kozen's and McIver & Morgan's work on probabilistic programs, this duality extends to $\mathsf{wp}\,[\![C]\!]\,(f) = 1 - \mathsf{wlp}[\![C]\!]\,(1 - f)$ for 1-bounded functions f. This concept is further generalized to $\mathsf{wp}\,[\![C]\!]\,(f) = -\mathsf{wlp}[\![C]\!]\,(-f)$ in the case of non-probabilistic programs and unbounded quantities, as demonstrated in Zhang and Kaminski [2022, Theorem 5.3].

In this section, we argue that there exists only a single whp calculus that behaves both as a non-liberal and a liberal transformer.

Theorem 6.4 (Liberal–Non-liberal Duality). *For any program C and any k−bounded hyperquantity ff, we have* $\mathsf{whp}\,[\![C]\!]\,(ff) \;=\; k - \mathsf{whp}\,[\![C]\!]\,(k - ff)$.

As a consequence of the liberal–non-liberal duality of Theorem 6.4, for hyperproperties we have:

$$\phi\!\!\!/ \implies \mathsf{whp}\,[\![C]\!]\,(\psi\!\!\!/) \qquad \text{iff} \qquad \mathsf{whp}\,[\![C]\!]\,(\neg\psi\!\!\!/) \implies \neg\phi\!\!\!/ \,.$$

6.3 Linear Hyperquantities

In this section, we explore a specific category of hyperquantities from which we can deduce simplified rules akin to established wp calculi.

Definition 6.5 (Linear Hyperquantities). A hyperquantity $ff \in \mathbb{AA}$ is *linear* if for any quantity $f \in \mathbb{A}$

$$ff(r \cdot g \oplus f) \quad = \quad r \cdot ff(g) \;\oplus\; ff(f)\,.$$

Theorem 6.6 (Weakest Hyper Pre for Linear Hyperquantities). *For linear hyperquantities $ff \in \mathbb{AA}$, the simpler rules in* Table 7 *are valid.*

Similarly to other quantitative settings [Kaminski 2019; McIver and Morgan 2005; Zhang and Kaminski 2022], the loop rule can be defined via a least fixed point of the characteristic function.

*Definition 6.7 (*whp−characteristic function*).* The whp−*characteristic function* (of $C^{\langle e,e' \rangle}$ w.r.t. ff) is:

$$\Phi_{ff}(X) \;=\; ff \odot [\![e']\!] \oplus \mathsf{whp}\,[\![C]\!]\,(X) \odot [\![e]\!]. \qquad \triangleleft$$

Indeed, we observe that $\mathsf{whp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mathit{ff}) = \mathsf{lfp}\,X.\,\Phi_{\mathit{ff}}(X)$ holds true within the natural order of the provided semiring. When examining the semiring $\langle \mathbb{R}^{\pm\infty}, \max, \min, -\infty, +\infty\rangle$, our calculus closely resembles the quantitative wp as described in Zhang and Kaminski [2022], albeit in a more expressive context. Further, by adopting $\langle \mathbb{R}^{\pm\infty}, \min, \max, +\infty, -\infty\rangle$, we derive rules analogous to quantitative wlp from Zhang and Kaminski [2022]. Notably, in the latter semiring, the natural order is reversed compared to the semiring $\langle \mathbb{R}^{\pm\infty}, \max, \min, -\infty, +\infty\rangle$. In essence, for $\langle \mathbb{R}^{\pm\infty}, \min, \max, +\infty, -\infty\rangle$, the least fixed point resulting from our iteration rule aligns with the rule of wlp defined through the greatest fixed point in Zhang and Kaminski [2022].

Among linear hyperquantities we have all those in Example 2.4 and of Section 5.5. Additionally, we contend that by combining these properties, we can extend our reasoning to encompass other hyperquantities, such as the covariance of a random variable.

Example 6.8 (Covariance).

$$\begin{aligned}
\mathsf{whp}\,[\![C]\!]\,(\mathsf{Cov}[f,g]) &= \mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[fg] - \mathbb{E}[f]\cdot\mathbb{E}[g]) \\
&= \mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[fg]) - \mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[f]\cdot\mathbb{E}[g]) && \text{(by Theorem 6.2)} \\
&= \mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[fg]) - \mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[f])\cdot\mathsf{whp}\,[\![C]\!]\,(\mathbb{E}[g]) && \text{(by Theorem 6.3)}
\end{aligned}$$

6.4 Loops rules for linear hyperquantities

Reasoning about loops is undecidable, even for classical properties. Previously, we have shown that our whp calculus uses least fixed points, which is often impractical. In this section, we show how to derive simpler rules that can aid in whp reasoning for loops. For linear hyperquantities, we obtain an inductive invariant based rule similar to the existing ones for quantitative transformers [Zhang and Kaminski 2022, Theorem 7.1].

Theorem 6.9 (Quantitative Inductive Reasoning for whp). *For any program C and any linear hyperquantity ff, we have:*

$$\Phi_{\mathit{ff}}(\mathit{ii}) \preceq \mathit{ii} \implies \mathsf{whp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mathit{ff}) \preceq \mathit{ii},$$

where $\Phi_{\mathit{ff}}(X) = \mathit{ff} \odot [\![e']\!] \oplus \mathsf{whp}\,[\![C]\!]\,(X) \odot [\![e]\!]$ is the characteristic function of $C^{\langle e,e'\rangle}$ w.r.t. ff.

As a corollary, one can derive simpler rules for guarded loops, for example, the analogue of Theorem 5.4 of Kaminski [2019], but in our hyper setting.

Corollary 6.10 (Quantitative Inductive Rule for `while`).

$$\frac{\mathit{ff} \odot [\neg\varphi] \oplus \mathsf{whp}\,[\![C]\!]\,(\mathit{ii}) \odot [\varphi] \;\preceq\; \mathit{ii} \;\preceq\; \mathit{gg} \qquad \mathit{ff} \text{ is linear}}{\mathsf{whp}\,[\![\mathtt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(\mathit{ff}) \;\preceq\; \mathit{gg}}\;\text{while}-\text{whp}$$

We shall observe that Corollary 6.10 subsumes *both* while-wp and while-wlp of Zhang and Kaminski [2022, Theorem 7.1]. This depends on the choice of the semiring: $\langle \mathbb{R}^{\pm\infty}, \min, \max, +\infty, -\infty\rangle$ for wp, and $\langle \mathbb{R}^{\pm\infty}, \max, \min, -\infty, +\infty\rangle$ for wlp.

Let us provide an intuition over while−whp in our quantitative hyper setting, for example taking into account the semiring $\mathsf{Prob} = \langle [0,1], +, \cdot, 0, 1\rangle$ and the expected value hyperquantity $\mathit{ff} = \mathbb{E}[f]$. Intuitively, the rule while−whp requires finding an invariant ii that satisfies three conditions:

(1) $\mathit{ii} \preceq \mathit{gg}$, meaning that gg is overapproximating the invariant ii;
(2) $\mathbb{E}[f]\cdot[\neg\varphi] \preceq \mathit{ii}$, meaning that the expected value of f, when evaluated in the filtered probability distribution (i.e., the loop is executed at most 0 times), is bounded by ii;
(3) $\mathsf{whp}\,[\![C]\!]\,(\mathit{ii})\cdot[\varphi] \preceq \mathit{ii}$, meaning that for any initial probability distribution μ, the value of ii computed over this initial distribution will be greater than or equal to the value of ii after performing one more iteration and computing it over the resulting distribution.

By induction, conditions (2) and (3), which represent the first premise of while-whp, imply that $\mathbb{i}$ overapproximates the expected value $\mathbb{E}[f]$ computed in the final probability distribution after the loop execution. Indeed, starting from the base case in (2), we assume for the inductive step that $\mathbb{i}$ over-approximates the expected value after n loop iterations. By condition (3), $\mathbb{i}$ is also an upper bound for $\text{whp}\,[\![C]\!]\,(\mathbb{i}) \cdot [\varphi] \leq \mathbb{i}$, meaning that it over-approximates the probability distribution obtained after $n+1$ iterations.

Condition (1) ensures that the initial expected value $\mathbb{g}$ overapproximates $\mathbb{i}$, and thus $\mathbb{g}$ computed in the initial probability distribution overapproximates the final expected value $\mathbb{E}[f]$.

We showcase an example of induction reasoning that extends [Kaminski 2019, Example 5.5] by taking into account probability distributions instead of single states.

Example 6.11 (Upper Bounds on whp*).* Consider the probabilistic loop $C_{\text{geo}} = x \coloneqq x+1^{\langle 0.5, 0.5\rangle}$ modeling a geometric distribution. We want to prove that $\mathbb{E}[x+1]$ is an upper bound of the expected value $\mathbb{E}[x]$ after executing C_{geo}. We have:

$$\mathbb{E}[x] \cdot [\![0.5]\!] + \text{whp}\,[\![x \coloneqq x+1]\!]\,(\mathbb{E}[x+1]) \cdot [\![0.5]\!] \;=\; \mathbb{E}[x \cdot 0.5] + \mathbb{E}[(x+2)\cdot 0.5] \;=\; E[x+1],$$

and hence by Corollary 6.10 we conclude that $\text{whp}\,[\![C_{\text{geo}}]\!]\,(\mathbb{E}[x]) \leq \mathbb{E}[x+1]$, i.e., $\mathbb{E}[x+1]$ (evaluated in the initial probability distribution) is an upper bound on $\mathbb{E}[x]$ (evaluated in the final probability distribution) after executing C_{geo}.

7 Case Studies

In this section, we demonstrate the efficacy of quantitative weakest hyper pre reasoning. We use the annotation style on the right to express that $g = \text{whp}\,[\![C]\!]\,(f)$ and furthermore that $g' = g$.

$$\begin{array}{l} {}^{=}/\!/\!/\; g' \\ /\!/\!/\; g \\ C \\ /\!/\!/\; f \end{array}$$

7.1 Proving hyperproperties

In this section we show how to prove noninterference [Goguen and Meseguer 1982] and generalized noninterference [McCullough 1987; McLean 1996] within whp.

NI. Noninterference, also known as observational nondeterminism [Clarkson et al. 2014, Equation 6], amounts to proving that any two executions of the program with the same low-sensitivity inputs must have the same low outputs. This can be formalised by defining $\text{low}(l) \triangleq \lambda S.\, \forall \sigma_1, \sigma_2 \in S.\, \sigma_1(l) = \sigma_2(l)$ and proving $\text{low}(l) \subseteq \text{whp}\,[\![C]\!]\,(\text{low}(l))$. For example consider the program and the whp annotations in Figure 3. The program satisfies NI since $\text{low}(l) \subseteq \lambda S.\, \forall \sigma_1, \sigma_2 \in S.\, \sigma_1(h) > 0 \wedge \sigma_2(h) > 0 \implies \sigma_1(l) = \sigma_2(l)$.

GNI. Generalized noninterference is a weaker property of NI: it permits two executions of the program with identical low-sensitivity inputs to yield different low outputs, provided that the discrepancy does not arise from their secret input. This concept can be formally expressed by defining $\text{glow}(l) \triangleq \lambda S.\, \forall \sigma_1, \sigma_2 \in S.\, \exists \sigma \in S.\, \sigma(h) = \sigma_1(h) \wedge \sigma(l) = \sigma_2(l)$, where σ denotes a potential third execution sharing the same secret input as σ_1 but producing the same low output as σ_2. GNI can be proved by checking $\text{low}(l) \subseteq \text{whp}\,[\![C]\!]\,(\text{glow}(l))$. For example consider the program and the whp annotations in Figure 4. The program satisfies GNI since $\text{low}(l) \subseteq \lambda S.\, \forall \sigma_1, \sigma_2 \in \{\sigma\,[y/\alpha] \mid \sigma \in S\}.\, \exists \sigma \in x\{\sigma\,[y/\alpha] \mid \sigma \in S\}.\, \sigma(h) = \sigma_1(h) \wedge \sigma(y+h) = \sigma_2(y+h)$.

$\overset{=}{/\!\!/\!\!/}\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(h) > 0 \land \sigma_2(h) > 0$
$\implies \sigma_1(l) = \sigma_2(l)$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in (h > 0)(S).\ \sigma_1(l) = \sigma_2(l)$
assume $h > 0$
$\overset{=}{/\!\!/\!\!/}\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(l) = \sigma_2(l)$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(l+1) = \sigma_2(l+1)$
$l := l + 1$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(l) = \sigma_2(l)$

Fig. 3. Proving noninterference

$\overset{=}{/\!\!/\!\!/}\ \lambda S.\ \forall \sigma_1, \sigma_2 \in \{\sigma[y/\alpha] \mid \sigma \in S\}.$
$\exists \sigma \in \{\sigma[y/\alpha] \mid \sigma \in S\}.\ \sigma(h) = \sigma_1(h) \land \sigma(y+h) = \sigma_2(y+h)$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in \exists_\alpha\ S[y/\alpha].$
$\exists \sigma \in \exists_\alpha\ S[y/\alpha].\ \sigma(h) = \sigma_1(h) \land \sigma(y+h) = \sigma_2(y+h)$
$y :=$ nondet()
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \exists \sigma \in S.\ \sigma(h) = \sigma_1(h) \land \sigma(y+h) = \sigma_2(y+h)$
$l := y + h$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \exists \sigma \in S.\ \sigma(h) = \sigma_1(h) \land \sigma(l) = \sigma_2(l)$

Fig. 4. Proving generalized noninterference (GNI)

7.2 Disproving hyperproperties

As pointed in Section 2, evaluating whether a program satisfies a specific hyperproperty necessitates proving two HHL triples. For instance, when tackling noninterference, one must attempt to establish *both* $\models_{\text{hh}} \{\ \text{low}(l)\ \}\ C_{\text{ni}}\ \{\ \text{low}(l)\ \}$ and $\models_{\text{hh}} \{\ Q\ \}\ C_{\text{ni}}\ \{\ \neg\text{low}(l)\ \}$ (for some $Q \implies \text{low}(l)$). In this section, we illustrate the advantage of our calculus by disproving NI and GNI.

NI. Disproving NI amounts to proving $\text{low}(l) \not\subseteq \text{whp}\ [\![C]\!]\ (\text{low}(l))$, which is true for the program in Figure 5. For example, take $S = \{\sigma_1, \sigma_2\}$ such that $\sigma_1(l) = \sigma_2(l) = 0$ and $\sigma_1(h) = 1 \neq \sigma_2(h) = 2$. Clearly $S \in \text{low}(l)$ but $S \notin \text{whp}\ [\![C]\!]\ (\text{low}(l))$.

$\overset{=}{/\!\!/\!\!/}\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(h) > 0 \land \sigma_2(h) > 0 \implies \sigma_1(l+h) = \sigma_2(l+h)$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in (h > 0)(S).\ \sigma_1(l+h) = \sigma_2(l+h)$
assume $h > 0$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(l+h) = \sigma_2(l+h)$
$l := l + h$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \sigma_1(l) = \sigma_2(l)$

Fig. 5. Disproving noninterference

GNI. Disproving GNI amounts to prove $\text{low}(l) \not\subseteq \text{whp}\ [\![C]\!]\ (\text{glow}(l))$, which is true for the program in Figure 6. For example, take $S = \{\sigma_1, \sigma_2\}$ such that $\sigma_1(l) = \sigma_2(l) = 0$ and $\sigma_1(h) = 1 \neq \sigma_2(h) = 100$. Clearly $S \in \text{low}(l)$ but $S \notin \text{whp}\ [\![C]\!]\ (\text{glow}(l))$.

