Electronic Notes in Theoretical Informatics And Computer Science

ENTICS

Volume 5
Proceedings of MFPS 2025

# Safety, Relative Tightness and the Probabilistic Frame Rule

Janez Ignacij Jereb[a,b] Alex Simpson[b,c,1,2]

[a] *Faculty of Computer and Information Science (FRI)*
*University of Ljubljana, Ljubljana, Slovenia*

[b] *Faculty of Mathematics and Physics (FMF)*
*University of Ljubljana, Ljubljana, Slovenia*

[c] *Institute for Mathematics, Physics and Mechanics (IMFM)*
*Lljubljana, Slovenia*

**Abstract**

Probabilistic separation logic offers an approach to reasoning about imperative probabilistic programs in which a separating conjunction is used as a mechanism for expressing independence properties. Crucial to the effectiveness of the formalism is the frame rule, which enables modular reasoning about independent probabilistic state. We explore a semantic formulation of probabilistic separation logic, in which the frame rule has the same simple formulation as in separation logic, without further side conditions. This is achieved by building a notion of safety into specifications, using which we establish a crucial property of specifications, called relative tightness, from which the soundness of the frame rule follows.

*Keywords:* probabilistic separation logic, separation logic, frame rule, partial state, operational semantics, partial correctness, total correctness, reasoning about independence

## 1 Introduction

Separation logic is an immensely successful formalism for the verification of imperative programs involving memory manipulation [8,12]. Critical to its success is the *frame rule*, which allows program verification to be carried out in a modular style, in which the specification and verification of subroutines makes reference only to resources local to the subroutine.

Probabilistic separation logic [3] adapts separation logic to programs with randomness. A main guiding aim is to utilise the power of the separating conjunction $*$ of separation logic and its associated frame rule to provide a modular means of verifying properties of probabilistic programs. In this setting, the formula $\Phi * \Psi$ asserts that $\Phi$ and $\Psi$ hold in *probabilistically independent* parts of memory. The original paper [3] established the framework for a fragment of pwhile (a simple probabilistic imperative language), and gave many examples of verification tasks (taken from cryptography) that can be handled by the approach. Subsequent developments have extended the framework with more elaborate probabilistic concepts such

[1] Email: Alex.Simpson@fmf.uni-lj.si

[2] Research supported by ARIS research programme P1 0294.

Rec'd Apr 3,2025; Pub'd Dec 15,2025
10.46298/entics.16743

Proceedings Available Online at
ENTICS MFPS 41 Proceedings

$$\frac{(\,\mathrm{FV}(\Theta)\cap \mathrm{MV}(C)=\emptyset \quad \mathrm{FV}(\Psi)\subseteq T\cup \mathrm{RV}(C)\cup \mathrm{WV}(C) \quad \models \Phi\to \mathbf{D}[T\cup \mathrm{RV}(C)]\,)\quad \{\Phi\}\,C\,\{\Psi\}}{\{\Phi * \Theta\}\,C\,\{\Psi * \Theta\}}$$

Fig. 1. The frame rule from [3]

as negative dependencies [2], and conditional independence [1]. An interesting related development is the Lilac program logic [7], which performs a similar task for probabilistic functional programs without mutable state. One of the advantages of the functional framework is that it allows the frame rule to have a simple, elegant formulation.

In the original imperative setting of [3,2], in contrast, the probabilistic frame rule has the more cumbersome form in Figure 1, with the three side conditions written in the top line of the rule. The first side condition, which appears also in the frame rule of ordinary (heap) separation logic [8,12], requires the program $C$ not to modify the values of the free variables of the assertion $\Theta$, a condition necessary to guarantee that the truth of $\Theta$ is preserved by the execution of $C$. The second and third side condition are specific to probabilistic separation logic. The second requires $\Psi$ to only involve free variables that are in the union of: a chosen set of variables $T$, the set $\mathrm{RV}(C)$ of variables that may be read by $C$ before being written to, and the set $\mathrm{WV}(C)$ of variables that $C$ must write to before they are read. The third side condition requires $\Phi$ to guarantee that all variables in $T$ and $\mathrm{RV}(C)$ have defined values.

In addition to the convoluted formulation of the frame rule, another feature of the papers [3,2] is that significant restrictions are imposed on the the programs they consider. One restriction is that a syntactic distinction is maintained between deterministic and probabilistic variables in programs, which leads to restrictions on the positioning of assignment statements, and which is too absolute to be able to cater for program reasoning in which one needs to take account of the local deterministic behaviour of a globally probabilistic variable. Another restriction is that loop guards are required to be deterministic. Yet another is that loops are restricted to a bounded number of iterations.

Since imperative probabilistic languages with mutable state are an important paradigm with many applications, and probabilistic separation logic is a promising approach to verification for such languages, it is worth investigating if some of the complications and restrictions above can be avoided. In this paper, we propose a framework for addressing this. Working with an unrestricted programming language (the full pwhile language), our main focus will be on understanding the frame rule and simplifying its formulation. We hope that future work will show that our framework offers a basis for extending the proof rules of [3,2] to support program verification for the unrestricted language.

Following the lead of heap separation logic [4], probabilistic separation logic has hitherto been built on the elegant and versatile framework of the logic of bunched implications [9,10] and its resource-monoid models [11]. In the probabilistic case [3,2], one obtains a satisfaction relation of the form $\Sigma \models \Phi$, where $\Sigma$ is a random state and $\Phi$ is an assertion. (We are deliberately simplifying the formulation to ease the discussion.) In order to obtain the frame rule of the present paper, we found we needed to take a small step back from this framework. While we also have a satisfaction 'relation' of the form $\Sigma \models \Phi$, we take the truth value of the relation as being *undefined* in the case that the random state $\Sigma$ does not provide sufficient information to determine the truth or falsity of $\Phi$. In the case that the relation $\Sigma \models \Phi$ is defined, it assumes either the value $\mathtt{tt}$ (which means $\Sigma$ satisfies $\Phi$) of $\mathtt{ff}$ (which means $\Sigma$ satisfies $\neg\Phi$), and indeed the laws of classical logic are validated. Thus we avoid one of the idiosyncrasies of the probabilistic separation logics in the literature, namely that they are inherently intuitionistic.