$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in A = \{\sigma[y/\alpha] \mid \sigma \in S, \alpha \in [0, 10]\}.\ \exists \sigma \in A.\ \sigma(h) = \sigma_1(h) \land \sigma(y+h) = \sigma_2(y+h)$
$y :=$ nondet()
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in A = \{\sigma \mid \sigma \in S \land \sigma(y) \in [0, 10]\}.\ \exists \sigma \in A.\ \sigma(h) = \sigma_1(h) \land \sigma(y+h) = \sigma_2(y+h)$
assume $0 \leq y \leq 10$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \exists \sigma \in S.\ \sigma(h) = \sigma_1(h) \land \sigma(y+h) = \sigma_2(y+h)$
$l := y + h$
$/\!\!/\!\!/\ \lambda S.\ \forall \sigma_1, \sigma_2 \in S.\ \exists \sigma \in S.\ \sigma(h) = \sigma_1(h) \land \sigma(l) = \sigma_2(l)$

Fig. 6. Disproving generalized noninterference

7.3 Quantitative reasoning

In this section, we demonstrate how whp enables quantitative reasoning.

7.3.1 Quantitative Information Flow. Consider the program C_{qif} in Figure 7. Similarly to [Zhang and Kaminski 2022, Section 8.1], we want to infer what is the maximum initial value that the secret variable h can have, by observing a final value l' for the low-sensitive variable l. By using whp, it is sufficient to consider the hyperpostquantity $f\!\!f_{l'} = \lambda f.\ \Upsilon_\tau\ ([l = l'] \odot f)(\tau)(h)$. Indeed, whp $[\![C_{\text{qif}}]\!]\ (f\!\!f_{l'})\ (h)$ tells, what is the maximum value of sp $[\![C_{\text{qif}}]\!]\ (h)\ (\tau)$ among those final states τ where the value l' has been observed. Since we know from [Zhang and Kaminski 2022] that sp $[\![C_{\text{qif}}]\!]\ (f)\ (\tau)$ produces the maximum initial value of h, we have that whp $[\![C_{\text{qif}}]\!]\ (f\!\!f_{l'})\ (h)$ correctly yields the maximum initial value of h. For example, whp $[\![C_{\text{qif}}]\!]\ (f\!\!f_{80})\ (h) = 7$, meaning that if we observe 80 as the value of l, we know that initially h would have been at most 7.

```
/// λf. ⋎_σ ([99 = l′ ∧ h > 7] ∨ [80 = l′ ∧ h ≤ 7])(σ) ⊙ f(σ)
if ( h > 7 ) {   /// λf. ⋎_τ ([99 = l′] ⊙ f)(τ)
    l := 99
} else {   /// λf. ⋎_τ ([80 = l′] ⊙ f)(τ)
    l := 80
}
/// λf. ⋎_τ ([l = l′] ⊙ f)(τ)
```

Fig. 7. Computing quantitative information flow

$$/\!/\!/ \bigoplus_{n\in\mathbb{N}} \mathbb{E}[(1+n)^2] \odot 0.5^{n+1} - \Big(\bigoplus_{n\in\mathbb{N}} \mathbb{E}[1+n] \odot 0.5^{n+1}\Big)^2$$

$x := 1$

$$/\!/\!/ \bigoplus_{n\in\mathbb{N}} \mathbb{E}[(x+n)^2] \odot 0.5^{n+1} - \Big(\bigoplus_{n\in\mathbb{N}} \mathbb{E}[x+n] \odot 0.5^{n+1}\Big)^2$$

$(x := x + 1)^{\langle \frac{1}{2}, \frac{1}{2} \rangle}$

$/\!/\!/ \; \mathbb{E}[x^2] - \mathbb{E}[x]^2$

$/\!/\!/ \; \mathrm{Cov}[x, x]$

Fig. 8. Computing the variance of a random variable

7.3.2 Variance. We show how to compute the variance of a random variable using whp. Let's consider the following gaming scenario: a player flips a fair coin continuously until a head appears. To assess the variance in the number of flips required to conclude the game, we model this scenario with the program in Figure 8. We leverage Example 6.8 to compute whp $[\![x := x + 1^{\langle \frac{1}{2}, \frac{1}{2} \rangle}]\!]$ $(\mathrm{Cov}[x, x])$ compositionally, by computing whp $[\![x := x + 1^{\langle \frac{1}{2}, \frac{1}{2} \rangle}]\!]$ $(\mathbb{E}[x^2])$ and whp $[\![x := x + 1^{\langle \frac{1}{2}, \frac{1}{2} \rangle}]\!]$ $(\mathbb{E}[x])$ individually, obtaining:

$$\begin{aligned} \mathrm{whp}\, [\![x := x + 1^{\langle \frac{1}{2}, \frac{1}{2} \rangle}]\!] \left(\mathbb{E}[x^2] - \mathbb{E}[x]^2\right) &= \bigoplus_{n\in\mathbb{N}} W^n_{0.5}(\mathbb{E}[x^2] \odot 0.5) - \Big(\bigoplus_{n\in\mathbb{N}} W^n_{0.5}(\mathbb{E}[x] \odot 0.5)\Big)^2 \\ &= \bigoplus_{n\in\mathbb{N}} \mathbb{E}[(x+n)^2] \odot 0.5^{n+1} - \Big(\bigoplus_{n\in\mathbb{N}} \mathbb{E}[x+n] \odot 0.5^{n+1}\Big)^2 \end{aligned}$$

Finally, we take as input any probability distribution μ and compute the variance via:

$$\begin{aligned} \mathrm{whp}\, [\![C]\!]\, (\mathrm{Cov}[x, x])\, (\mu) &= \Big(\bigoplus_{n\in\mathbb{N}} \mathbb{E}[(1+n)^2] \odot 0.5^{n+1} - \Big(\bigoplus_{n\in\mathbb{N}} \mathbb{E}[1+n] \odot 0.5^{n+1}\Big)^2\Big)(\mu) \\ &= \sum (1+n)^2 \cdot 0.5^{n+1} - \Big(\sum (1+n) \cdot 0.5^{n+1}\Big)^2 \;=\; 6 - 4 = 2\,. \end{aligned}$$

We contend that employing whp offers the advantage of mechanization and compositional computation without necessitating specialized knowledge of probability theory.

7.4 Automation

Unsurprisingly, our whp calculus (in its full generality) cannot be fully automated, since we generalize existing undecidable calculi, expressing both termination and reachability properties for a Turing-complete computational model—both of which are known to be undecidable [Rice 1953; Turing 1936].

For this reason, we have proposed a fully theoretical framework, providing a holistic view of different program logics and serving as a foundation for future tools to automate quantitative proofs. This approach is common in foundational program logic research such as Hoare Logic, Probabilistic PDL, Incorrectness Logic, Hyper Hoare Logic, and Outcome Logic.

Nevertheless, we believe that our calculi are at least syntactically mechanizable. Accordingly, we plan to investigate an expressive "assertion" language for hyperquantities, such as the one proposed by Batz et al. [2021] for quantitative reasoning about probabilistic programs. This would allow us to prove *relative completeness* in the sense of Cook [1978], i.e., decidability modulo checking whether $\mathbb{g} \preceq \mathbb{f}$ holds, where $\mathbb{g}, \mathbb{f}$ may contain suprema and infima. A similar result (decidability modulo checking a logical implication) is well known for classical predicate transformer and Hoare Logic [Cook 1978]. Once such a relatively complete language is found, we expect it will be possible to fully automate whp reasoning for syntactic fragments of the programming and the assertion language.

8 Related Work

Relational program logics. Relational Hoare Logics were initially introduced by Benton [2004]. Subsequently, several extensions emerged, including to reason about probabilistic programs via couplings [Barthe et al. 2009]. Later, Maillard et al. [2019b], proposed a general framework for developing relational program logics with effects based on Dijkstra Monads [Maillard et al. 2019a]. While effective, this framework is limited to 2-properties and thus does not apply to, e.g., monotonicity and transitivity, which are properties of *more than* two executions.

D'Osualdo et al. [2022]; Sousa and Dillig [2016] introduced logics for k-safety properties, but they cannot prove liveness. Dickerson et al. [2022] introduced the first logic tailored for $\forall^*\exists^*$-hyperproperties, enabling, among others, proof and disproof of k-safety properties. Nonetheless, it has limited under-approximation capabilities: e.g., it does not suport incorrectness à la O'Hearn [2020], and cannot disprove triples within the same logic. For instance, it cannot disprove GNI, a task which can only be completed by—to the best of our knowledge—HHL, OL, and our framework.

Unified Program Logics. Similar to Outcome Logic (OL) [Zilberstein et al. 2023, 2024] and Weighted Programming [Batz et al. 2022], our calculus utilizes semirings to capture branch weights. This approach enables the development of a weakest-pre style calculus for Outcome Logic. While OL is relatively complete [Zilberstein 2024], the derivations are not always straightforward. Weakest Hyper-pre can be used to *mechanically* derive OL triples with the weakest precondition for a given postcondition. Weakest Hyper-pre also subsumes Hyper Hoare Logic [Dardinier and Müller 2024], which is similar to OL, but specialized to nondeterministic programs.

Our approach surpasses Weighted Programming by facilitating reasoning about multiple outcomes. Our calculus also supports quantitative reasoning, demonstrating its versatility by encompassing various existing quantitative wp instances through the adaptation of hyperquantities.

Predicate Transformers. These were first introduced by Dijkstra [1976]; Dijkstra and Scholten [1990], who created propositional weakest pre- and strongest postcondition calculi. Kozen [1985]; McIver and Morgan [2005] lifted these to a quantitative setting, introducing Probabilistic Propositional Dynamic Logic and weakest preexpectations for computing expected values over probabilistic programs. Many variants of weakest preexpectation now exist [Batz et al. 2018; Kaminski 2019]. We build on this line of work by extending these predicate transformers to hyperproperties. This gives us the flexibility to express a broader range of quantitative properties, as shown in Section 7.

9 Conclusion

Recent years have seen a focus on logics for proving properties other than classical partial correctness. E.g., program *security* is a *hyperproperty*, and *incorrectness* must *witness a faulty execution.*

Recent work on Outcome Logic [Zilberstein 2024; Zilberstein et al. 2023, 2024] and Hyper Hoare Logic [Dardinier and Müller 2024] has shown that all of these properties can be captured via a single proof system. In this paper, we build upon those logics, but approach the problem using quantitative predicate transformers. This has allowed us to create a single calculus that can be used to *prove*, but also *disprove*, a variety of correctness properties. In addition, it can be used to derive advanced quantitative properties for programs too, such as variance in probabilistic programs.

The predicate transformer approach has two key benefits. First, it provides a calculus to mechanically derive specifications. Second, it finds the *most precise* pre, so as to remove guesswork around obtaining a precondition in the aforementioned logics. As we have demonstrated, this brings about new ways of proving—and disproving—hyperproperties for a variety of program types.

References

Flavio Ascari, Roberto Bruni, Roberta Gori, and Francesco Logozzo. 2023. Sufficient Incorrectness Logic: SIL and Separation SIL. arXiv:2310.18156 [cs.LO]

Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. 2009. Formal certification of code-based cryptographic proofs. *SIGPLAN Not.* 44, 1 (jan 2009), 90–101. https://doi.org/10.1145/1594834.1480894

Kevin Batz, Adrian Gallus, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Tobias Winkler. 2022. Weighted Programming: A Programming Paradigm for Specifying Mathematical Models. *Proc. ACM Program. Lang.* 6, OOPSLA1 (2022), 1–30.

Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2021. Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–30.

Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. 2018. Quantitative Separation Logic. *CoRR* abs/1802.10467 (2018). arXiv:1802.10467 http://arxiv.org/abs/1802.10467

Nick Benton. 2004. Simple Relational Correctness Proofs for Static Analyses and Program Transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '04)* (proceedings of the 31st acm sigplan-sigact symposium on principles of programming languages (popl '04) ed.). ACM, 43. https://www.microsoft.com/en-us/research/publication/simple-relational-correctness-proofs-for-static-analyses-and-program-transformations/

Roberto Bruni, Roberto Giacobazzi, Roberta Gori, and Francesco Ranzato. 2021. A Logic for Locally Complete Abstract Interpretations. In *LICS*. IEEE, 1–13.

Edmund Melson Clarke. 1979. Programming Language Constructs for Which It Is Impossible To Obtain Good Hoare Axiom Systems. *J. ACM* 26, 1 (jan 1979), 129–147. https://doi.org/10.1145/322108.322121

Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal Logics for Hyperproperties. In *Principles of Security and Trust*, Martín Abadi and Steve Kremer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *J. Comput. Secur.* 18, 6 (2010), 1157–1210.

Stephen A. Cook. 1978. Soundness and Completeness of an Axiom System for Program Verification. *SIAM J. Comput.* 7 (1978), 70–90.

Patrick Cousot. 2024. Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation. *Proc. ACM Program. Lang.* 8, POPL, Article 7 (jan 2024), 34 pages. https://doi.org/10.1145/3632849

Thibault Dardinier and Peter Müller. 2024. Hyper Hoare Logic: (Dis-)Proving Program Hyperproperties. *Proc. ACM Program. Lang.* 8, PLDI, Article 207 (jun 2024), 25 pages. https://doi.org/10.1145/3656437

Edsko de Vries and Vasileios Koutavas. 2011. Reverse Hoare Logic. In *SEFM (Lecture Notes in Computer Science, Vol. 7041)*. Springer, 155–171.

Jerry den Hartog. 2002. *Probabilistic Extensions of Semantical Models.* Ph. D. Dissertation. Vrije Universiteit Amsterdam. https://core.ac.uk/reader/15452110

Robert Dickerson, Qianchuan Ye, Michael K. Zhang, and Benjamin Delaware. 2022. RHLE: Modular Deductive Verification of Relational ∀∃ Properties. In *Programming Languages and Systems: 20th Asian Symposium, APLAS 2022, Auckland, New Zealand, December 5, 2022, Proceedings* (Auckland, New Zealand). Springer-Verlag, Berlin, Heidelberg, 67–87. https://doi.org/10.1007/978-3-031-21037-2_4

Edsger Wybe Dijkstra. 1975. Guarded Commands, Nondeterminacy and Formal Derivation of Programs. *Commun. ACM* 18, 8 (1975), 453–457.

Edsger W. Dijkstra. 1976. *A Discipline of Programming.* Prentice-Hall. I–XVII, 1–217 pages.

Edsger W. Dijkstra and Carel S. Scholten. 1990. *Predicate Calculus and Program Semantics.* Springer-Verlag, Berlin, Heidelberg.

Emanuele D'Osualdo, Azadeh Farzan, and Derek Dreyer. 2022. Proving hypersafety compositionally. *Proc. ACM Program. Lang.* 6, OOPSLA2, Article 135 (oct 2022), 26 pages. https://doi.org/10.1145/3563298

Patrice Godefroid, Aditya V. Nori, Sriram K. Rajamani, and Sai Deep Tetali. 2010. Compositional may-must program analysis: unleashing the power of alternation. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Madrid, Spain) *(POPL '10)*. Association for Computing Machinery, New York, NY, USA, 43–56. https://doi.org/10.1145/1706299.1706307

J. A. Goguen and J. Meseguer. 1982. Security Policies and Security Models. In *1982 IEEE Symposium on Security and Privacy*. 11–11. https://doi.org/10.1109/SP.1982.10014

Jonathan S. Golan. 2003. Semirings and Affine Equations over Them: Theory and Applications. https://api.semanticscholar.org/CorpusID:117996620

Marcel Hark, Benjamin Lucien Kaminski, Jürgen Giesl, and Joost-Pieter Katoen. 2019. Aiming low is harder: induction for lower bounds in probabilistic program verification. *Proc. ACM Program. Lang.* 4, POPL, Article 37 (dec 2019), 28 pages. https://doi.org/10.1145/3371105

C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. 12, 10 (1969).

Claire Jones. 1990. *Probabilistic Non-Determinism.* Ph. D. Dissertation. University of Edinburgh, UK.

Benjamin Lucien Kaminski. 2019. *Advanced Weakest Precondition Calculi for Probabilistic Programs.* Ph. D. Dissertation. RWTH Aachen University, Germany.