Another minor departure from the literature is our formulation of random state. There are two natural ways to approach this. The standard approach in the literature is to view a random state as given by a probability distribution on states. This approach goes back to [6]. In this paper, we instead view a random state as a state-valued random variable, defined in the usual way as a function from a sample space to states. We find that this approach allows a more intuitive and mathematically convenient formalism for working with separating conjunction and its associated frame rule. Nevertheless, it should be emphasised

that the two approaches are equivalent in the sense that one can translate from either one to the other.

Within the framework of random state as a random variable, the separating conjunction $\Phi * \Psi$ has a very straightforward meaning. It holds in a random state $\Sigma$ just when there are sets $U, V$ of program variables such that: $\Phi$ and $\Psi$ hold in the projections $\Sigma\restriction_U$ and $\Sigma\restriction_V$ of $\Sigma$ to $U$ and $V$ respectively, and additionally $\Sigma\restriction_U$ and $\Sigma\restriction_V$ are independent random variables. Departing from the literature on imperative probabilistic separation logic, we do not impose the requirement that the sets $U$ and $V$ be disjoint. [3] If these sets do overlap, then the independence condition forces all variables in the overlap to be deterministic. This allows us to accommodate the sharing of deterministic variables between $\Phi$ and $\Psi$, a feature of probabilistic separation logic which is important to applications, without needing a separate syntactic class of deterministic variables.

Our approach to probabilistic separation logic will be semantic. Rather than fixing a specific syntax for the logic, we define a semantic notion of assertion, and introduce connectives as operations on semantic assertions. This approach, which allows us to focus on semantic principles, is very much influenced by [12], which uses a similar approach to provide a semantic underpinning for the frame rule of heap separation logic. Our development is also greatly influenced by the conceptual analysis of the frame rule in *op. cit.*, in which it is related to two properties of specifications: *safety* and *tightness*. Safety means that specifications $\{\Phi\}C\{\Psi\}$ include a guarantee that if $C$ is run from a state satisfying the precondition $\Phi$ then the execution does not fault. Tightness means that the specification mentions all resources involved in the execution of $C$. In our probabilistic framework, specifications $\{\Phi\}C\{\Psi\}$ will indeed include a *safety* component: if a random state $\Sigma$ satisfies $\Phi$, then $C$, run from $\Sigma$, does not incur a memory fault. As a consequence of this safety property, we shall derive a precisely formulated version of tightness, which we call *relative tightness*: $\Phi$ necessarily specifies everything about the random state $\Sigma$ that is relevant to the behaviour of $C$ on all resources needed for interpreting the postcondition $\Psi$. (Here "relative" means relative to $\Psi$.) In our framework, in which random states are random variables, this relative tightness property has an elegant formulation in terms of conditional independence (Theorem 6.3). The soundness of the frame rule, which appears in its original formulation without further side conditions, then follows from relative tightness by a pleasingly abstract argument exploiting general properties of independence and conditional independence (Corollary 6.4).

After reviewing mathematical preliminaries in Section 2, we present the pwhile language in Section 3 together with a small-step operational semantics. In Section 4, we introduce our semantic notion of assertion based on random state. Section 5 treats specifications, given as Hoare triples, and defines their partial-correctness and total-correctness interpretations, which, importantly, come with a built-in safety guarantee. The main contributions of the paper then appear in Section 6. We present the frame rule and prove its soundness utilising the notion of relative tightness, which is formulated in Theorem 6.3. We also provide a simple counterexample showing that relative tightness and the soundness of the frame rule both fail if the safety guarantee of specifications is dropped. Finally, in Section 7, we discuss prospects for extending the setting in this paper to a fully-fledged verification logic for the pwhile language.

## 2 Mathematical preliminaries

A *(discrete probability) distribution* on a set $A$ is a function $d\colon A \to [0,1]$ that satisfies

$$\sum_{a\in A} d(a) \;=\; 1\,.$$

Its *support* is defined by

$$\mathsf{Supp}(d) \;:=\; \{a \in A \mid d(a) > 0\}\,.$$

[3] The disjointness requirement is also not present in the functional setting of [7].

The support $\mathsf{Supp}(d)$ is necessarily a countable set. We write $\mathcal{D}(A)$ for the set of all distributions on $A$. A function $f : A \to B$ induces a function $f_! : \mathcal{D}(A) \to \mathcal{D}(B)$ that maps every $d \in \mathcal{D}(A)$ to its *pushforward*

$$f_!(d)(b) \;:=\; \sum_{a \in f^{-1}(b)} d(a)\,.$$

For an arbitrary set $A$, an $A$-valued *random variable* is a function $\Sigma : \Omega \to \mathsf{State}$, where $\Omega$ is a (discrete) *sample space*, namely a (necessarily countable) set equipped with a probability distribution $p_\Omega \colon \Omega \to [0,1]$, for which we require the additional (not strictly necessary, but natural) condition that $\mathsf{Supp}(d) = \Omega$; that is, there are no zero-probability sample-space elements. It will sometimes be useful to change sample space along a *morphism of sample spaces*, which is a function $q : \Omega' \to \Omega$ satisfying, $q_!(p_{\Omega'}) = p_\Omega$. Since sample spaces have no zero-probability elements, every morphism of sample spaces is surjective. To emphasise this point in the sequel, we shall write $q : \Omega' \twoheadrightarrow \Omega$ for sample space morphisms.

Given a random variable $S : \Omega \to A$, we write $p_S : A \to [0,1]$ for the *distribution* of $S$, given by the pushforward probability distribution $S_!(p_\Omega)$ on $A$. We define the *support* of a random variable $S$ by $\mathsf{Supp}(S) := \mathsf{Supp}(p_S)$.

Two random variables $S : \Omega \to A$ and $T : \Omega \to B$ are said to be *independent* (notation $S \perp\!\!\!\perp T$) if, for all $a \in A$ and $b \in B$, it holds that $p_{(S,T)}(a,b) = p_S(a) \cdot p_T(b)$, where the left-hand side refers to the induced random variable $(S,T) : \Omega \to A \times B$. It is obvious from this definition that independence is a symmetric relation. We shall need the following standard property of independence.

$$\text{If } S \perp\!\!\!\perp T \text{ then, for any } f : B \to C, \text{ it holds that } S \perp\!\!\!\perp f \circ T\,. \tag{1}$$

Given two random variables $S : \Omega \to A$ and $T : \Omega \to B$, and given $a \in \mathsf{Supp}(S)$, the *conditioned random variable* $T_{|S=a} : \Omega_{|S=a} \to B$ is defined by

$$\begin{aligned} \Omega_{|S=a} \;&:=\; \{\omega \in \Omega \mid S(\omega) = a\} \\ p_{\Omega_{|S=a}}(\omega) \;&:=\; \frac{p_\Omega(\omega)}{p_S(a)} \\ T_{|S=a}(\omega) \;&:=\; T(\omega)\,. \end{aligned}$$

Independence can be characterised in terms of conditioning, *viz*

$$S \perp\!\!\!\perp T \iff \forall b, b' \in \mathsf{Supp}(T),\; p_{S_{|T=b}} = p_{S_{|T=b'}}\,. \tag{2}$$

The relation of *conditional independence* of $S : \Omega \to A$ and $T : \Omega \to B$ *given* $U : \Omega \to C$ is defined by

$$S \perp\!\!\!\perp T \mid U \;:\Longleftrightarrow\; \forall c \in \mathsf{Supp}(U),\; S_{|U=c} \perp\!\!\!\perp T_{|U=c}\,. \tag{3}$$

We shall make use of the following standard property of conditional independence.

$$\text{If } S \perp\!\!\!\perp T \mid U \text{ and } S \perp\!\!\!\perp U \text{ then } S \perp\!\!\!\perp (T,U)\,. \tag{4}$$

The paper will study a programming language pwhile, introduced in Section 3, which manipulates state. For us, a *state* $\sigma$ is a finite partial function $\sigma : \mathsf{Var} \rightharpoonup \mathbb{Z}$, where $\mathsf{Var}$ is a fixed set of program variables. Equivalently, a state is a total function from a finite set $\mathsf{Dom}(\sigma) \subseteq \mathsf{Var}$, the *domain* of $\sigma$, to $\mathbb{Z}$. We write $\mathsf{State}$ for the set of states, which carries a natural partial order

$$\sigma \sqsubseteq \sigma' \;:\Longleftrightarrow\; \forall X \in \mathsf{Var},\, \sigma(X)\!\downarrow \implies \sigma'(X) = \sigma(X)\,, \tag{5}$$

where we write $\sigma(X)\downarrow$ to mean that $X \in \mathsf{Dom}(\sigma)$. A useful operation will be to *restrict* a state $\sigma$ to a given finite $V \subseteq \mathsf{Var}$, resulting in the state

$$\sigma\restriction_V(X) \;:=\; \begin{cases}\sigma(X) & \text{if } X \in V \\ \bot & \text{otherwise}\,,\end{cases} \tag{6}$$

where we use $\bot$ to denote undefinedness. Note that $\sigma\restriction_V(X)$ is undefined for those $X \in V$ for which $\sigma(X)$ is undefined. In the special case that $\sigma(X)$ is defined for all $X \in V$ (equivalently if $\mathsf{Dom}(\sigma) \subseteq V$), we say that $\sigma$ is *$V$-total*.

In order to model the probabilistic assertion logic of Section 4, we need to randomise state. A *random state* is simply a $\mathsf{State}$-valued random variable $\Sigma : \Omega \to \mathsf{State}$. The partial order on $\mathsf{State}$ (5) induces an associated pointwise partial order on the set of $\mathsf{State}$-valued random variables with common sample space $\Omega$. Suppose $\Sigma, \Sigma' : \Omega \to \mathsf{State}$, then

$$\Sigma \sqsubseteq \Sigma' \;:\Longleftrightarrow\; \forall \omega \in \Omega,\, \Sigma(\omega) \sqsubseteq \Sigma'(\omega)\,. \tag{7}$$

The restriction operation on states (6) induces an associated operation on random state. Given a random state $\Sigma : \Omega \to \mathsf{State}$ and finite $V \subseteq \mathsf{Var}$, the restricted random state $\Sigma\restriction_V : \Omega \to \mathsf{State}$ is defined by:

$$\Sigma\restriction_V(\omega) \;:=\; \Sigma(\omega)\restriction_V .$$

The random state $\Sigma$ is said to be *$V$-total* if $\mathsf{Pr}[\Sigma \text{ is } V\text{-total}] = 1$. In spite of appearances, this is not a circular definition. We are of course using standard notational conventions for random variables to abbreviate the property

$$\sum_{\omega \in \Omega} \begin{cases} p_\Omega(\omega) & \Sigma(\omega) \text{ is } V\text{-total} \\ 0 & \text{otherwise.}\end{cases} \;=\; 1\,.$$

In the case that $\Sigma$ is $\{X\}$-total, we write $\Sigma^X : \Omega \to \mathbb{Z}$ for the random integer

$$\Sigma^X(\omega) \;:=\; \Sigma(\omega)(X)\,. \tag{8}$$

## 3 Language

We introduce a version of the simple probabilistic while language pwhile. We use $X, Y, Z \ldots$ to range over an infinite set $\mathsf{Var}$ of integer variables. We use $E, \ldots$ to range over integer expressions and $B, \ldots$ to range over boolean expressions. Such expressions are built as usual using variables, basic arithmetic operations, basic relations and logical connectives. We leave the precise syntax open. In the programming language, we also include *distribution expressions*, which denote probability distributions over the integers. Distribution expressions may contain integer subexpressions, allowing us to define distributions parametrically. For example, we might have an expression $\mathsf{uniform}(X)$, denoting the uniform probability distribution on the integer interval $[\min(0, X), \max(0, X)]$. We use $D, \ldots$ to range over distribution expressions. Again, we leave the precise syntax unspecified. With the above conventions, the syntax of commands is given by

$$C \;::=\; X := E \mid X \overset{\$}{\leftarrow} D \mid \mathsf{skip} \mid C; C \mid \mathsf{if}\ B\ \mathsf{then}\ C\ \mathsf{else}\ C \mid \mathsf{while}\ B\ \mathsf{do}\ C\,,$$

where the command $X \overset{\$}{\leftarrow} D$ randomly samples from the distribution $D$ and assigns the returned value to the variable $X$.