Georg Karner. 2004. Continuous monoids and semirings. *Theoretical Computer Science* 318, 3 (2004), 355–372. https://doi.org/10.1016/j.tcs.2004.01.020

Donald E. Knuth. 1992. Two Notes on Notation. *Am. Math. Monthly* 99, 5 (May 1992), 403–422.

Dexter Kozen. 1985. A Probabilistic PDL. *J. Comput. System Sci.* 30, 2 (1985), 162–178.

Dexter Kozen. 1997. Kleene Algebra with Tests. *ACM Trans. Program. Lang. Syst.* 19, 3 (may 1997), 427–443. https://doi.org/10.1145/256167.256195

Kenji Maillard, Danel Ahman, Robert Atkey, Guido Martínez, Cătălin Hriţcu, Exequiel Rivas, and Éric Tanter. 2019a. Dijkstra monads for all. *Proc. ACM Program. Lang.* 3, ICFP, Article 104 (jul 2019), 29 pages. https://doi.org/10.1145/3341708

Kenji Maillard, Cătălin Hriţcu, Exequiel Rivas, and Antoine Van Muylder. 2019b. The next 700 relational program logics. *Proc. ACM Program. Lang.* 4, POPL, Article 4 (dec 2019), 33 pages. https://doi.org/10.1145/3371072

Petar Maksimović, Caroline Cronjäger, Andreas Lööw, Julian Sutherland, and Philippa Gardner. 2023. Exact Separation Logic: Towards Bridging the Gap Between Verification and Bug-Finding. In *37th European Conference on Object-Oriented Programming (ECOOP 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 263)*, Karim Ali and Guido Salvaneschi (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 19:1–19:27. https://doi.org/10.4230/LIPIcs.ECOOP.2023.19

Daryl McCullough. 1987. Specifications for Multi-Level Security and a Hook-Up. In *1987 IEEE Symposium on Security and Privacy*. 161–161. https://doi.org/10.1109/SP.1987.10009

Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems.* Springer.

J. McLean. 1996. A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering* 22, 1 (1996), 53–67. https://doi.org/10.1109/32.481534

Bernhard Möller, Peter O'Hearn, and Tony Hoare. 2021. On Algebra of Program Correctness and Incorrectness. In *Relational and Algebraic Methods in Computer Science*, Uli Fahrenberg, Mai Gehrke, Luigi Santocanale, and Michael Winter (Eds.). Springer International Publishing, Cham, 325–343.

Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic Predicate Transformers. *ACM Trans. Program. Lang. Syst.* 18, 3 (1996), 325–353.

Peter W. O'Hearn. 2020. Incorrectness Logic. *Proc. ACM Program. Lang.* 4, POPL, Article 10 (Dec. 2020), 32 pages. https://doi.org/10.1145/3371078

Kevin R. O'Neill, Michael R. Clarkson, and Stephen Chong. 2006. Information-Flow Security for Interactive Programs. In *Proceedings of the 19th IEEE Workshop on Computer Security Foundations (CSFW '06)*. IEEE Computer Society, USA, 190–201. https://doi.org/10.1109/CSFW.2006.16

David Michael Ritchie Park. 1969. Fixpoint Induction and Proofs of Program Properties, Vol. 5. Machine intelligence.

Vaughan R. Pratt. 1976. Semantical Considerations on Floyd-Hoare Logic. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*. 109–121. https://doi.org/10.1109/SFCS.1976.27

Azalea Raad, Josh Berdine, Hoang-Hai Dang, Derek Dreyer, Peter O'Hearn, and Jules Villard. 2020. Local Reasoning About the Presence of Bugs: Incorrectness Separation Logic. In *Computer Aided Verification*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer International Publishing, Cham, 225–252.

Azalea Raad, Julien Vanegue, and Peter O'Hearn. 2024. Compositional Non-Termination Proving. https://www.soundandcomplete.org/papers/Unter.pdf

H.G. Rice. 1953. Classes of recursively enumerable sets and their decision problems. *Trans. Amer. Math. Soc.* 74 (1953), 358–366. https://doi.org/10.2307/1990888

Marcelo Sousa and Isil Dillig. 2016. Cartesian hoare logic for verifying k-safety properties. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Santa Barbara, CA, USA) *(PLDI '16)*. Association for Computing Machinery, New York, NY, USA, 57–69. https://doi.org/10.1145/2908080.2908092

Alan Turing. 1936. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* 42, 1 (1936), 230–265. https://doi.org/10.2307/2268810

Joakim von Wright. 2002. From Kleene Algebra to Refinement Algebra. In *International Conference on Mathematics of Program Construction.* https://api.semanticscholar.org/CorpusID:2003560

Linpeng Zhang and Benjamin Lucien Kaminski. 2022. Quantitative Strongest Post: A Calculus for Reasoning about the Flow of Quantitative Information. *Proc. ACM Program. Lang.* 6, OOPSLA1 (2022), 1–29.

Noam Zilberstein. 2024. A Relatively Complete Program Logic for Effectful Branching. arXiv:2401.04594 [cs.LO]

Noam Zilberstein, Derek Dreyer, and Alexandra Silva. 2023. Outcome Logic: A Unifying Foundation of Correctness and Incorrectness Reasoning. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 93 (Apr 2023), 29 pages. https://doi.org/10.1145/3586045

Noam Zilberstein, Angelina Saliling, and Alexandra Silva. 2024. Outcome Separation Logic: Local Reasoning for Correctness and Incorrectness with Computational Effects. *Proc. ACM Program. Lang.* 8, OOPSLA1 (Apr 2024). https://doi.org/10.1145/3649821

A Quantitative Strongest Post and Weakest Pre

A.1 Proof of Soundness for sp, Thereom 4.2

THEOREM 4.2 (CHARACTERIZATION OF sp). *For all programs $C \in$ wReg and final states $\tau \in \Sigma$,*

$$\mathsf{sp}\,[\![C]\!]\,(\mu)\,(\tau) \quad = \quad \bigoplus_{\sigma\in\Sigma} \mu(\sigma) \odot [\![C]\!](\sigma,\tau)\ .$$

PROOF. We prove Theorem 4.2 by induction on the structure of C. For the induction base, we have the atomic statements:

The assignment $x \coloneqq e$: We have

$$\begin{aligned}
\mathsf{sp}\,[\![x \coloneqq e]\!]\,(f)\,(\tau) &= \Big(\bigoplus_{\alpha} f\,[x/\alpha] \odot [x = e\,[x/\alpha]]\,\Big)(\tau) \\
&= \bigoplus_{\alpha\colon\, \tau(x)=\tau(e[x/\alpha])} f\,[x/\alpha]\,(\tau) \\
&= \bigoplus_{\alpha\colon\, \tau(x)=\tau(e[x/\alpha])} f(\tau\,[x/\alpha]) \\
&= \bigoplus_{\alpha\colon\, \tau[x/\alpha][x/\tau(e[x/\alpha])]=\tau} f(\tau\,[x/\alpha]) \\
&= \bigoplus_{\alpha\colon\, \tau[x/\alpha][x/\tau[x/\alpha](e)]=\tau} f(\tau\,[x/\alpha]) \\
&= \bigoplus_{\sigma\in\Sigma,\sigma[x/\sigma(e)]=\tau} f(\sigma) \qquad\qquad \text{(by taking } \sigma = \tau\,[x/\alpha]) \\
&= \bigoplus_{\sigma\in\Sigma} f(\sigma) \odot [\sigma\,[x/\sigma(e)] = \tau] \\
&= \bigoplus_{\sigma\in\Sigma} f(\sigma) \odot [\![x \coloneqq e]\!](\sigma,\tau)\ .
\end{aligned}$$

The nondeterministic assignment $x \coloneqq \mathtt{nondet()}$: We have

$$\begin{aligned}
\mathsf{sp}\,[\![x \coloneqq \mathtt{nondet()}]\!]\,(f)\,(\tau) &= \Big(\bigoplus_{\alpha} f\,[x/\alpha]\,\Big)(\tau) \\
&= \bigoplus_{\alpha} f(\tau\,[x/\alpha]) \\
&= \bigoplus_{\sigma\in\Sigma,\exists\alpha\bullet\,\tau[x/\alpha]=\sigma} f(\sigma) \qquad\qquad \text{(by taking } \sigma = \tau\,[x/\alpha]) \\
&= \bigoplus_{\sigma\in\Sigma} f(\sigma) \odot \bigoplus_{\alpha\in\mathbb{N}} [\sigma\,[x/\alpha] = \tau] \\
&= \bigoplus_{\sigma\in\Sigma} f(\sigma) \odot [\![x \coloneqq \mathtt{nondet()}]\!](\sigma,\tau)\ .
\end{aligned}$$

The weighting $\odot\, w$: We have

$$\begin{aligned}
\mathsf{sp}\,[\![\odot\, w]\!]\,(f)\,(\tau) &= (f \odot w)(\tau) \\
&= f(\tau) \odot w(\tau) \\
&= \bigoplus_{\sigma\in\Sigma} f(\sigma) \odot w(\tau) \odot [\sigma = \tau]
\end{aligned}$$

$$
\begin{aligned}
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot w(\sigma) \odot [\sigma = \tau] \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![\odot\, w]\!](\sigma, \tau) \,.
\end{aligned}
$$

This concludes the proof for the atomic statement.

Induction Hypothesis: For arbitrary but fixed programs C, C_1, C_2, we proceed with the inductive step on the composite statements.

The sequential composition $C_1 \fatsemi C_2$: We have

$$
\begin{aligned}
\mathrm{sp}\, [\![C_1 \fatsemi C_2]\!]\, (f)\, (\tau) &= \mathrm{sp}\, [\![C_2]\!] \left(\mathrm{sp}\, [\![C_1]\!]\, (f)\right) (\tau) \\
&= \bigoplus_{\sigma' \in \Sigma} \mathrm{sp}\, [\![C_1]\!]\, (f)\, (\sigma') \odot [\![C_2]\!](\sigma', \tau) && \text{(by I.H. on } C_2) \\
&= \bigoplus_{\sigma' \in \Sigma} \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![C_1]\!](\sigma, \sigma') \odot [\![C_2]\!](\sigma', \tau) && \text{(by I.H. on } C_1) \\
&= \bigoplus_{\sigma \in \Sigma} \bigoplus_{\sigma' \in \Sigma} f(\sigma) \odot [\![C_1]\!](\sigma, \sigma') \odot [\![C_2]\!](\sigma', \tau) && \text{(by commutativity of } \oplus) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \bigoplus_{\sigma' \in \Sigma} [\![C_1]\!](\sigma, \sigma') \odot [\![C_2]\!](\sigma', \tau) && \text{(by distributivity of } \odot) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![C_1 \fatsemi C_2]\!](\sigma, \tau) \,.
\end{aligned}
$$

The nondeterministic choice $\{ C_1 \} \square \{ C_2 \}$: We have

$$
\begin{aligned}
\mathrm{sp}\, [\![\{ C_1 \} \square \{ C_2 \}]\!]\, (f)\, (\tau) &= \mathrm{sp}\, [\![C_1]\!]\, (f) \oplus \mathrm{sp}\, [\![C_2]\!]\, (f) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![C_1]\!](\sigma, \tau) \oplus \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![C_2]\!](\sigma, \tau) && \text{(by I.H. on } C_1, C_2) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \left([\![C_1]\!](\sigma, \tau) \oplus [\![C_2]\!](\sigma, \tau)\right) && \text{(by distributivity of } \odot) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![\{ C_1 \} \square \{ C_2 \}]\!](\sigma, \tau) \,.
\end{aligned}
$$

The Iteration $C^{\langle e, e' \rangle}$: Let

$$
\Psi_f(X) \;=\; f \oplus \mathrm{sp}\, [\![C]\!] \left(X \odot [\![e]\!]\right) \,,
$$

be the sp-characteristic function of the iteration $C^{\langle e, e' \rangle}$ with respect to any preanticipation f and

$$
F(X)(\sigma, \tau) \;=\; \sigma(e) \odot \left(\bigoplus_{\sigma' \in \Sigma} [\![C]\!](\sigma, \sigma') \odot X(\sigma', \tau) \right) \oplus \sigma(e') \odot [\sigma = \tau] \,,
$$

be the denotational semantics characteristic function of the loop $C^{\langle e, e' \rangle}$ for any input $\sigma, \tau \in \Sigma$. We first prove by induction on m that, for all $\tau \in \Sigma, f \in \mathbb{A}$ we have:

$$
\bigoplus_{\sigma \in \Sigma} \Psi_f^m(\mathbb{0})(\sigma) \odot \sigma(e) \odot [\![C]\!](\sigma, \tau) \;=\; \bigoplus_{\sigma \in \Sigma} \Psi^m_{\lambda \sigma'. f(\sigma) \odot \sigma(e) \odot [\![C]\!](\sigma, \sigma')}(\mathbb{0})(\tau) \,. \tag{1}
$$

For the induction base $m = 0$, consider the following:

$$
\bigoplus_{\sigma \in \Sigma} \Psi_f^0(\mathbb{0})(\sigma) \odot \sigma(e) \odot [\![C]\!](\sigma, \tau) \;=\; \mathbb{0}
$$

$$= \bigoplus_{\sigma \in \Sigma} \Psi^{0}_{\lambda\sigma'.f(\sigma)\odot\sigma(e)\odot\llbracket C\rrbracket(\sigma,\sigma')}(\mathbb{0})(\tau) .$$

As induction hypothesis, we have for arbitrary but fixed m and all $\tau \in \Sigma, f \in \mathbb{A}$

$$\bigoplus_{\sigma \in \Sigma} \Psi^{m}_{f}(\mathbb{0})(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) = \bigoplus_{\sigma \in \Sigma} \Psi^{m}_{\lambda\sigma'.f(\sigma)\odot\sigma(e)\odot\llbracket C\rrbracket(\sigma,\sigma')}(\mathbb{0})(\tau) .$$

For the induction step $m \longrightarrow m+1$, consider the following:

$$\begin{aligned}
&\bigoplus_{\sigma \in \Sigma} \Psi^{m+1}_{f}(\mathbb{0})(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&= \bigoplus_{\sigma \in \Sigma} \big(f \oplus \mathrm{sp}\,\llbracket C\rrbracket \left(\Psi^{m}_{f}(\mathbb{0}) \odot \llbracket e\rrbracket\right)\big)(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \oplus \mathrm{sp}\,\llbracket C\rrbracket \left(\Psi^{m}_{f}(\mathbb{0}) \odot \llbracket e\rrbracket\right)(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&\qquad\qquad \text{(by distributivity of } \odot) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \oplus \Big(\bigoplus_{\sigma' \in \Sigma} \Psi^{m}_{f}(\mathbb{0})(\sigma') \odot \sigma'(e) \odot \llbracket C\rrbracket(\sigma',\sigma)\Big) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&\qquad\qquad \text{(by I.H. on } C) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \oplus \Big(\bigoplus_{\sigma' \in \Sigma} \Psi^{m}_{\lambda\sigma''.f(\sigma')\odot\sigma'(e)\odot\llbracket C\rrbracket(\sigma',\sigma'')}(\mathbb{0})(\sigma)\Big) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&\qquad\qquad \text{(by I.H. on } m) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&\qquad \oplus \bigoplus_{\sigma \in \Sigma} \Big(\bigoplus_{\sigma' \in \Sigma} \Psi^{m}_{\lambda\sigma''.f(\sigma')\odot\sigma'(e)\odot\llbracket C\rrbracket(\sigma',\sigma'')}(\mathbb{0})(\sigma)\Big) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \qquad \text{(by associativity of } \oplus) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \\
&\qquad \oplus \bigoplus_{\sigma \in \Sigma} \bigoplus_{\sigma' \in \Sigma} \Psi^{m}_{\lambda\sigma''.f(\sigma')\odot\sigma'(e)\odot\llbracket C\rrbracket(\sigma',\sigma'')}(\mathbb{0})(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \qquad \text{(by distributivity of } \odot) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \quad \oplus \quad \bigoplus_{\sigma \in \Sigma} \bigoplus_{\sigma' \in \Sigma} \Psi^{m}_{\lambda\sigma''.f(\sigma)\odot\sigma(e)\odot\llbracket C\rrbracket(\sigma,\sigma'')}(\mathbb{0})(\sigma') \odot \sigma'(e) \odot \llbracket C\rrbracket(\sigma',\tau) \\
&\qquad\qquad \text{(by commutativity of } \oplus) \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \llbracket C\rrbracket(\sigma,\tau) \quad \oplus \quad \mathrm{sp}\,\llbracket C\rrbracket \left(\Psi^{m}_{\lambda\sigma''.f(\sigma)\odot\sigma(e)\odot\llbracket C\rrbracket(\sigma,\sigma'')}(\mathbb{0}) \odot \llbracket e\rrbracket\right)(\tau) \\
&\qquad\qquad \text{(by I.H. on } C \text{ and associativity of } \oplus) \\
&= \bigoplus_{\sigma \in \Sigma} \Psi^{m+1}_{\lambda\sigma''.f(\sigma)\odot\sigma(e)\odot\llbracket C\rrbracket(\sigma,\sigma'')}(\mathbb{0})(\tau) \\
&= \bigoplus_{\sigma \in \Sigma} \Psi^{m+1}_{\lambda\sigma'.f(\sigma)\odot\sigma(e)\odot\llbracket C\rrbracket(\sigma,\sigma')}(\mathbb{0})(\tau)
\end{aligned}$$