We define a small-step operational semantics, by specifying a probabilistic transition system, with mid-execution transitions between configurations of the form $C, \sigma$, where $\sigma \in \mathsf{State}$ as defined in Section 2 Execution may terminate in a terminal configuration $\sigma$. It may also abort, due to a memory fault, in the $\mathsf{fault}$ error node. The transition system thus has the following nodes.

- *Nonterminal configurations:* $C, \sigma$, where $C$ is a command and $\sigma$ a state.

$$\frac{}{X := E\,,\,\sigma \longrightarrow \sigma[X \mapsto n]}\ [\![E]\!]_\sigma = n \qquad \frac{}{X := E\,,\,\sigma \longrightarrow \mathsf{fault}}\ \mathsf{Var}(E) \not\subseteq \mathsf{Dom}(\sigma)$$

$$\frac{}{X \overset{\$}{\leftarrow} D\,,\,\sigma \xrightarrow{n \overset{\$}{\leftarrow} d} \sigma[X \mapsto n]}\ [\![D]\!]_\sigma = d\,,\ n \in \mathsf{Supp}(d) \qquad \frac{}{X \overset{\$}{\leftarrow} D\,,\,\sigma \longrightarrow \mathsf{fault}}\ \mathsf{Var}(D) \not\subseteq \mathsf{Dom}(\sigma)$$

$$\frac{}{\mathsf{skip}, \sigma \longrightarrow \sigma}$$

$$\frac{C_1, \sigma \xrightarrow{L} C_1', \sigma'}{C_1; C_2\,,\,\sigma \xrightarrow{L} C_1'; C_2\,,\,\sigma'} \qquad \frac{C_1, \sigma \xrightarrow{L} \sigma'}{C_1; C_2\,,\,\sigma \xrightarrow{L} C_2\,,\,\sigma'} \qquad \frac{C_1, \sigma \longrightarrow \mathsf{fault}}{C_1; C_2\,,\,\sigma \longrightarrow \mathsf{fault}}$$

$$\frac{}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\,,\,\sigma \longrightarrow C_1, \sigma}\ [\![B]\!]_\sigma = \mathsf{tt} \qquad \frac{}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\,,\,\sigma \longrightarrow C_2, \sigma}\ [\![b]\!]_\sigma = \mathsf{ff}$$

$$\frac{}{\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\,,\,\sigma \longrightarrow \mathsf{fault}}\ \mathsf{Var}(b) \not\subseteq \mathsf{Dom}(\sigma)$$

$$\frac{}{\mathsf{while}\ B\ \mathsf{do}\ C\,,\,\sigma \longrightarrow \sigma}\ [\![B]\!]_\sigma = \mathsf{ff} \qquad \frac{}{\mathsf{while}\ B\ \mathsf{do}\ C\,,\,\sigma \longrightarrow C\,;\mathsf{while}\ B\ \mathsf{do}\ C\,,\,\sigma}\ [\![B]\!]_\sigma = \mathsf{tt}$$

$$\frac{}{\mathsf{while}\ B\ \mathsf{do}\ C\,,\,\sigma \longrightarrow \mathsf{fault}}\ \mathsf{Var}(b) \not\subseteq \mathsf{Dom}(\sigma)$$

Fig. 2. Operational Semantics

- *Terminal configurations:* $\sigma$, where $\sigma$ is a state.
- *The error node:* fault.

There are two types of transition.

- *Deterministic transitions:* unlabelled transitions $u \longrightarrow v$, where $u$ is nonterminal.
- *Probabilistic transitions:* labelled transitions $u \xrightarrow{n \overset{\$}{\leftarrow} d} v$, where $u$ is nonterminal, $d$ is a probability distribution on $\mathbb{Z}$ and $n \in \mathsf{Supp}(d)$.

The single-step transitions between nodes are determined inductively using the rules in Figure 2. In these rules, we write $\mathsf{Var}(E)$ for the set of variables appearing in an integer expression $E$, and we use similar notation for boolean expressions $B$ and distribution expressions $D$. The semantic interpretation $[\![E]\!]_\sigma$ of an integer expression $E$, relative to a state $\sigma$, is defined if and only if $\mathsf{Var}(E) \subseteq \mathsf{Dom}(\sigma)$. Similarly, for boolean and distribution expressions, $[\![B]\!]_\sigma$ and $[\![D]\!]_\sigma$ are defined if and only $\mathsf{Var}(B) \subseteq \mathsf{Dom}(\sigma)$ and $\mathsf{Var}(D) \subseteq \mathsf{Dom}(\sigma)$ respectively. When defined, we have $[\![E]\!]_\sigma \in \mathbb{Z}$, $[\![B]\!]_\sigma \in \{\mathsf{tt}, \mathsf{ff}\}$ (our notation for truth values) and $[\![D]\!]_\sigma \in \mathcal{D}(\mathbb{Z})$. Whenever an expression is evaluated in a state in which some of its variables do not have a value, a memory fault is triggered. In the two rules involving variable assignment, $\sigma[X \mapsto n]$ is used for the state that maps $X$ to $n$, and agrees with $\sigma$ (which may or may not contain $X$ in its domain) otherwise.The other notational convention in Figure 2 is that $L$ is a meta-variable for transition labels, which may be empty in the case of deterministic transitions, or of the form $n \overset{\$}{\leftarrow} d$ in the case of probabilistic transitions.