This concludes the induction on m. We now prove by induction on n that, for all $\tau \in \Sigma, f \in \mathbb{A}$

$$\Psi^{n}_{f}(\mathbb{0})(\tau) \odot \tau(e') = \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot F^{n}(\mathbb{0})(\sigma,\tau) . \tag{2}$$

For the induction base $n = 0$, consider the following:

$$\Psi^{0}_{f}(\mathbb{0})(\tau) \odot \tau(e') = \mathbb{0}$$

$$= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot F^0(\mathbb{0})(\sigma, \tau) .$$

As induction hypothesis, we have for arbitrary but fixed n and all $\tau \in \Sigma, f \in \mathbb{A}$

$$\Psi_f^n(\mathbb{0})(\tau) \odot \tau(e') = \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot F^n(\mathbb{0})(\sigma, \tau) .$$

For the induction step $n \longrightarrow n+1$, consider the following:

$$\begin{aligned}
&\Psi_f^{n+1}(\mathbb{0})(\tau) \odot \tau(e') \\
&= \left(f \oplus \mathrm{sp}\, [\![C]\!] \left(\Psi_f^n(\mathbb{0}) \odot [\![e]\!]\right)\right)(\tau) \odot \tau(e') \\
&= f(\tau) \odot \tau(e') \oplus \mathrm{sp}\, [\![C]\!] \left(\Psi_f^n(\mathbb{0}) \odot [\![e]\!]\right)(\tau) \odot \tau(e') \\
&= f(\tau) \odot \tau(e') \oplus \bigoplus_{\sigma \in \Sigma} \Psi_f^n(\mathbb{0})(\sigma) \odot \sigma(e) \odot [\![C]\!](\sigma, \tau) \odot \tau(e') && \text{(by I.H. on } C\text{)} \\
&= f(\tau) \odot \tau(e') \oplus \bigoplus_{\sigma \in \Sigma} \Psi^n_{\lambda\sigma'.f(\sigma)\odot\sigma(e)\odot[\![C]\!](\sigma,\sigma')}(\mathbb{0})(\tau) \odot \tau(e') && \text{(by Equation 1)} \\
&= f(\tau) \odot \tau(e') \oplus \bigoplus_{\sigma \in \Sigma} \bigoplus_{\sigma' \in \Sigma} f(\sigma) \odot \sigma(e) \odot [\![C]\!](\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau) && \text{(by I.H. on } n\text{)} \\
&= f(\tau) \odot \tau(e') \oplus \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} [\![C]\!](\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau) && \text{(by distributivity of } \odot\text{)} \\
&= \Big(\bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e') \odot [\sigma = \tau]\Big) \oplus \Big(\bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} [\![C]\!](\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau)\Big) \\
&= \Big(\bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} [\![C]\!](\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau)\Big) \oplus \Big(\bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sigma(e') \odot [\sigma = \tau]\Big) \\
& && \text{(by commutativity of } \oplus\text{)} \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \left(\Big(\sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} [\![C]\!](\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau)\Big) \oplus \sigma(e') \odot [\sigma = \tau]\right) \\
& && \text{(by associativity of } \oplus \text{ and distributivity of } \odot\text{)} \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot F^{n+1}(\mathbb{0})(\sigma, \tau) .
\end{aligned}$$

This concludes the induction on n. Now we have:

$$\begin{aligned}
\mathrm{sp}\, [\![C^{\langle e,e' \rangle}]\!]\, (f)\, (\tau) &= \big(\mathrm{lfp}\, X.\, f \oplus \mathrm{sp}\, [\![C]\!] \left(X \odot [\![e]\!]\right)\big)(\tau) \odot [\![e']\!](\tau) \\
&= \big(\sup_{n \in \mathbb{N}} \Psi_f^n(0)(\tau)\big) \odot \tau(e') && \text{(by Kleene's fixpoint theorem)} \\
&= \sup_{n \in \mathbb{N}} \Psi_f^n(0)(\tau) \odot \tau(e') \\
&= \sup_{n \in \mathbb{N}} \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot F^n(\mathbb{0})(\sigma, \tau) && \text{(by Equation 2)} \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot \sup_{n \in \mathbb{N}} F^n(\mathbb{0})(\sigma, \tau) && \text{(by continuity of } \lambda X.\, \textstyle\bigoplus_\sigma f(\sigma) \odot X(\sigma, \tau)\text{)} \\
&= \bigoplus_{\sigma \in \Sigma} f(\sigma) \odot [\![C^{\langle e,e' \rangle}]\!](\sigma, \tau) . && \text{(by Kleene's fixpoint theorem)}
\end{aligned}$$

and this concludes the proof. □

C	$\mathbf{wp}\,[\![C]\!]\,(f)$
$x \coloneqq e$	$f\,[x/e]$
$x \coloneqq \texttt{nondet()}$	$\bigoplus_{\alpha} f\,[x/\alpha]$
$\odot\, w$	$w \odot f$
$C_1 \,⨟\, C_2$	$\mathsf{wp}\,[\![C_1]\!]\,\big(\mathsf{wp}\,[\![C_2]\!]\,(f)\big)$
$\{\, C_1 \,\} \,\Box\, \{\, C_2 \,\}$	$\mathsf{wp}\,[\![C_1]\!]\,(f) \oplus \mathsf{wp}\,[\![C_2]\!]\,(f)$
$C^{\langle e, e' \rangle}$	$\mathsf{lfp}\, X_{\bullet}\, [\![e']\!] \odot f \oplus [\![e]\!] \odot \mathsf{wp}\,[\![C]\!]\,(X)$

Table 8. Rules for the weakest pre transformer.

A.2 A Weakest Pre Calculus for wReg

Theorem 4.4 (Extended Kozen Duality For Weighted Programming). *For all programs $C \in$ wReg and final states $\tau \in \Sigma$, the following equality holds:*

$$\mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma) \quad = \quad \bigoplus_{\tau \in \Sigma} \quad [\![C]\!](\sigma, \tau) \odot f(\tau)\ .$$

Proof. We define our weighted wp in Table 8. We prove Theorem 4.4 by induction on the structure of C. For the induction base, we have the atomic statements:

The assignment $x \coloneqq e$: We have

$$\begin{aligned}
\mathsf{wp}\,[\![x \coloneqq e]\!]\,(f)\,(\sigma) &= f\,[x/e]\,(\sigma) \\
&= f(\sigma\,[x/\sigma(e)]) \\
&= \bigoplus_{\tau \in \Sigma} [\sigma\,[x/\sigma(e)] = \tau] \odot f(\tau) \\
&= \bigoplus_{\tau \in \Sigma} [\![x \coloneqq e]\!](\sigma, \tau) \odot f(\tau)\ .
\end{aligned}$$

The nondeterministic assignment $x \coloneqq \texttt{nondet()}$: We have

$$\begin{aligned}
\mathsf{wp}\,[\![x \coloneqq \texttt{nondet()}]\!]\,(f)\,(\sigma) &= \Big(\bigoplus_{\alpha} f\,[x/\alpha]\Big)(\sigma) \\
&= \bigoplus_{\alpha} f(\sigma\,[x/\alpha]) \\
&= \bigoplus_{\tau \in \Sigma, \exists \alpha_{\bullet}\, \sigma[x/\alpha] = \tau} f(\tau) && \text{(by taking } \tau = \sigma\,[x/\alpha]\text{)} \\
&= \bigoplus_{\tau \in \Sigma} \bigoplus_{\alpha \in \mathbb{N}} [\sigma\,[x/\alpha] = \tau] \odot f(\tau) \\
&= \bigoplus_{\tau \in \Sigma} [\![x \coloneqq \texttt{nondet()}]\!](\sigma, \tau) \odot f(\tau)\ .
\end{aligned}$$

The weighting $\odot\, w$: We have

$$\begin{aligned}
\mathsf{wp}\,[\![\odot\, w]\!]\,(f)\,(\sigma) &= (w \odot f)(\sigma) \\
&= w(\sigma) \odot f(\sigma) \\
&= \bigoplus_{\tau \in \Sigma} w(\sigma) \odot [\sigma = \tau] \odot f(\sigma)
\end{aligned}$$

$$\begin{aligned} &= \bigoplus_{\tau\in\Sigma} w(\sigma) \odot [\sigma = \tau] \odot f(\tau) \\ &= \bigoplus_{\tau\in\Sigma} [\![\odot\, w]\!](\sigma,\tau) \odot f(\tau)\ . \end{aligned}$$

This concludes the proof for the atomic statement.

Induction Hypothesis: For arbitrary but fixed programs C, C_1, C_2, we proceed with the inductive step on the composite statements.

The sequential composition $C_1 \mathbin{⨾} C_2$: We have

$$\begin{aligned} \mathsf{wp}\,[\![C_1 ⨾ C_2]\!]\,(f)\,(\sigma) &= \mathsf{wp}\,[\![C_1]\!]\,\big(\mathsf{wp}\,[\![C_2]\!]\,(f)\big)\,(\sigma) \\ &= \bigoplus_{\sigma'\in\Sigma} [\![C_1]\!](\sigma,\sigma') \odot \mathsf{wp}\,[\![C_2]\!]\,(f)\,(\sigma') && \text{(by I.H. on } C_1) \\ &= \bigoplus_{\sigma'\in\Sigma} [\![C_1]\!](\sigma,\sigma') \odot \bigoplus_{\tau\in\Sigma} [\![C_2]\!](\sigma',\tau) \odot f(\tau) && \text{(by I.H. on } C_2) \\ &= \bigoplus_{\sigma'\in\Sigma}\bigoplus_{\tau\in\Sigma} [\![C_1]\!](\sigma,\sigma') \odot\ [\![C_2]\!](\sigma',\tau) \odot f(\tau) && \text{(by distributivity of } \odot) \\ &= \bigoplus_{\tau\in\Sigma} \Big(\bigoplus_{\sigma'\in\Sigma} [\![C_1]\!](\sigma,\sigma') \odot [\![C_2]\!](\sigma',\tau)\Big) \odot f(\tau) && \text{(by commutativity of } \oplus) \\ &= \bigoplus_{\tau\in\Sigma} [\![C_1 ⨾ C_2]\!](\sigma,\tau) \odot f(\tau)\ . \end{aligned}$$

The nondeterministic choice $\{\, C_1 \,\} \mathbin{\square} \{\, C_2 \,\}$: We have

$$\begin{aligned} \mathsf{wp}\,[\![\{\, C_1 \,\} \square \{\, C_2 \,\}]\!]\,(f)\,(\sigma) &= \mathsf{wp}\,[\![C_1]\!]\,(f) \oplus \mathsf{wp}\,[\![C_2]\!]\,(f) \\ &= \bigoplus_{\tau\in\Sigma}[\![C_1]\!](\sigma,\tau) \odot f(\tau) \oplus \bigoplus_{\tau\in\Sigma}[\![C_2]\!](\sigma,\tau) \odot f(\tau) && \text{(by I.H. on } C_1, C_2) \\ &= \bigoplus_{\tau\in\Sigma} ([\![C_1]\!](\sigma,\tau) \oplus [\![C_2]\!](\sigma,\tau)) \odot f(\tau) && \text{(by distributivity of } \odot) \\ &= \bigoplus_{\tau\in\Sigma} [\![\{\, C_1 \,\} \square \{\, C_2 \,\}]\!](\sigma,\tau) \odot f(\sigma)\ . \end{aligned}$$

The Iteration $C^{\langle e,e'\rangle}$: Let

$$\Phi_f(X) \;=\; [\![e']\!] \odot f \oplus [\![e]\!] \odot \mathsf{wp}\,[\![C]\!]\,(X)\ ,$$

be the wp-characteristic function of the iteration $C^{\langle e,e'\rangle}$ with respect to any preanticipation f and

$$F(X)(\sigma,\tau) \;=\; \sigma(e) \odot \left(\bigoplus_{\sigma'\in\Sigma}[\![C]\!](\sigma,\sigma') \odot X(\sigma',\tau)\right) \oplus \sigma(e') \odot [\sigma = \tau]\ ,$$

be the denotational semantics characteristic function of the loop $C^{\langle e,e'\rangle}$ for any input $\sigma, \tau \in \Sigma$. We first prove by induction on n that, for all $\sigma \in \Sigma, f \in \mathbb{A}$

$$\Phi_f^n(\mathbb{0})(\sigma) \;=\; \bigoplus_{\tau\in\Sigma} F^n(\mathbb{0})(\sigma,\tau) \odot f(\tau)\ . \tag{3}$$

For the induction base $n = 0$, consider the following:

$$\begin{aligned} \Phi_f^n(\mathbb{0})(\sigma) &= \mathbb{0} \\ &= \bigoplus_{\tau\in\Sigma} F^0(\mathbb{0})(\sigma,\tau) \odot f(\tau)\ . \end{aligned}$$

As induction hypothesis, we have for arbitrary but fixed n and all $\tau \in \Sigma, f \in \mathbb{A}$

$$\Phi_f^n(\mathbb{0})(\sigma) \;=\; \bigoplus_{\tau \in \Sigma} F^n(\mathbb{0})(\sigma, \tau) \odot f(\tau)\;.$$