Every transition path from a nonterminal configuration $C_0, \sigma_0$ to a terminal configuration has the form

$$C_0, \sigma_0 \xrightarrow{L_0} C_1, \sigma_1 \xrightarrow{L_1} \cdots\ C_{n-1}, \sigma_{n-1} \xrightarrow{L_{n-1}} \tau\,, \tag{9}$$

for some $n \geq 1$, where each label $L_i$ is either empty or of the form $n_i \xleftarrow{\$} d_i$. We call such paths *terminal paths*. The *probability* and the *final state* of a terminal path $\pi$, of the form (9), are defined by

$$\begin{aligned} \mathsf{Pr}(\pi) &:= \prod_{i<n} \begin{cases} d_i(n_i) & \text{if } L_i \text{ is } n_i \xleftarrow{\$} d_i \\ 1 & \text{if } L_i \text{ is empty} \end{cases} , \\ \mathsf{final}(\pi) &:= \tau \,. \end{aligned} \tag{10}$$

We write $\mathcal{TP}_{C,\sigma}$ for the set of all terminal paths from initial configuration $C, \sigma$. The *termination probability* for $C, \sigma$ is defined by

$$\mathsf{Pr}[C, \sigma \text{ terminates}] := \sum_{\pi \in \mathcal{TP}_{C,\sigma}} \mathsf{Pr}(\pi) \,.$$

We say that $C, \sigma$ is *terminating* if $\mathsf{Pr}[C, \sigma \text{ terminates}] = 1$. This is the form of termination known as almost sure termination. We shall not concern ourselves with other forms of termination. A terminating configuration $C, \sigma$ determines a probability distribution on result states $p_{C,\Sigma} : \mathsf{State} \to [0, 1]$, defined by:

$$p_{C,\Sigma}(\tau) := \sum_{\pi \in \mathcal{TP}_{C,\sigma}} \mathsf{Pr}(\pi) \cdot \delta_{\tau,\, \mathsf{final}(\pi)} \,, \tag{11}$$

where $\delta$ is the Krönecker delta.

We consider *divergence* to be the computational behaviour that follows an infinite path of nonterminal configurations

$$C_0, \sigma_0 \xrightarrow{L_0} C_1, \sigma_1 \xrightarrow{L_1} C_1, \sigma_1 \xrightarrow{L_2} \cdots \,. \tag{12}$$

We write $\mathcal{IP}_{C_0,\sigma_0}$ for the set of all such infinite paths from initial configuration $C_0, \sigma_0$. We write $\mathcal{IP}_{C_0,\sigma_0}{\restriction_n}$ for the set of $n$-length prefixes of paths in $\mathcal{IP}_{C_0,\sigma_0}$, where the length is the number of transitions. The *divergence probability* for $C_0, \sigma_0$ is defined by

$$\mathsf{Pr}[C_0, \sigma_0 \text{ diverges}] := \lim_{n\to\infty} \sum_{\gamma \in \mathcal{IP}_{C_0,\sigma_0}{\restriction_n}} \mathsf{Pr}(\gamma) \,, \tag{13}$$

where $\mathsf{Pr}(\gamma)$ is calculated analogously to (10) above. It is always the case that

$$\mathsf{Pr}[C_0, \sigma_0 \text{ terminates}] + \mathsf{Pr}[C_0, \sigma_0 \text{ diverges}] \leq 1 \,. \tag{14}$$

The inequality is an equality precisely in the case that the configuration $C_0, \sigma_0$ is *fault-free*, meaning that it is not the case that $C, \sigma \longrightarrow^* \mathsf{fault}$, where we write $\longrightarrow^*$ for the transitive-reflexive closure of the single-step transition relation, including both labelled and unlabelled transitions in the relation. Note that terminating configurations are *a fortiori* fault-free.

## 4 Assertions

Rather than focusing on a specific assertion logic, we work with a general semantic notion of assertion. To motivate this, we give some examples of assertions that fit into the framework. A first example is

$$X \sim d \,,$$

where $d$ is some probability distribution on $\mathbb{Z}$. This says that the value of the program variable $X$ is distributed probabilistically according to $d$. Another simple example is

$$[X = Y] \,.$$

Since the values of variables are in general randomly distributed, we interpret the above equality as stating that $X$ and $Y$ take the same value with probability 1; that is, considered as random variables, $X$ and $Y$ are almost surely equal.

In order to accommodate randomness in the semantics of assertions, our main satisfaction relation will have the form $\Sigma \models \Phi$, relating random states $\Sigma$ and semantic assertions $\Phi$. However, we add one further ingredient to the picture. We allow the truth value of the 'relation' $\Sigma \models \Phi$ to be undefined for certain ineligible random states $\Sigma$. This possibility of undefinedness is included as a natural mathematical mechanism for dealing with the partiality of state. For example, we consider $\Sigma \models [X = Y]$ to be defined if and only if the random state $\Sigma$ assigns values to both $X$ and $Y$ with probability one; that is, if $\Sigma$ is $\{X, Y\}$-total. We write $(\Sigma \models \Phi)\!\downarrow$ to mean that the value of the satisfaction relation $\Sigma \models \Phi$ is defined, in which case we either have $(\Sigma \models \Phi) = \mathsf{tt}$ or $(\Sigma \models \Phi) = \mathsf{ff}$. Again, in the case that $\Phi$ is the assertion $[X = Y]$, we have that $(\Sigma \models [X = Y])\!\downarrow$ if and only if $\Sigma^X, \Sigma^Y : \Omega \to \mathbb{Z}$, as in (8), are well-defined. Then $(\Sigma \models [X = Y]) = \mathsf{tt}$ if and only if $\mathsf{Pr}[\Sigma_X = \Sigma_Y] = 1$, or equivalently, because there are no zero-probability sample-space elements, if the functions $\Sigma^X$ and $\Sigma^Y$ are equal. Similarly, $(\Sigma \models [X = Y]) = \mathsf{ff}$ if and only if $\mathsf{Pr}[\Sigma^X = \Sigma^Y] < 1$, or equivalently $\Sigma^X \neq \Sigma^Y$. This and many other similar examples illustrate that allowing undefinedness enables us to make a natural distinction between the property expressed by $\Phi$ not making sense in the context of $\Sigma$, in which case $\Sigma \models \Phi$ is undefined, and the case in which the property does make sense, in which case $\Sigma \models \Phi$ possesses an actual truth value.

We now precisely define our semantic notion of assertion. A *semantic assertion* $\Phi$ is given by a *partial* function from random states to the set of truth values $\{\mathsf{tt}, \mathsf{ff}\}$. We write this function as

$$\Sigma \;\mapsto\; (\Sigma \models \Phi)\,.$$

The function is required to satisfy three conditions.

**(SA1)** $\Sigma \sqsubseteq \Sigma'$ and $(\Sigma \models \Phi)\!\downarrow$ implies $(\Sigma \models \Phi) = (\Sigma' \models \Phi)$. (The $\sqsubseteq$ relation is defined in (7).)

**(SA2)** $\Phi$ has an associated finite set $\mathrm{FV}(\Phi) \subseteq \mathsf{Var}$, its *footprint variables*,[4] satisfying

$$(\Sigma \models \Phi)\!\downarrow \quad\Longleftrightarrow\quad \Sigma \text{ is } \mathrm{FV}(\Phi)\text{-total.}$$

**(SA3)** If $q : \Omega' \twoheadrightarrow \Omega$ is a morphism of sample spaces then $(\Sigma \circ q \models \Phi) = (\Sigma \models \Phi)$.

One consequence of (SA2) is that it prevents the everywhere undefined function $(\Sigma \mapsto \bot)$ from being a semantic assertion, which is no loss. Secondly, (SA2) determines $\mathrm{FV}(\Phi)$ uniquely. Indeed, suppose we have two finite $U, V \subseteq \mathsf{Var}$ such that $(\Sigma \models \Phi)\!\downarrow$ iff $\Sigma$ is $U$-total iff $\Sigma$ is $V$-total. Let $\Sigma$ be some random state such that $(\Sigma \models \Phi)\!\downarrow$. (Such a state exists, by the first observation about (SA2) above.) Since $\Sigma$ is $U$-total, so is $\Sigma\restriction_U$, and hence $(\Sigma\restriction_U \models \Phi)\!\downarrow$. This means that $\Sigma\restriction_U$ is $V$-total, which entails $V \subseteq U$. A symmetric argument establishes $U \subseteq V$. Hence $U = V$.

In combination, (SA1) and (SA2) have the following simple consequence that we shall use frequently.

$$(\Sigma \models \Phi)\!\downarrow \quad\Longrightarrow\quad (\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Phi)\!\downarrow \text{ and } (\Sigma \models \Phi) = (\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Phi)\,. \tag{15}$$

We remark also that (SA3) is equivalent to saying that, for any assertion $\Phi$, the value of $\Sigma \models \Phi$ depends only on the probability distribution $p_\Sigma$. Thus the logic could perfectly well be given an equivalent formulation in terms of distributions on $\mathsf{State}$ rather than random state, as has been the norm hitherto in the literature on probabilistic separation logic [3,2].

We give several examples of semantic assertions and constructions on them, to illustrate the broad scope of the notion.

[4] When assertions are given syntactically as formulas, $\mathrm{FV}(\Phi)$ can be understood as the *free variables* of the formula $\Phi$. For semantic assertions, we prefer the more neutral terminology *footprint variables*, which is motivated by thinking of $\mathrm{FV}(\Phi)$ as the memory footprint associated with $\Phi$.

- If $B$ is a boolean expression then the assertion $[B]$ is defined by

$$\begin{aligned} \mathrm{FV}([B]) &:= \mathsf{Var}(B) \\ \Sigma \models [B] &:\Longleftrightarrow \mathsf{Pr}[[\![B]\!]_\Sigma = \mathsf{tt}] = 1\,. \end{aligned}$$

  This definition illustrates the general style we shall follow below. The first clause specifies the set of footprint variables, which determines when the value of $(\Sigma \models [B])$ is defined. The second clause, is then only considered in the case that $(\Sigma \models [B])\!\downarrow$.
- A special case of the above is the assertion $[E_1 = E_2]$, for integer expressions $E_1, E_2$.
- If $E$ and $D$ are integer and distribution expressions respectively then the assertion $E \sim D$ is defined by

$$\begin{aligned} \mathrm{FV}(E \sim D) &:= \mathsf{Var}(E) \cup \mathsf{Var}(D) \\ \Sigma \models E \sim D &:\Longleftrightarrow \forall d \in \mathsf{Supp}([\![D]\!]_\Sigma),\ p_{([\![E]\!]_\Sigma)_{|[\![D]\!]_\Sigma = d}} = d\,. \end{aligned}$$

  Here, the expression $[\![D]\!]_\Sigma$ is considered as the distribution-valued random variable $\omega \mapsto [\![D]\!]_{\Sigma(\omega)}$, and the property asserts that the distribution of $E$, conditional on any possible distribution $d$ arising from $D$, is itself $d$.
- If $E$ is an integer expression then the assertion $\mathsf{Det}(E)$ is defined by

$$\begin{aligned} \mathrm{FV}(\mathsf{Det}(E)) &:= \mathsf{Var}(E) \\ \Sigma \models \mathsf{Det}(E) &:\Longleftrightarrow \exists n \in \mathbb{Z},\ \mathsf{Pr}[[\![E]\!]_\Sigma = n] = 1\,. \end{aligned}$$

  Informally: $E$ is deterministic.
- If $\Phi$ is a semantic assertion then so is $\neg\Phi$ defined classically by

$$\begin{aligned} \mathrm{FV}(\neg\Phi) &:= \mathrm{FV}(\Phi) \\ \Sigma \models \neg\Phi &:\Longleftrightarrow \Sigma \not\models \Phi\,. \end{aligned}$$