For the induction step $n \longrightarrow n+1$, consider the following:

$$\begin{aligned}
&\Phi_f^{n+1}(\mathbb{0})(\sigma) \\
&= \Big(\llbracket e' \rrbracket \odot f \oplus \llbracket e \rrbracket \odot \mathsf{wp}\,\llbracket C \rrbracket \Big(\Phi_f^n(\mathbb{0})\Big)\Big)(\sigma) \\
&= \llbracket e' \rrbracket(\sigma) \odot f(\sigma) \oplus \llbracket e \rrbracket(\sigma) \odot \mathsf{wp}\,\llbracket C \rrbracket \Big(\Phi_f^n(\mathbb{0})\Big)(\sigma) \\
&= \sigma(e') \odot f(\sigma) \oplus \sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} \llbracket C \rrbracket(\sigma, \sigma') \odot \Phi_f^n(\mathbb{0})(\sigma') && \text{(by I.H. on } C\text{)} \\
&= \sigma(e') \odot f(\sigma) \oplus \sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} \llbracket C \rrbracket(\sigma, \sigma') \odot \bigoplus_{\tau \in \Sigma} F^n(\mathbb{0})(\sigma', \tau) \odot f(\tau) && \text{(by I.H. on } n\text{)} \\
&= \sigma(e') \odot f(\sigma) \oplus \bigoplus_{\tau \in \Sigma} \big(\sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} \llbracket C \rrbracket(\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau)\big) \odot f(\tau) \\
& && \text{(by distributivity of } \odot\text{, commutativity and associativity of } \oplus\text{)} \\
&= \big(\bigoplus_{\tau \in \Sigma} \sigma(e') \odot [\sigma = \tau] \odot f(\tau)\big) \oplus \bigoplus_{\tau \in \Sigma} \big(\sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} \llbracket C \rrbracket(\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau)\big) \odot f(\tau) \\
&= \bigoplus_{\tau \in \Sigma} \big(\sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} \llbracket C \rrbracket(\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau)\big) \odot f(\tau) \oplus \big(\bigoplus_{\tau \in \Sigma} \sigma(e') \odot [\sigma = \tau] \odot f(\tau)\big) \\
& && \text{(by commutativity of } \oplus\text{)} \\
&= \bigoplus_{\tau \in \Sigma} \big(\sigma(e) \odot \bigoplus_{\sigma' \in \Sigma} \llbracket C \rrbracket(\sigma, \sigma') \odot F^n(\mathbb{0})(\sigma', \tau) \oplus \sigma(e') \odot [\sigma = \tau]\,\big) \odot f(\tau) \\
& && \text{(by associativity of } \oplus \text{ and distributivity of } \odot\text{)} \\
&= \bigoplus_{\tau \in \Sigma} F^{n+1}(\mathbb{0})(\sigma, \tau) \odot f(\tau)\;.
\end{aligned}$$

This concludes the induction on n. Now we have:

$$\begin{aligned}
\mathsf{wp}\,\llbracket C^{\langle e, e' \rangle} \rrbracket\,(f)\,(\sigma) &= \big(\mathsf{lfp}\, X.\; \llbracket e' \rrbracket \odot f \oplus \llbracket e \rrbracket \odot \mathsf{wp}\,\llbracket C \rrbracket\,(X)\,\big)(\sigma) \\
&= \sup_{n \in \mathbb{N}}\; \Phi_f^n(0)(\sigma) && \text{(by Kleene's fixpoint theorem)} \\
&= \sup_{n \in \mathbb{N}} \bigoplus_{\tau \in \Sigma} F^n(\mathbb{0})(\sigma, \tau) \odot f(\tau) && \text{(by Equation 3)} \\
&= \bigoplus_{\tau \in \Sigma} \sup_{n \in \mathbb{N}} F^n(\mathbb{0})(\sigma, \tau) \odot f(\tau) && \text{(by continuity of } \lambda X.\; \textstyle\bigoplus_\tau X(\sigma, \tau) \odot f(\tau)\text{)} \\
&= \bigoplus_{\tau \in \Sigma} \llbracket C^{\langle e, e' \rangle} \rrbracket(\sigma, \tau) \odot f(\tau)\;. && \text{(by Kleene's fixpoint theorem)}
\end{aligned}$$

and this concludes the proof. □

A.3 Proof of sp-wp Duality for probabilistic programs, Thereom 4.5

Theorem 4.5 (Weighted sp-wp Duality). *For all programs C and all functions $\mu, g \in \mathbb{A}$, we have*

$$\bigoplus_{\tau \in \Sigma} \mathsf{sp}\,\llbracket C \rrbracket\,(\mu)\,(\tau) \odot g(\tau) \quad = \quad \bigoplus_{\sigma \in \Sigma} \mu(\sigma) \odot \mathsf{wp}\,\llbracket C \rrbracket\,(g)\,(\sigma)\;.$$

Proof.

$$\begin{aligned}
\bigoplus_{\tau\in\Sigma} \mathsf{sp}\,[\![C]\!]\,(\mu)\,(\tau) \odot g(\tau) &= \bigoplus_{\tau\in\Sigma}\bigoplus_{\sigma\in\Sigma} \mu(\sigma) \odot [\![C]\!](\sigma,\tau) \odot g(\tau) && \text{(by Theorem 4.2)}\\
&= \bigoplus_{\sigma\in\Sigma}\bigoplus_{\tau\in\Sigma} \mu(\sigma) \odot [\![C]\!](\sigma,\tau) \odot g(\tau)\\
&= \bigoplus_{\sigma\in\Sigma} \mu(\sigma) \odot \bigoplus_{\tau\in\Sigma} [\![C]\!](\sigma,\tau) \odot g(\tau)\\
&= \bigoplus_{\sigma\in\Sigma} \mu(\sigma) \odot \mathsf{wp}\,[\![C]\!]\,(g)\,(\sigma)\;. && \text{(by Theorem 4.4)}
\end{aligned}$$

□

B Quantitative Weakest Hyper Pre

Theorem 4.12 (Characterization of whp). *For all programs C, hyperquantities $\mathit{ff} \in \mathbb{AA}$ and quantities $f \in \mathbb{A}$:* $\mathsf{whp}\,[\![C]\!]\,(\mathit{ff})\,(f) = \mathit{ff}(\mathsf{sp}\,[\![C]\!]\,(f))$.

Proof. We prove Theorem 4.12 by induction on the structure of C. For the induction base, we have the atomic statement:

The assignment $x \coloneqq e$: We have

$$\begin{aligned}
\mathsf{whp}\,[\![x \coloneqq e]\!]\,(\mathit{ff})\,(\mu) &= \mathit{ff}(\bigoplus_{\alpha} [x = e\,[x/\alpha]] \odot \mu\,[x/\alpha])\\
&= \mathit{ff}(\mathsf{sp}\,[\![x \coloneqq e]\!]\,(\mu))\;.
\end{aligned}$$

The nondeterministic assignment $x \coloneqq$ `nondet()`: We have

$$\begin{aligned}
\mathsf{whp}\,[\![x \coloneqq \mathtt{nondet()}]\!]\,(\mathit{ff})\,(\mu) &= \mathit{ff}(\bigoplus_{\alpha} \mu\,[x/\alpha])\\
&= \mathit{ff}(\mathsf{sp}\,[\![x \coloneqq \mathtt{nondet()}]\!]\,(\mu))\;.
\end{aligned}$$

The weighting $\odot\, w$: We have

$$\begin{aligned}
\mathsf{whp}\,[\![\odot\, w]\!]\,(\mathit{ff})\,(\mu) &= (\mathit{ff} \odot w)(\mu)\\
&= \mathit{ff}(\mu \odot w)\\
&= \mathit{ff}(\mathsf{sp}\,[\![\odot\, w]\!]\,(\mu))\;.
\end{aligned}$$

This concludes the proof for the atomic statements.

Induction Hypothesis: For arbitrary but fixed programs C, C_1, C_2, we proceed with the inductive step on the composite statements.

The sequential composition $C_1 \fatsemi C_2$: We have

$$\begin{aligned}
\mathsf{whp}\,[\![C_1 \fatsemi C_2]\!]\,(\mathit{ff})\,(\mu) &= \mathsf{whp}\,[\![C_1]\!]\,\big(\mathsf{whp}\,[\![C_2]\!]\,(\mathit{ff})\big)\,(\mu)\\
&= \mathsf{whp}\,[\![C_2]\!]\,(\mathit{ff})\,(\mathsf{sp}\,[\![C_1]\!]\,(\mu)) && \text{(by I.H. on } C_1)\\
&= \mathit{ff}(\mathsf{sp}\,[\![C_2]\!]\,\big(\mathsf{sp}\,[\![C_1]\!]\,(\mu)\big)) && \text{(by I.H. on } C_2)\\
&= \mathit{ff}(\mathsf{sp}\,[\![C_1 \fatsemi C_2]\!]\,(\mu))
\end{aligned}$$

The nondeterministic choice $\{\, C_1 \,\} \mathbin{\Box} \{\, C_2 \,\}$: We have

$$\mathsf{whp}\,[\![\{\, C_1 \,\} \mathbin{\Box} \{\, C_2 \,\}]\!]\,(\mathit{ff})\,(\mu) = \bigoplus_{\nu_1,\nu_2} \mathit{ff}(\nu_1 \oplus \nu_2) \odot \mathsf{whp}\,[\![C_1]\!]\,([\nu_1])\,(\mu) \odot \mathsf{whp}\,[\![C_2]\!]\,([\nu_2])\,(\mu)$$

$$= \bigoplus_{\nu_1,\nu_2} \mathit{ff}(\nu_1 \oplus \nu_2) \odot [\nu_1]\,(\mathrm{sp}\,[\![C_1]\!]\,(\mu)) \odot [\nu_2]\,(\mathrm{sp}\,[\![C_2]\!]\,(\mu))$$

(by I.H. on C_1, C_2)

$$= \mathit{ff}(\mathrm{sp}\,[\![C_1]\!]\,(\mu) \oplus \mathrm{sp}\,[\![C_2]\!]\,(\mu))$$
$$= \mathit{ff}(\mathrm{sp}\,[\![\{\, C_1 \,\} \mathbin{\square} \{\, C_2 \,\}]\!]\,(\mu))\ .$$

The Iteration $C^{\langle e,e'\rangle}$*:*

$$\mathrm{whp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mathit{ff})\,(\mu) = \mathit{ff}\big(\big(\mathrm{lfp}\,X\boldsymbol{.}\ \mu \oplus \mathrm{sp}\,[\![C]\!]\,\big(X \odot [\![e]\!]\big)\,\big) \odot [\![e']\!]\big)$$
$$= \mathit{ff}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu))\ .$$

and this concludes the proof. □

B.1 Proof of Consistency of iteration rule, Theorem 4.5

Proposition 4.10 (Consistency of iteration rule). *Let*

$$\Phi(\mathit{trnsf}) = \lambda \mathit{hh}\, \lambda f\boldsymbol{.} \bigoplus_{\nu} \mathit{hh}(\nu \oplus f \odot [\![e']\!]) \odot \mathrm{whp}\,[\![C]\!]\,(\mathit{trnsf}([\nu]))\,(f \odot [\![e]\!])$$

Then, $\mathrm{whp}[\![C^{\langle e,e'\rangle}]\!]$ *is a fixpoint of the higher order function* $\Phi(\mathit{trnsf})$*, that is:*

$$\Phi(\lambda \mathit{ff}\, \lambda\mu\boldsymbol{.}\ \mathit{ff}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu))) = \lambda \mathit{ff}\, \lambda\mu\boldsymbol{.}\ \mathit{ff}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu))$$

Proof.

$$\Phi(\lambda \mathit{ff}\, \lambda\mu\boldsymbol{.}\ \mathit{ff}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu)))$$
$$= \lambda \mathit{hh}\, \lambda f\boldsymbol{.} \bigoplus_{\nu} \mathit{hh}(\nu \oplus f \odot [\![e']\!]) \odot \mathrm{whp}\,[\![C]\!]\,\Big(\lambda\mu\boldsymbol{.}\ [\nu]\,(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu))\Big)\,(f \odot [\![e]\!])$$
$$= \lambda \mathit{hh}\, \lambda f\boldsymbol{.} \bigoplus_{\nu} \mathit{hh}(\nu \oplus f \odot [\![e']\!]) \odot [\nu]\,(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,\big(\mathrm{sp}\,[\![C]\!]\,\big(f \odot [\![e]\!]\big)\big)) \qquad \text{(by I.H. on } C\text{)}$$
$$= \lambda \mathit{hh}\, \lambda f\boldsymbol{.}\ \mathit{hh}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,\big(\mathrm{sp}\,[\![C]\!]\,\big(f \odot [\![e]\!]\big)\big) \oplus f \odot [\![e']\!])$$
$$= \lambda \mathit{hh}\, \lambda f\boldsymbol{.}\ \mathit{hh}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(f))$$

($\mathrm{sp}[\![C^{\langle e,e'\rangle}]\!]$ is a fixpoint of $\Psi(X) = \lambda f\boldsymbol{.}\ X(\mathrm{sp}\,[\![C]\!]\,\big(f \odot [\![e]\!]\big) \oplus f \odot [\![e']\!])$)

$$= \lambda \mathit{ff}\, \lambda\mu\boldsymbol{.}\ \mathit{ff}(\mathrm{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu))\ .$$

□

B.2 Properties

Theorem 5.1 (Subsumption of HHL). *For hyperpredicates* ψ*,* ϕ *and non-probabilistic program* C*:*

$$\models_{\mathrm{hh}} \{\, \psi \,\}\ C\ \{\, \phi \,\} \quad \text{iff} \quad \mathrm{supp}\,([\psi]) \subseteq \mathrm{supp}\,\big(\mathrm{whp}\,[\![C]\!]\,([\phi])\big)$$

Proof.

$$\begin{aligned}
\models_{\mathrm{hh}} \{\, \psi \,\}\ C\ \{\, \phi \,\} \quad & \text{iff} \quad \forall S \in \mathcal{P}(\Sigma)\boldsymbol{.}\ S \in \psi \implies [\![C]\!](S) \in \phi \\
& \text{iff} \quad \forall S \in \mathcal{P}(\Sigma)\boldsymbol{.}\ S \in \psi \implies \mathrm{supp}\,\big(\mathrm{sp}\,[\![C]\!]\,([S])\big) \in \phi \\
& \text{iff} \quad \forall S \in \mathcal{P}(\Sigma)\boldsymbol{.}\ [\psi]\,([S]) \leq [\phi]\,(\mathrm{sp}\,[\![C]\!]\,([S])) \\
& \text{iff} \quad \forall S \in \mathcal{P}(\Sigma)\boldsymbol{.}\ [\psi]\,([S]) \leq \mathrm{whp}\,[\![C]\!]\,([\phi])\,([S]) \\
& \text{iff} \quad \forall \mu \in \mathbb{A}\boldsymbol{.}\ [\psi]\,(\mu) \leq \mathrm{whp}\,[\![C]\!]\,([\phi])\,(\mu) \\
& \text{iff} \quad \mathrm{supp}\,([\psi]) \implies \mathrm{supp}\,\big(\mathrm{whp}\,[\![C]\!]\,([\phi])\big)
\end{aligned}$$

□

Theorem 5.2 (Falsifying correctness triples via correctness triples).

$$\begin{aligned}
\models_{\mathrm{pc}} \{\,P\,\}\, C\, \{\,Q\,\} \quad &\text{iff} \quad \forall \sigma \in P\textbf{.}\ \not\models_{\mathrm{atc}} \{\,\{\sigma\}\,\}\, C\, \{\,\neg Q\,\} \\
\models_{\mathrm{atc}} \{\,P\,\}\, C\, \{\,Q\,\} \quad &\text{iff} \quad \forall \sigma \in P\textbf{.}\ \not\models_{\mathrm{pc}} \{\,\{\sigma\}\,\}\, C\, \{\,\neg Q\,\} \\
\models_{\mathrm{pi}} [\,P\,]\, C\, [\,Q\,] \quad &\text{iff} \quad \forall \sigma \in Q\textbf{.}\ \not\models_{\mathrm{ti}} [\,\neg P\,]\, C\, [\,\{\sigma\}\,] \\
\models_{\mathrm{ti}} [\,P\,]\, C\, [\,Q\,] \quad &\text{iff} \quad \forall \sigma \in Q\textbf{.}\ \not\models_{\mathrm{pi}} [\,\neg P\,]\, C\, [\,\{\sigma\}\,]
\end{aligned}$$

Proof. First, let us observe that

$$A \subseteq B \quad \text{iff} \quad \forall x \in A\textbf{.}\ \{x\} \cap B \neq \emptyset$$

Now, we have:

(1)

$$\begin{aligned}
\models_{\mathrm{pc}} \{\,P\,\}\, C\, \{\,Q\,\} \quad &\text{iff} \quad P \subseteq \mathsf{wlp}[\![C]\!]\,(Q) \\
&\text{iff} \quad \forall \sigma \in P\textbf{.}\ \{\sigma\} \cap \mathsf{wlp}[\![C]\!]\,(Q) \neq \emptyset \\
&\text{iff} \quad \forall \sigma \in P\textbf{.}\ \not\models_{\mathrm{atc}} \{\,\{\sigma\}\,\}\, C\, \{\,\neg Q\,\}
\end{aligned}$$