  We emphasise that the second line only applies in the case that $(\Sigma \models \neg\Phi)\!\downarrow$.
- If $\Phi, \Psi$ are semantic assertions then $\Phi \vee \Psi$, $\Phi \wedge \Psi$ and $\Phi \to \Psi$, are defined in the standard classical way, for example

$$\begin{aligned} \mathrm{FV}(\Phi \to \Psi) &:= \mathrm{FV}(\Phi) \cup \mathrm{FV}(\Psi) \\ \Sigma \models \Phi \to \Psi &:\Longleftrightarrow \Sigma \not\models \Phi \text{ or } \Sigma \models \Psi,\,. \end{aligned}$$

- If $E$ is an integer expression and $\Phi$ is a semantic assertion then so is $\mathbf{C}_E\,\Phi$ defined by

$$\begin{aligned} \mathrm{FV}(\mathbf{C}_E\,\Phi) &:= \mathsf{Var}(E) \cup \mathrm{FV}(\Phi) \\ \Sigma \models \mathbf{C}_E\,\Phi &:\Longleftrightarrow \forall n \in \mathsf{Supp}([\![E]\!]_\Sigma),\ \Sigma_{|[\![E]\!]_\Sigma = n} \models \Phi\,. \end{aligned}$$

  Here $\mathbf{C}_E$ is essentially the *conditioning modality* introduced in [7], which we model by conditioning the random state $\Sigma$, using the conditioning operation on random variables from Section 2.
- If $\Phi, \Psi$ are semantic assertions then so is $\Phi * \Psi$ defined by

$$\begin{aligned} \mathrm{FV}(\Phi * \Psi) &:= \mathrm{FV}(\Phi) \cup \mathrm{FV}(\Psi) \\ \Sigma \models \Phi * \Psi &:\Longleftrightarrow \Sigma \models \Phi \text{ and } \Sigma \models \Psi \text{ and } \Sigma\restriction_{\mathrm{FV}(\Phi)} \perp\!\!\!\perp \Sigma\restriction_{\mathrm{FV}(\Psi)}\,, \end{aligned}$$

The last assertion above is of course the separating conjunction, which will play a critical role in the frame rule. One might expect the second clause in its definition to have a different form; for example,

$$\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Phi \text{ and } \Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Psi \text{ and } \Sigma\restriction_{\mathrm{FV}(\Phi)} \perp\!\!\!\perp \Sigma\restriction_{\mathrm{FV}(\Psi)}\,; \tag{16}$$

but this is equivalent by (15). Another equivalent is given by

$$\exists U, V \subseteq \mathsf{Var},\ \Sigma\restriction_U \models \Phi \text{ and } \Sigma\restriction_V \models \Psi \text{ and } \Sigma\restriction_U \perp\!\!\!\perp \Sigma\restriction_V . \tag{17}$$

Indeed, it is immediate that (16) implies (17). For the converse, suppose we have $U, V$ as in (17). Since $\Sigma\restriction_U \models \Phi$ and $\Sigma\restriction_V \models \Psi$, we have, by (SA2), that $\mathrm{FV}(\Phi) \subseteq U$ and $\mathrm{FV}(\Psi) \subseteq V$. So $\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Phi$, by (15), because $\Sigma\restriction_{\mathrm{FV}(\Phi)} = \Sigma\restriction_U\restriction_{\mathrm{FV}(\Phi)}$. Similarly, $\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Psi$. Finally, since, if we compose $\sigma \mapsto \sigma\restriction_{\mathrm{FV}\Phi}$ with the left-hand side and $\sigma \mapsto \sigma\restriction_{\mathrm{FV}\Psi}$ with the right-hand side of the independence statement $\Sigma\restriction_U \perp\!\!\!\perp \Sigma\restriction_V$, then it follows from (1) (and its symmetric version) that $\Sigma_{\mathrm{FV}(\Phi)} \perp\!\!\!\perp \Sigma_{\mathrm{FV}(\Psi)}$.

As a final comment about $*$, we remark that it is possible for $\Sigma \models \Phi * \Psi$ to hold in cases in which $\mathrm{FV}(\Phi) \cap \mathrm{FV}(\Psi) \neq \emptyset$. In such cases, by composing both sides of $\Sigma\restriction_{\mathrm{FV}(\Phi)} \perp\!\!\!\perp \Sigma\restriction_{\mathrm{FV}(\Psi)}$ with the function $\sigma \mapsto \sigma\restriction_V$, where $V := \mathrm{FV}(\Phi) \cap \mathrm{FV}(\Psi)$, it follows from (1) that $\Sigma\restriction_V \perp\!\!\!\perp \Sigma\restriction_V$. This means that $\Sigma\restriction_V$ is a deterministic random variable. That is, there exists a $V$-defined state $\sigma$ such that $\mathsf{Pr}[\Sigma\restriction_V = \sigma] = 1$. Thus our formulation of the separating conjunction, automatically entails the property that the two sides $\Phi$ and $\Psi$ can share deterministic variables, which in the original probabilistic separation logics is built-in via a syntactic distinction between deterministic and probabilistic variables [3,2].

## 5 Specifications

A *specification* is given by a Hoare triple

$$\{\Phi\}\, C\, \{\Psi\}\,,$$

where $\Phi$ and $\Psi$ are semantic assertions, and $C$ is a command from the programming language of Section 3. As is standard, we shall have two forms of correctness for specifications, *partial* and *total*, whose precise form, in which *safety* is built in, is influenced by heap separation logic [8,12]. Partial correctness will say that if $C$ is run in any random state $\Sigma$ for which $\Sigma \models \Phi$, then the execution of $C$ is fault-free (this is the safety component) and, if $C$ almost surely terminates, then $\mathrm{T} \models \Psi$, where T is the induced random state in which $C$ terminates. Total correctness more simply says that if $C$ is run in any random state $\Sigma$, for which $\Sigma \models \Phi$, then $C$ almost surely terminates (hence is *a fortiori* fault-free), and $\mathrm{T} \models \Psi$, where T is the termination random state. To make this precise, we have to define the random state in which a terminating program terminates.