(2)

$$\begin{aligned}
\models_{\mathrm{atc}} \{\,P\,\}\, C\, \{\,Q\,\} \quad &\text{iff} \quad P \subseteq \mathsf{wp}\,[\![C]\!]\,(Q) \\
&\text{iff} \quad \forall \sigma \in P\textbf{.}\ \{\sigma\} \cap \mathsf{wp}\,[\![C]\!]\,(Q) \neq \emptyset \\
&\text{iff} \quad \forall \sigma \in P\textbf{.}\ \not\models_{\mathrm{pc}} \{\,\{\sigma\}\,\}\, C\, \{\,\neg Q\,\}
\end{aligned}$$

(3)

$$\begin{aligned}
\models_{\mathrm{pi}} [\,P\,]\, C\, [\,Q\,] \quad &\text{iff} \quad Q \subseteq \mathsf{slp}[\![C]\!]\,(P) \\
&\text{iff} \quad \forall \sigma \in Q\textbf{.}\ \{\sigma\} \cap \mathsf{slp}[\![C]\!]\,(P) \neq \emptyset \\
&\text{iff} \quad \forall \sigma \in Q \not\models_{\mathrm{ti}} [\,\neg P\,]\, C\, [\,\{\sigma\}\,]
\end{aligned}$$

(4)

$$\begin{aligned}
\models_{\mathrm{ti}} [\,P\,]\, C\, [\,Q\,] \quad &\text{iff} \quad Q \subseteq \mathsf{sp}\,[\![C]\!]\,(P) \\
&\text{iff} \quad \forall \sigma \in Q\textbf{.}\ \{\sigma\} \cap \mathsf{sp}\,[\![C]\!]\,(P) \neq \emptyset \\
&\text{iff} \quad \forall \sigma \in Q \not\models_{\mathrm{pi}} [\,\neg P\,]\, C\, [\,\{\sigma\}\,]
\end{aligned}$$

□

Theorem 5.7 (Subsumption of Quantitative wp, wlp for Nondeterministic Programs [Zhang and Kaminski 2022]). *Let* $\mathcal{A} = \langle \mathbb{R}^{\pm\infty}, \max, \min, \mathbb{0}, \mathbb{1} \rangle$*. For any quantities* g, f *and any program* C *satisfying the syntax of* [Zhang and Kaminski 2022, Section 2]*:*

$$\mathsf{whp}\,[\![C]\!]\left(\curlywedge [f]\right)(\mathbf{1}_\sigma) \;=\; \mathsf{wlp}[\![C]\!]\,(f)\,(\sigma) \qquad \text{and} \qquad \mathsf{whp}\,[\![C]\!]\left(\curlyvee [f]\right)(\mathbf{1}_\sigma) \;=\; \mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma)$$

Proof.

$$\begin{aligned}
\mathsf{whp}\,[\![C]\!]\left(\curlyvee [f]\right)(\mathbf{1}_\sigma) \;&=\; \curlyvee [f](\mathsf{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)) \\
&=\; \curlyvee_{\tau\colon\, \mathsf{sp}\,[\![C]\!](\mathbf{1}_\sigma)(\tau) > 0} f(\tau) \\
&=\; \mathsf{wp}\,[\![C]\!]\,(f)\,(\sigma) \\
\mathsf{whp}\,[\![C]\!]\left(\curlywedge [f]\right)(\mathbf{1}_\sigma) \;&=\; \curlywedge [f](\mathsf{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)) \\
&=\; \curlywedge_{\tau\colon\, \mathsf{sp}\,[\![C]\!](\mathbf{1}_\sigma)(\tau) > 0} f(\tau)
\end{aligned}$$

$$= \text{wlp}[\![C]\!]\,(f)\,(\sigma)$$

□

Theorem 5.8 (Subsumption of Quantitative wp, wlp for probabilistic programs [Kaminski 2019]). *Let Prob* $= \langle [0,1], +, \cdot, 0, 1\rangle$. *For any quantities* g, f *and any* non-non*deterministic program* C*:*

$$\text{whp}\,[\![C]\!]\,(\mathbb{E}[f])\,(\mathbf{1}_\sigma) = \text{wp}\,[\![C]\!]\,(f)\,(\sigma) \quad \textit{and} \quad \text{whp}\,[\![C]\!]\,(\mathbb{E}[f] + 1 - \mathbb{E}[1])\,(\mathbf{1}_\sigma) = \text{wlp}[\![C]\!]\,(f)\,(\sigma).$$

Proof.

$$\begin{aligned}\text{whp}\,[\![C]\!]\,(\mathbb{E}[f])\,(\mathbf{1}_\sigma) &= \mathbb{E}[f](\text{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)) \\ &= \text{wp}\,[\![C]\!]\,(f)\,(\sigma)\end{aligned}$$

$$\begin{aligned}\text{whp}\,[\![C]\!]\,(\mathbb{E}[f] + 1 - \mathbb{E}[1])\,(\mathbf{1}_\sigma) &= (\mathbb{E}[f] + 1 - \mathbb{E}[1])(\text{sp}\,[\![C]\!]\,(\mathbf{1}_\sigma)) \\ &= \text{wp}\,[\![C]\!]\,(f)\,(\sigma) + 1 - \text{wp}\,[\![C]\!]\,(1)\,(\sigma) \\ &= \text{wlp}[\![C]\!]\,(f)\,(\sigma)\end{aligned}$$

□

C Proofs of Section 6

C.1 Proof of Healthiness Properties of Quantitative Transformers, Theorem 6.1

Each of the properties is proven individually below.

- Quantitative universal conjunctiveness: Theorem C.1;
- Quantitative universal disjunctiveness: Theorem C.2;
- Strictness: Corollary C.3;
- Costrictness: Corollary C.4;
- Monotonicity: Corollary C.5.

Theorem C.1 (Quantitative universal conjunctiveness of whp). *For any set of quantities* $S \subseteq \mathbb{AA}$,

$$\text{whp}\,[\![C]\!]\left(\prod S\right) = \prod_{\boldsymbol{f} \in S} \text{whp}\,[\![C]\!]\,(\boldsymbol{f\!f}) \;.$$

Proof.

$$\begin{aligned}\text{whp}\,[\![C]\!]\left(\prod S\right) &= \lambda\mu\textbf{.}\,(\prod S)(\text{sp}\,[\![C]\!]\,(\mu)) && \text{(by Theorem 4.12)} \\ &= \lambda\mu\textbf{.}\,\prod_{\boldsymbol{f} \in S} \boldsymbol{f\!f}(\text{sp}\,[\![C]\!]\,(\mu)) \\ &= \prod_{\boldsymbol{f} \in S} \text{whp}\,[\![C]\!]\,(\boldsymbol{f\!f}) \;. && \text{(by Theorem 4.12)}\end{aligned}$$

□

Theorem C.2 (Quantitative universal disjunctiveness of whp). *For any set of quantities* $S \subseteq \mathbb{AA}$,

$$\text{whp}\,[\![C]\!]\left(\sum S\right) = \sum_{\boldsymbol{f} \in S} \text{whp}\,[\![C]\!]\,(\boldsymbol{f\!f}) \;.$$

Proof.

$$\text{whp}\,[\![C]\!]\left(\sum S\right) = \lambda\mu\textbf{.}\,(\sum S)(\text{sp}\,[\![C]\!]\,(\mu)) \qquad \text{(by Theorem 4.12)}$$

$$
\begin{aligned}
&= \lambda\mu .\ \sum_{\boldsymbol{f} \in S} ff(\mathrm{sp}\ [\![C]\!]\ (\mu)) \\
&= \sum_{\boldsymbol{f} \in S} \mathrm{whp}\ [\![C]\!]\ (ff)\ . && \text{(by Theorem 4.12)}
\end{aligned}
$$

□

Corollary C.3 (Strictness of whp). *For all programs C,* whp$[\![C]\!]$ *is strict, i.e.*

$$\mathrm{whp}\ [\![C]\!]\ (0) \ = \ 0\ .$$

Proof.

$$
\begin{aligned}
\mathrm{whp}\ [\![C]\!]\ (0) &= \lambda\mu .\ (0)(\mathrm{sp}\ [\![C]\!]\ (\mu)) && \text{(by Theorem 4.12)} \\
&= 0\ .
\end{aligned}
$$

□

Corollary C.4 (Co-strictness of whp). *For all programs C,* wp$[\![C]\!]$ *is co-strict, i.e.*

$$\mathrm{whp}\ [\![C]\!]\ (+\infty) \ = \ +\infty\ .$$

Proof.

$$
\begin{aligned}
\mathrm{whp}\ [\![C]\!]\ (+\infty) &= \lambda\mu .\ (+\infty)(\mathrm{sp}\ [\![C]\!]\ (\mu)) && \text{(by Theorem 4.12)} \\
&= +\infty\ .
\end{aligned}
$$

□

Corollary C.5 (Monotonicity of Quantitative Transformers). *For all programs C, $ff, gg \in$, we have*

$$ff \ \preceq \ gg \qquad \text{implies} \qquad \mathrm{whp}\ [\![C]\!]\ (ff) \ \preceq \ \mathrm{whp}\ [\![C]\!]\ (gg)$$

Proof.

$$
\begin{aligned}
\mathrm{whp}\ [\![C]\!]\ (ff) &= \lambda\mu .\ ff(\mathrm{sp}\ [\![C]\!]\ (\mu)) && \text{(by Theorem 4.12)} \\
&\preceq \lambda\mu .\ gg(\mathrm{sp}\ [\![C]\!]\ (\mu)) && (ff \ \preceq \ gg) \\
&= \mathrm{whp}\ [\![C]\!]\ (gg) && \text{(by Theorem 4.12)}
\end{aligned}
$$

□

C.2 Proof of Linearity, Theorem 6.2

Theorem 6.2 (Linearity). *For all programs C,* whp$[\![C]\!]$ *is linear, i.e. for all $ff, gg \in \mathbb{AA}$ and* non-negative *constants* $r \in \mathbb{R}_{\geq 0}$, $\mathrm{whp}\ [\![C]\!]\ (r \cdot ff + gg) = r \cdot \mathrm{whp}\ [\![C]\!]\ (ff) + \mathrm{whp}\ [\![C]\!]\ (gg)$.

Proof.

$$
\begin{aligned}
&\mathrm{whp}\ [\![C]\!]\ (r \cdot ff + gg) \\
&= \lambda\mu .\ (r \cdot ff + gg)(\mathrm{sp}\ [\![C]\!]\ (\mu)) && \text{(by Theorem 4.12)} \\
&= \lambda\mu .\ (r \cdot ff)(\mathrm{sp}\ [\![C]\!]\ (\mu)) + gg(\mathrm{sp}\ [\![C]\!]\ (\mu)) \\
&= \lambda\mu .\ r \cdot ff(\mathrm{sp}\ [\![C]\!]\ (\mu)) + gg(\mathrm{sp}\ [\![C]\!]\ (\mu)) \\
&= r \cdot \mathrm{whp}\ [\![C]\!]\ (ff) + \mathrm{whp}\ [\![C]\!]\ (gg)\ . && \text{(by Theorem 4.12)}
\end{aligned}
$$

□

C.3 Proof of Multiplicativity, Theorem 6.3

THEOREM 6.3 (MULTIPLICATIVITY). *For all programs C,* whp$[\![C]\!]$ *is multiplicative, i.e. for all $ff, gg \in \mathbb{AA}$ and* non-negative *constants $r \in \mathbb{R}_{\geq 0}$,* $\mathsf{whp}\,[\![C]\!]\,(r \cdot ff \cdot gg) = r \cdot \mathsf{whp}\,[\![C]\!]\,(ff) \cdot \mathsf{whp}\,[\![C]\!]\,(gg)$.

PROOF.

$$
\begin{aligned}
&\mathsf{whp}\,[\![C]\!]\,(r \cdot ff \cdot gg) \\
&= \lambda\mu.\, (r \cdot ff \cdot gg)(\mathsf{sp}\,[\![C]\!]\,(\mu)) && \text{(by Theorem 4.12)} \\
&= \lambda\mu.\, r \cdot ff(\mathsf{sp}\,[\![C]\!]\,(\mu)) \cdot gg(\mathsf{sp}\,[\![C]\!]\,(\mu)) \\
&= r \cdot \mathsf{whp}\,[\![C]\!]\,(ff) \cdot \mathsf{whp}\,[\![C]\!]\,(gg)\ . && \text{(by Theorem 4.12)}
\end{aligned}
$$

□

C.4 Proof of Liberal-Non-liberal Duality, Theorem 6.4

THEOREM 6.4 (LIBERAL–NON-LIBERAL DUALITY). *For any program C and any $k-$bounded hyperquantity ff, we have* $\mathsf{whp}\,[\![C]\!]\,(ff) \;=\; k - \mathsf{whp}\,[\![C]\!]\,(k - ff)$.

PROOF.

$$
\begin{aligned}
\mathsf{whp}\,[\![C]\!]\,(ff) &= \lambda\mu.\, ff(\mathsf{sp}\,[\![C]\!]\,(\mu)) f(\tau) && \text{(by Theorem 4.12)} \\
&= k - \lambda\mu.\, k - ff(\mathsf{sp}\,[\![C]\!]\,(\mu)) \\
&= k - \mathsf{whp}\,[\![C]\!]\,(k - ff)\ .
\end{aligned}
$$

□

Proof of rules for linear hyperquantities, Theorem 6.6

THEOREM 6.6 (WEAKEST HYPER PRE FOR LINEAR HYPERQUANTITIES). *For linear hyperquantities $ff \in \mathbb{AA}$, the simpler rules in* Table 7 *are valid.*

PROOF. We prove Theorem 4.12 by induction on the structure of C. For the induction base, we have the atomic statement:

The assignment $x \coloneqq e$: We have

$$
\begin{aligned}
\mathsf{whp}\,[\![x \coloneqq e]\!]\,(ff)\,(\mu) &= \bigoplus_{\alpha} ff([x = e\,[x/\alpha]] \odot \mu\,[x/\alpha]) \\
&= ff(\bigoplus_{\alpha} [x = e\,[x/\alpha]] \odot \mu\,[x/\alpha]) \\
&= ff(\mathsf{sp}\,[\![x \coloneqq e]\!]\,(\mu))\ .
\end{aligned}
$$

The nondeterministic assignment $x \coloneqq$ `nondet()`*:* We have

$$
\begin{aligned}
\mathsf{whp}\,[\![x \coloneqq \mathtt{nondet()}]\!]\,(ff)\,(\mu) &= ff(\bigoplus_{\alpha} \mu\,[x/\alpha]) \\
&= ff(\mathsf{sp}\,[\![x \coloneqq \mathtt{nondet()}]\!]\,(\mu))\ .
\end{aligned}
$$

The weighting $\odot\, a$: We have

$$
\begin{aligned}
\mathsf{whp}\,[\![\odot\, w]\!]\,(ff)\,(\mu) &= (ff \odot w)(\mu) \\
&= ff(\mu \odot w) \\
&= ff(\mathsf{sp}\,[\![\odot\, w]\!]\,(\mu))\ .
\end{aligned}
$$

This concludes the proof for the atomic statements.

Induction Hypothesis: For arbitrary but fixed programs C, C_1, C_2, we proceed with the inductive step on the composite statements.

The sequential composition $C_1 \fatsemi C_2$: We have

$$\begin{aligned}
\mathsf{whp}\,[\![C_1 \fatsemi C_2]\!]\,(ff)\,(\mu) &= \mathsf{whp}\,[\![C_1]\!]\,\big(\mathsf{whp}\,[\![C_2]\!]\,(ff)\big)\,(\mu) \\
&= \mathsf{whp}\,[\![C_2]\!]\,(ff)\,(\mathsf{sp}\,[\![C_1]\!]\,(\mu)) && \text{(by I.H. on } C_1) \\
&= ff(\mathsf{sp}\,[\![C_2]\!]\,\big(\mathsf{sp}\,[\![C_1]\!]\,(\mu)\big)) && \text{(by I.H. on } C_2) \\
&= ff(\mathsf{sp}\,[\![C_1 \fatsemi C_2]\!]\,(\mu))
\end{aligned}$$

The nondeterministic choice $\{ C_1 \} \square \{ C_2 \}$: We have

$$\begin{aligned}
\mathsf{whp}\,[\![\{ C_1 \} \square \{ C_2 \}]\!]\,(ff)\,(\mu) &= \mathsf{whp}\,[\![C_1]\!]\,(ff)\,(\mu) \oplus \mathsf{whp}\,[\![C_2]\!]\,(ff)\,(\mu) \\
&= ff(\mathsf{sp}\,[\![C_1]\!]\,(\mu)) \oplus ff(\mathsf{sp}\,[\![C_2]\!]\,(\mu)) && \text{(by I.H. on } C_1, C_2) \\
&= ff(\mathsf{sp}\,[\![C_1]\!]\,(\mu) \oplus \mathsf{sp}\,[\![C_2]\!]\,(\mu)) && \text{(by Definition 6.5)} \\
&= \bigoplus_{\nu_1,\nu_2} ff(\nu_1 \oplus \nu_2) \odot [\nu_1]\,(\mathsf{sp}\,[\![C_1]\!]\,(\mu)) \odot [\nu_2]\,(\mathsf{sp}\,[\![C_2]\!]\,(\mu)) \\
&= ff(\mathsf{sp}\,[\![C_1]\!]\,(\mu) \oplus \mathsf{sp}\,[\![C_2]\!]\,(\mu)) \\
&= ff(\mathsf{sp}\,[\![\{ C_1 \} \square \{ C_2 \}]\!]\,(\mu))\,.
\end{aligned}$$

The Iteration $C^{\langle e,e' \rangle}$: Let $W_e(X) = \mathsf{whp}\,[\![C]\!]\,(X) \odot [\![e]\!]$ and $S(X) = \mathsf{sp}\,[\![C]\!]\,\big(X \odot [\![e]\!]\big)$. We first prove by induction on n that:

$$W_e^n(ff \odot [\![e']\!])(\mu) \;=\; ff(S^n(\mu) \odot [\![e']\!])$$

For the induction base $n = 0$, consider the following:

$$\begin{aligned}
W_e^n(ff \odot [\![e']\!])(\mu) &= (ff \odot [\![e']\!])(\mu) \\
&= ff(\mu \odot [\![e']\!]) \\
&= ff(S^n(\mu) \odot [\![e']\!])\,.
\end{aligned}$$

As induction hypothesis, we have for arbitrary but fixed n and all μ

$$W_e^n(ff \odot [\![e']\!])(\mu) \;=\; ff(S^n(\mu) \odot [\![e']\!])$$

For the induction step $n \longrightarrow n + 1$, consider the following:

$$\begin{aligned}
W_e^{n+1}(ff \odot [\![e']\!])(\mu) &= \big(W_e(W_e^n(ff \odot [\![e']\!]))\big)(\mu) \\
&= \big(\mathsf{whp}\,[\![C]\!]\,\big(W^n(ff \odot [\![e']\!])\big) \odot [\![e]\!]\big)(\mu) \\
&= \big(\mathsf{whp}\,[\![C]\!]\,\big(W^n(ff \odot [\![e']\!])\big)\,\big)(\mu \odot [\![e]\!]) \\
&= W_e^n(ff \odot [\![e']\!])(\mathsf{sp}\,[\![C]\!]\,\big(\mu \odot [\![e]\!]\big)) && \text{(by I.H. on } C) \\
&= ff(S^n(\mathsf{sp}\,[\![C]\!]\,\big(\mu \odot [\![e]\!]\big)) \odot [\![e']\!]) && \text{(by I.H. on } n) \\
&= ff(S^n(S(\mu)) \odot [\![e']\!]) \\
&= ff(S^{n+1}(\mu) \odot [\![e']\!])
\end{aligned}$$

This concludes the induction on n. Now we have:

$$\begin{aligned}
\mathsf{whp}\,[\![C^{\langle e,e' \rangle}]\!]\,(ff)\,(\mu) &= \bigoplus_{n \in \mathbb{N}} W_e^n(ff \odot [\![e']\!])(\mu) \\
&= \bigoplus_{n \in \mathbb{N}} ff(S^n(\mu) \odot [\![e']\!])
\end{aligned}$$

$$= ff\Big(\big(\bigoplus_{n\in\mathbb{N}} S^n(\mu)\big) \odot [\![e']\!]\Big) \qquad \text{(by Definition 6.5)}$$
$$= ff\Big(\big(\mathsf{lfp}\ X.\ \mu \oplus \mathsf{sp}\,[\![C]\!]\,(X \odot [\![e]\!])\,\big) \odot [\![e']\!]\Big)$$
$$= ff(\mathsf{sp}\,[\![C^{\langle e,e'\rangle}]\!]\,(\mu))\ .$$

□

Proof of Quantitative Inductive Reasoning for whp, Theorem 6.9

Theorem 6.9 (Quantitative Inductive Reasoning for whp). *For any program C and any linear hyperquantity ff, we have:*

$$\Phi_{ff}(ii) \preceq ii \implies \mathsf{whp}\,[\![C^{\langle e,e'\rangle}]\!]\,(ff) \preceq ii,$$

where $\Phi_{ff}(X) = ff \odot [\![e']\!] \oplus \mathsf{whp}\,[\![C]\!]\,(X) \odot [\![e]\!]$ is the characteristic function of $C^{\langle e,e'\rangle}$ w.r.t. ff.

Proof.

$$\Phi_{ff}(ii) \ \preceq\ ii \qquad \text{(Premise of the implication)}$$
$$\implies \mathsf{lfp}\ X.\ \Phi_{ff}(X) \ \preceq\ ii$$
(by Park's Induction [Park 1969], since Φ_{ff} is continuous ($\oplus, \odot$ and whp are continuous))
$$\implies \mathsf{whp}\,[\![C^{\langle e,e'\rangle}]\!]\,(ff) \ \preceq\ ii \qquad (ff \text{ is linear})$$

□

Proof of Quantitative Inductive Rule for `while`, Corollary 6.10

Corollary 6.10 (Quantitative Inductive Rule for `while`).

$$\frac{ff \odot [\neg\varphi] \oplus \mathsf{whp}\,[\![C]\!]\,(ii) \odot [\varphi] \ \preceq\ ii \ \preceq\ gg \qquad ff \text{ is linear}}{\mathsf{whp}\,[\![\mathtt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(ff) \ \preceq\ gg}\ \text{while}-\text{whp}$$

Proof.

$$ff \odot [\neg\varphi] \oplus \mathsf{whp}\,[\![C]\!]\,(ii) \odot [\varphi] \ \preceq\ ii \qquad \text{(Premise of the rule)}$$
$$\implies \mathsf{lfp}\ X.\ ff \odot [\neg\varphi] \oplus \mathsf{whp}\,[\![C]\!]\,(X) \odot [\varphi] \ \preceq\ ii \qquad \text{(by Park's Induction [Park 1969])}$$
$$\implies \mathsf{whp}\,[\![\mathtt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(ff) \ \preceq\ ii$$
$$\implies \mathsf{whp}\,[\![\mathtt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!]\,(ff) \ \preceq\ gg \qquad (ii \ \preceq\ gg \text{ and transitivity of } \preceq\,)$$

□

D Well-definedness of the semantics

In this section we prove that the denotational semantics of Section 3 is a total function.

D.1 Additional definitions omitted from the main text

We assume that the operations $\oplus$, $\odot$ belong to a complete, Scott continuous, naturally ordered, partial semiring with a top element.

Definition D.1 (Complete semirings [Golan 2003]). A (partial) semiring $\langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$ is complete if there is a sum operator $\bigoplus_{i\in I}$ with the following properties:

(1) If $I = \{i_1, \dots, i_n\}$ is finite, then $\bigoplus_{i\in I} u_i = u_{i_1} + \cdots + u_{i_n}$.
(2) If $\bigoplus_{i\in I} x_i$ is defined, then $v \odot \bigoplus_{i\in I} u_i = \bigoplus_{i\in I} v \odot u_i$ and $(\bigoplus_{i\in I} u_i) \odot v = \bigoplus_{i\in I} u_i \odot v$.
(3) Let $(J_k)_{k\in K}$ be a family of nonempty disjoint subsets of I ($I = \bigcup_{k\in K} J_k$ and $J_k \cap J_l = \emptyset$ if $k \neq l$), then $\bigoplus_{k\in K} \bigoplus_{j\in J_k} u_j = \bigoplus_{i\in I} u_i$.

Definition D.2 (Scott Continuity [Karner 2004]). A (partial) semiring with order $\leq$ is Scott Continuous if for any directed set $D \subseteq X$ (where all pairs of elements in D have a supremum), the following hold:

$$\sup_{x \in D}(x \oplus y) = (\sup D) \oplus y$$
$$\sup_{x \in D}(x \odot y) = (\sup D) \odot y$$
$$\sup_{x \in D}(y \odot x) = y \odot \sup D$$

D.2 Fixed point existence

Proposition D.3. *Let* $\Phi_{C,e,e'}(X)(\sigma,\tau) = [\![e]\!](\sigma) \odot \left(\bigoplus_{\iota:\, [\![C]\!](\sigma,\iota)\neq \mathbb{0}} [\![C]\!](\sigma,\iota) \odot X(\iota,\tau)\right) \;\oplus\; [\![e']\!](\sigma) \odot [\sigma = \tau]$. *If* $\Phi_{C,e,e'}$ *is a total function, the semantics of loops:*

$$[\![C^{\langle e,e'\rangle}]\!](\sigma,\tau) = (\mathsf{lfp}\, X.\, \Phi_{C,e,e'}(X))(\sigma,\tau)$$

is well-defined, i.e., the least fixed point of $\Phi_{C,e,e'}$ *exists.*

Proof. It is sufficient to show that $\Phi_{C,e,e'}$ is Scott-continuous and rely on Kleene's fixpoint theorem to conclude that the fixpoint exists. For all directed sets $D \subseteq (\Sigma \times \Sigma \to W(\Sigma))$ we have:

$$\begin{aligned}
&\sup_{f \in D}\; \Phi_{C,e,e'}(f)(\sigma,\tau)\\
&= \sup_{f\in D} [\![e]\!](\sigma) \odot \Big(\bigoplus_{\iota\in\Sigma} [\![C]\!](\sigma,\iota) \odot f(\iota,\tau)\Big) \;\oplus\; [\![e']\!](\sigma) \odot [\sigma=\tau]\\
&= [\![e]\!](\sigma) \odot \Big(\sup_{f\in D} \bigoplus_{\iota\in\Sigma} [\![C]\!](\sigma,\iota) \odot f(\iota,\tau)\Big) \;\oplus\; [\![e']\!](\sigma) \odot [\sigma=\tau] \qquad \text{(by continuity of } \oplus \text{ and } \odot)\\
&= [\![e]\!](\sigma) \odot \Big(\bigoplus_{\iota\in\Sigma} [\![C]\!](\sigma,\iota) \odot \sup\; D(\iota,\tau)\Big) \;\oplus\; [\![e']\!](\sigma) \odot [\sigma=\tau]\\
&\qquad \text{(by [Zilberstein 2024, Lemma A.4] with } f_\iota(X) = [\![C]\!](\sigma,\iota) \odot X(\iota,\tau) \text{ for } \iota \in \Sigma)\\
&= \Phi_{C,e,e'}(\sup\; D)(\sigma,\tau)
\end{aligned}$$

And hence we conclude by Kleene's fixpoint theorem. □

D.3 Syntactic restrictions for partial semirings

Proposition D.3 ensures the well-definedness of the iteration rule, provided that $\Phi_{C,e,e'}$ is total. In this section, we investigate syntactic constraints to ensure the totality of $\Phi_{C,e,e'}$ (and all other statements). Notably, challenges arise in partial semirings only, where $\oplus$ might be undefined. The constraints and results above are adapted from [Zilberstein 2024, Appendix A.3] to our framework.

Definition D.4 (Compatibility [Zilberstein 2024]). The expressions e_1 and e_2 are compatible in semiring $A = \langle U, \oplus, \odot, \mathbb{0}, \mathbb{1}\rangle$ if $[\![e_1]\!](\sigma) \oplus [\![e_2]\!](\sigma)$ is defined for any $\sigma \in \Sigma$.

Proposition D.5. *If* e_1, e_2 *are compatible and* $[\![C_1]\!], [\![C_2]\!]$ *are total functions, then*

$$[\![\{ \odot e_1 \fatsemi C_1 \} \;\square\; \{ \odot e_2 \fatsemi C_2 \}]\!]$$

is a total function.

Proof.

$$\begin{aligned}
&[\![\{ \odot e_1 \fatsemi C_1 \} \;\square\; \{ \odot e_2 \fatsemi C_2 \}]\!](\sigma)\\
&= [\![\odot e_1 \fatsemi C_1]\!](\sigma,\tau) \;\oplus\; [\![\odot e_2 \fatsemi C_2]\!](\sigma,\tau)
\end{aligned}$$

$$
\begin{aligned}
&= \bigoplus_{\iota:\ [\![\odot\, e_1]\!](\sigma,\iota)\neq \mathbb{0}} [\![\odot\, e_1]\!](\sigma,\iota) \odot [\![C_1]\!](\iota,\tau) \\
&\quad \oplus \bigoplus_{\iota:\ [\![\odot\, e_2]\!](\sigma,\iota)\neq \mathbb{0}} [\![\odot\, e_2]\!](\sigma,\iota) \odot [\![C_2]\!](\iota,\tau) \\
&= \bigoplus_{\iota:\ [\![e_1]\!](\sigma)\odot[\sigma=\iota]\neq \mathbb{0}} [\![e_1]\!](\sigma) \odot [\sigma=\iota] \odot [\![C_1]\!](\iota,\tau) \\
&\quad \oplus \bigoplus_{\iota:\ [\![e_2]\!](\sigma)\odot[\sigma=\iota]\neq \mathbb{0}} [\![e_2]\!](\sigma) \odot [\sigma=\iota] \odot [\![C_2]\!](\iota,\tau) \\
&= [\![e_1]\!](\sigma) \odot [\![C_1]\!](\sigma,\tau) \oplus [\![e_2]\!](\sigma) \odot [\![C_2]\!](\sigma,\tau)
\end{aligned}
$$

which is well-defined by [Zilberstein 2024, Lemma A.5] (since $[\![e_1]\!](\sigma) \oplus [\![e_2]\!](\sigma)$ is well-defined). □

Proposition D.6 (Well-definedness of $C^{\langle e,e'\rangle}$). *If e, e' are compatible and $[\![C]\!]$ is a total function, then $[\![C^{\langle e,e'\rangle}]\!]$ is a total function.*

Proof. Let $\Phi_{C,e,e'}(X)(\sigma,\tau) = [\![e]\!](\sigma) \odot \left(\bigoplus_{\iota\in\Sigma} [\![C]\!](\sigma,\iota) \odot X(\iota,\tau)\right) \oplus [\![e']\!](\sigma) \odot [\sigma=\tau]$. By [Zilberstein 2024, Lemma A.5], $\Phi_{C,e,e'}(X)(\sigma,\tau)$ is well-defined, ensuring the well-definedness of $[\![C^{\langle e,e'\rangle}]\!]$ as well (as per Proposition D.3). □

E Nontermination and Unreachability

However, we can represent these situations using "angelic partial correctness" and "demonic total correctness" triples, respectively.

Triple	Property
$\models_{\text{apc}} \{P\}\ C\ \{\text{false}\}$	May-Nontermination
$\models_{\text{dtc}} \{P\}\ C\ \{\text{true}\}$	Must-Termination
$\not\models_{\text{apc}} \{P\}\ C\ \{\text{false}\}$	Must-Termination
$\not\models_{\text{dtc}} \{P\}\ C\ \{\text{true}\}$	May-Nontermination

Table 9. Nontermination and unreachability.

for a reasonable definition of $[\![C^\star]\!](\sigma)$ may diverge which we omit as this is not the main focus of the paper.

As angelic total correctness triples can be expressed by whp, our calculus also subsume nontermination proving, i.e., the following holds:

$$(\lambda\rho.\ P \cap \rho \neq \emptyset) \subseteq \text{whp}\ [\![C]\!]\ (\lambda\rho.\ P \cap \rho \neq \emptyset) \implies \forall\sigma\in P.\ [\![C^\star]\!](\sigma) \text{ may diverge}$$

Whilst [Raad et al. 2024, Section 1, "Formal Interpretation of Divergent Triples"] focuses on a stronger interpretation of triples where $\models_{\text{atc}} \{P\}\ C\ \{\infty\}$ means *every* state $\sigma \in P$ have *at least* a diverging trace, our framework allows to express *three* novel interpretation as well. We start with the weaker interpretation that mandates the existence of at least one state in the precondition that may diverge.

$$\{P\} \subseteq \text{whp}\ [\![C]\!]\ (\lambda\rho. P \cap \rho \neq \emptyset) \implies \exists\sigma\in P.\ [\![C^\star]\!](\sigma) \text{ may diverge}$$

which can be rewritten as a program logics, using Table 5

$$\frac{\not\models_{\text{pc}} \{P\}\ C\ \{\neg P\}}{\exists\sigma\in P.\ [\![C^\star]\!](\sigma) \text{ may diverge}}$$

It's not surprising that the premise involves the falsification of a triple since the objective is to establish an $\exists$ property. It's worth noting that we can always convert it back to a valid triple in some other logics through Corollary 5.3. However, we choose not to do so, as it would introduce an additional quantifier.

For the remaining two interpretations, we will focus on what we term *must divergence*. Unlike *may divergence*, *must divergence* asserts that all traces originating from a given initial state must diverge. We highlight the inadequacy of $C^\star$ due to its semantics implicitly assuming that divergence should never be necessary. Consequently, our subsequent exploration will revolve around $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$, and we will present rules for all four interpretations.

First all, we show the nontermination rules for $\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}$ via whp.

$$P \subseteq \varphi \quad \text{and} \quad (\lambda\rho.\, P \cap \rho \neq \emptyset) \subseteq \mathsf{whp}\,[\![C]\!]\,(\lambda\rho.\, P \cap \rho \neq \emptyset) \implies \forall\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ may diverge}$$
$$P \subseteq \varphi \quad \text{and} \quad \{P\} \subseteq \mathsf{whp}\,[\![C]\!]\,(\lambda\rho. P \cap \rho \neq \emptyset) \implies \exists\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ may diverge}$$
$$P \subseteq \varphi \quad \text{and} \quad \{P\} \subseteq \mathsf{whp}\,[\![C]\!]\,(\lambda\rho.\, \rho \subseteq P) \implies \forall\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ must diverge}$$
$$P \subseteq \varphi \quad \text{and} \quad \exists\sigma \in P.\, \{\{\sigma\}\} \subseteq \mathsf{whp}\,[\![C]\!]\,(\lambda\rho.\, \rho \subseteq P) \implies \exists\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ must diverge}$$

These can be straightforwardly converted into rules for program logics.

$$\frac{\models_{\text{atc}} \{\,P\,\}\, C\, \{\,P\,\} \qquad P \subseteq \varphi}{\forall\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ may diverge}} \qquad \frac{\not\models_{\text{pc}} \{\,P\,\}\, C\, \{\,\neg P\,\} \qquad P \subseteq \varphi}{\exists\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ may diverge}}$$
$$\frac{\models_{\text{pc}} \{\,P\,\}\, C\, \{\,P\,\} \qquad P \subseteq \varphi}{\forall\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ must diverge}} \qquad \frac{\not\models_{\text{atc}} \{\,P\,\}\, C\, \{\,\neg P\,\} \qquad P \subseteq \varphi}{\exists\sigma \in P.\, [\![\texttt{while}\,(\,\varphi\,)\,\{\,C\,\}]\!](\sigma) \text{ must diverge}}$$

The duality in this context is twofold: moving from left to right, total correctness aligns with the falsification of partial correctness (by Corollary 5.3, essentially capturing the duality between ∀ and ∃). On the other hand, from top to bottom, the duality is determined by the choices made in our interpretation of nondeterminism and bears resemblance to the one highlighted in [Zhang and Kaminski 2022].

As pointed in Table 9, angelic partial correctness and demonic total correctness have a key role in proving may-nontermination and must-termination. It is thus surprising that [Raad et al. 2024] chose to combine (angelic) total correctness and total incorrectness logics for their sound and complete proof system that allows to prove may-nontermination.

In this section, we show how a standard angelic partial correctness proof system relates with the rules in [Raad et al. 2024]. We consider guarded imperative languages with nondeterministic choices (i.e., with while constructs instead of Kleene star), and the rules for angelic partial correctness as analogous to those for standard partial correctness, except for the nondeterministic choice [Kaminski 2019, Definition 4.5]. In particular, it is well known that by coinduction, the following rule holds:

$$\frac{\models_{\text{apc}} \{\,P \land \varphi\,\}\, C\, \{\,P\,\}}{\models_{\text{apc}} \{\,P\,\}\, \texttt{while}\,(\,\varphi\,)\,\{\,C\,\}\, \{\,\neg\varphi \land P\,\}}$$

We shall observe that angelic partial correctness is a complete proof system (for guarded imperative languages), and this already means that every may-nontermination triple can be proved. However, let us show how we can derive simpler rules (analogous to those in [Raad et al. 2024]) without the need to add explicit rules for may-nontermination.

Theorem E.1. *The following rules are valid in angelic partial correctness logic:*

$$\frac{\models_{\text{apc}} \{\,P\,\}\, C_1\, \{\,\mathsf{false}\,\}}{\models_{\text{apc}} \{\,P\,\}\, C_1 \mathbin{⨟} C_2\, \{\,\mathsf{false}\,\}} \qquad \frac{\models_{\text{apc}} \{\,P\,\}\, C_1\, \{\,Q\,\} \qquad \models_{\text{apc}} \{\,Q\,\}\, C_2\, \{\,\mathsf{false}\,\}}{\models_{\text{apc}} \{\,P\,\}\, C_1 \mathbin{⨟} C_2\, \{\,\mathsf{false}\,\}}$$

$$\frac{\models_{\text{apc}} \{\, P \,\}\ C_i\ \{\, \text{false} \,\}\ \textit{for some } i \in \{1,2\}}{\models_{\text{apc}} \{\, P \,\}\ \{\, C_1 \,\} \,\square\, \{\, C_2 \,\}\ \{\, \text{false} \,\}} \qquad \frac{\models_{\text{apc}} \{\, P \wedge \varphi \,\}\ C\ \{\, P \wedge \varphi \,\}}{\models_{\text{apc}} \{\, P \wedge \varphi \,\}\ \texttt{while}\,(\, \varphi \,)\,\{\, C \,\}\ \{\, \text{false} \,\}}$$

The rules above resemble to those in [Raad et al. 2024], but again we stress that here we are not developing a new complex logic. It is also easy to show that the loop rule for while loops in [Raad et al. 2024] can be very easily proved:

$$\cfrac{\cfrac{\models_{\text{atc}} \{\, P \wedge \varphi \,\}\ C\ \{\, P \wedge \varphi \,\}}{\models_{\text{apc}} \{\, P \wedge \varphi \,\}\ C\ \{\, P \wedge \varphi \,\}}}{\models_{\text{apc}} \{\, P \wedge \varphi \,\}\ \texttt{while}\,(\, \varphi \,)\,\{\, C \,\}\ \{\, \text{false} \,\}}$$

E.1 Nontermination and Unreachability

It's worth noting that in all four rules, we are concerned with correctness triples rather than incorrectness ones. This emphasis is due to our focus on the termination of the forward semantics. Analogous rules for partial incorrectness and total incorrectness triples would facilitate the identification of nonterminating states in the backward semantics. For instance, we can establish:

$$\frac{\models_{\text{ti}} [\, P \,]\ C\ [\, P \,]}{\forall \sigma \in P.\ [\![C^\star]\!]^{-1}(\sigma) \text{ may diverge}} \qquad \frac{\not\models_{\text{pi}} [\, \neg P \,]\ C\ [\, P \,]}{\exists \sigma \in P.\ [\![C^\star]\!]^{-1}(\sigma) \text{ may diverge}}$$

The rules can be used in the context of program inversion to assess whether one could compute the pre-image by simply executing the inverted program.

The correlation between nontermination and unreachability, as highlighted in [Zhang and Kaminski 2022], may lead one to question whether proving states as unreachable is related to demonstrating nontermination. However, when considering backward semantics, a single nonterminating trace doesn't provide enough information to establish unreachability. It is essential for all backward traces to be nonterminating, aligning with the concept of must-termination in backward semantics, precisely corresponding to what is conventionally meant by unreachability. This insight strengthens the connection described in [Zhang and Kaminski 2022], where their dualities between nontermination and unreachability arise from the resolution of nondeterministic choices. In other words, when [Zhang and Kaminski 2022] refers to nontermination, they essentially mean must-nontermination.

Backward Must-Nontermination. Again, when reasoning about *must-nontermination* on $C^\star$, it is trivially false for the backward semantics as well. To make it worse, we argue that it is trivial for $\texttt{while}\,(\, \varphi \,)\,\{\, C \,\}$ as well: if our final state $\tau \models \varphi$, then it is clearly unreachable and otherwise it is reachable (in 0 iterations).

F Full calculations and examples omitted from the main text

F.1 Full calculations of Section 7.3.2

To compute $\text{whp}\,[\![x \coloneqq x + 1^{\langle \frac{1}{2}, \frac{1}{2} \rangle}]\!]\,(\mathbb{E}[x^2])$, we compute subsequent Kleene's iterates obtaining:

$$\begin{aligned}
W^0_{0.5}(\mathbb{E}[x^2] \odot 0.5) &= \mathbb{E}[x^2] \odot 0.5 \\
W^1_{0.5}(\mathbb{E}[x^2] \odot 0.5) &= \text{whp}\,[\![x \coloneqq x + 1]\!]\,\big(\mathbb{E}[x^2] \odot 0.5\big) \odot 0.5 \;=\; \mathbb{E}[(x+1)^2] \odot 0.5^2 \\
W^2_{0.5}(\mathbb{E}[x^2] \odot 0.5) &= \mathbb{E}[(x+2)^2] \odot 0.5^3 \\
&\;\;\vdots \\
W^n_{0.5}(\mathbb{E}[x^2] \odot 0.5) &= \mathbb{E}[(x+n)^2] \odot 0.5^{n+1}
\end{aligned}$$

This leads to:

$$\begin{aligned}&\mathsf{whp}\,[\![x \coloneqq x + 1^{\langle \frac{1}{2},\frac{1}{2}\rangle}]\!]\,(\mathbb{E}[x^2])\\ &= \bigoplus_{n\in\mathbb{N}} W_{0.5}^n(\mathbb{E}[x^2] \odot 0.5)\\ &= \bigoplus_{n\in\mathbb{N}} \mathbb{E}[(x+n)^2] \odot 0.5^{n+1}\end{aligned}$$

To compute $\mathsf{whp}\,[\![x \coloneqq x + 1^{\langle \frac{1}{2},\frac{1}{2}\rangle}]\!]\,(\mathbb{E}[x])$, we compute subsequent Kleene's iterates obtaining:

$$\begin{aligned}W_{0.5}^0(\mathbb{E}[x] \odot 0.5) &= \mathbb{E}[x] \odot 0.5\\ W_{0.5}^1(\mathbb{E}[x] \odot 0.5) &= \mathsf{whp}\,[\![x \coloneqq x+1]\!]\,(\mathbb{E}[x] \odot 0.5) \odot 0.5 = \mathbb{E}[x+1] \odot 0.5^2\\ W_{0.5}^2(\mathbb{E}[x] \odot 0.5) &= \mathbb{E}[x+2] \odot 0.5^3\\ &\vdots\\ W_{0.5}^n(\mathbb{E}[x] \odot 0.5) &= \mathbb{E}[x+n] \odot 0.5^{n+1}\end{aligned}$$

This leads to:

$$\begin{aligned}&\mathsf{whp}\,[\![x \coloneqq x + 1^{\langle \frac{1}{2},\frac{1}{2}\rangle}]\!]\,(\mathbb{E}[x]^2)\\ &= \Big(\bigoplus_{n\in\mathbb{N}} W_{0.5}^n(\mathbb{E}[x] \odot 0.5)\Big)^2\\ &= \Big(\bigoplus_{n\in\mathbb{N}} \mathbb{E}[x+n] \odot 0.5^{n+1}\Big)^2\end{aligned}$$

F.2 Conditional expected values

You decide to play a coin-toss game where winning yields 1, and losing results in a loss of 5. You plan ahead by adding specially crafted fake coins to your pocket that guarantee a win when tossed. In addition, you ensure you have some genuine fair coins to display to your opponent. How many coins must be in your pocket (at least) to have a non-negative expected return?

```
/// [c = 0] · 𝔼[1] + [c ≠ 0] · (1/2 𝔼[−5] + 1/2 𝔼[1])
if ( c = 0 ) {
    /// 𝔼[1]
    x := 1
    /// 𝔼[x]
} else {
    /// 1/2 𝔼[−5] + 1/2 𝔼[1]
    { x := −5 } [ 1/2 ] { x := 1 }
    /// 𝔼[x]
}
/// 𝔼[x]
```

With an input boolean variable c we represent whether we have a fair or a fake coin. We represent the game with the simple program C above and compute whp $[\![C]\!]\ (\mathbb{E}[x])$ which yields the expected return for a given input distribution. We observe that the shape of the input distribution must be $\mu = \frac{n-1}{n} \cdot \mathbf{1}_{c=0} + \frac{1}{n} \cdot \mathbf{1}_{c=1}$ and solve: whp $[\![C]\!]\ (\mathbb{E}[x])\ (\mu) \geq 0$, leading to:

$$\text{whp}\ [\![C]\!]\ (\mathbb{E}[x])\ (\frac{n-1}{n} \cdot \mathbf{1}_{c=0} + \frac{1}{n} \cdot \mathbf{1}_{c=1}) \geq 0$$

$$\big(\ [c=0] \cdot \mathbb{E}[1] + [c \neq 0] \cdot (\frac{1}{2}\mathbb{E}[-5] + \frac{1}{2}\mathbb{E}[1])\big)\big(\frac{n-1}{n} \cdot \mathbf{1}_{c=0} + \frac{1}{n} \cdot \mathbf{1}_{c=1}\big) \geq 0$$

$$\big(\ [c=0] \cdot \mathbb{E}[1]\big)\big(\frac{n-1}{n} \cdot \mathbf{1}_{c=0} + \frac{1}{n} \cdot \mathbf{1}_{c=1}\big) + \big(\ [c \neq 0] \cdot (\frac{1}{2}\mathbb{E}[-5] + \frac{1}{2}\mathbb{E}[1])\big)\big(\frac{n-1}{n} \cdot \mathbf{1}_{c=0} + \frac{1}{n} \cdot \mathbf{1}_{c=1}\big) \geq 0$$

$$\frac{n-1}{n} - \frac{2}{n} \geq 0$$

$$\frac{n-3}{n} \geq 0$$

$$n \geq 3$$

The result obtained, implies that you need at least 3 coins in your pocket (at least two fake coins and one fair coin) to guarantee a non-negative expected return in this coin-toss game.

Received 2024-04-06; accepted 2024-08-18