Let $\Sigma : \Omega \to \mathsf{State}$ be a random state. We view the pair $C, \Sigma$ as a random configuration (with deterministic first component). The notions of fault-freeness and termination for deterministic configurations, defined in Section 3, extend to random configurations in the natural way. Specifically, we say that $C, \Sigma$ is *fault-free* if $\mathsf{Pr}[C, \Sigma \text{ is fault-free}] = 1$, [5] and is *terminating* if $\mathsf{Pr}[C, \Sigma \text{ is terminating}] = 1$.

$$\begin{aligned} \Omega_{C,\Sigma} &:= \{(\omega, \pi) \mid \omega \in \Omega \text{ and } \pi \text{ is a terminal path from } C, \Sigma(\omega)\} \\ p_{\Omega_{C,\Sigma}}(\omega, \pi) &:= p_\Omega(\omega) \cdot \mathsf{Pr}(\pi)\,. \end{aligned}$$

Note that it is because of the condition that $C, \Sigma$ is terminating that $p_{\Omega_{C,\Sigma}}$ is a probability distribution on $\Omega_{C,\Sigma}$. Also note that, for the same reason, the projection function $q_{C,\Sigma} : \Omega_{C,\Sigma} \twoheadrightarrow \Omega$, defined by $q_{C,\Sigma}(\omega, \pi) := \omega$, is a morphism of sample spaces. Define $\mathrm{T}_{C,\Sigma} : \Omega_{C,\Sigma} \to \mathsf{State}$ by

$$\mathrm{T}_{C,\Sigma}(\omega, \pi) \ := \ \mathsf{final}(\pi)\,.$$

It is $\mathrm{T}_{C,\Sigma}$ that provides the desired termination random state of the random configuration $C, \Sigma$.

**Definition 5.1** [Partial correctness] A specification $\{\Phi\}\, C\, \{\Psi\}$ is *partially correct* if, for every random state $\Sigma : \Omega \to \mathsf{State}$ for which $\Sigma \models \Phi$ we have:

[5] Because $\mathsf{Supp}(p_\Omega) = \Omega$, this is equivalent to the probability-free: for every $\omega \in \Omega$, the configuration $C, \Sigma(\omega)$ is fault-free. However, the probabilistic formulation is preferable, since it is independent of the design choice that $\mathsf{Supp}(p_\Omega) = \Omega$.

$$\frac{\{\Phi\}\, C\, \{\Psi\}}{\{\Phi * \Theta\}\, C\, \{\Psi * \Theta\}}\ \mathrm{FV}(\Theta) \cap \mathrm{MV}(C) = \emptyset$$

Fig. 3. The frame rule

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{MV}(X \overset{\$}{\leftarrow} D) & := & \{X\} & \mathrm{MV}(C_1; C_2) & := & \mathrm{MV}(C_1) \cup \mathrm{MV}(C_2) \\
\mathrm{MV}(X := E) & := & \{X\} & \mathrm{MV}(\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) & := & \mathrm{MV}(C_1) \cup \mathrm{MV}(C_2) \\
\mathrm{MV}(\mathsf{skip}) & := & \emptyset & \mathrm{MV}(\mathsf{while}\ B\ \mathsf{do}\ C) & := & \mathrm{MV}(C)\,.
\end{array}$$

Fig. 4. Modified variables $\mathrm{MV}(C)$

- $C, \Sigma$ is fault-free (the *safety guarantee*), and
- if $C, \Sigma$ is terminating then $\mathrm{T}_{C,\Sigma} \models \Psi$.

**Definition 5.2** [Total correctness] A specification $\{\Phi\}\, C\, \{\Psi\}$ is *totally correct* if, for every random state $\Sigma : \Omega \to \mathsf{State}$ for which $\Sigma \models \Phi$ we have:

- $C, \Sigma$ is terminating and $\mathrm{T}_{C,\Sigma} \models \Psi$.

It is trivial that every totally correct specification is partially correct.

## 6 The frame rule

The frame rule is given in Figure 3. Its formulation is identical to the original frame rule formulation in heap separation logic [12]. The single side condition involves the set $\mathrm{MV}(C)$ of all variables that the program $C$ is able to modify, which is defined (in the obvious way) in Figure 4. The lemmas below simply state that variables outside $\mathrm{MV}(C)$ do not change during the execution of $C$. The statement we need later is given by Lemma 6.2, which asserts this for random state. This is in turn a simple consequence of Lemma 6.1, which states the property for deterministic state. The straightforward proofs are omitted.

**Lemma 6.1** *If $V \subseteq \mathsf{Var}$ is such that $V \cap \mathrm{MV}(C) = \emptyset$, and if $C, \sigma \longrightarrow^* \tau$ then $\tau \restriction_V = \sigma \restriction_V$.*

**Lemma 6.2** *If $V \subseteq \mathsf{Var}$ is such that $V \cap \mathrm{MV}(C) = \emptyset$, and if $C, \Sigma$ is a terminating random configuration then $\mathrm{T}_{C,\Sigma} \restriction_V = \Sigma \restriction_V \circ q_{C,\Sigma}$.*

Rather than using the syntactic definition of $\mathrm{MV}(C)$ in Figure 4, we remark that one could take the satisfaction of Lemma 6.1 as a more permissive semantic definition of the set $\mathrm{MV}(C)$ , and the proof of the frame rule below remains valid.

The main work in proving the soundness of the frame rule lies in proving the theorem below, which formulates the *relative tightness* property outlined in Section 1. This asserts a fundamental property of partial (and hence also total) correctness specifications: the portion of the final state $\mathrm{T}_{C,\Sigma}$ that is relevant to interpreting the postcondition $\Psi$, namely $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)}$, depends only on that part of the start state $\Sigma$ that is relevant to interpreting the precondition $\Phi$, namely $\Sigma \restriction_{\mathrm{FV}(\Phi)}$. Since the state is randomised, the property of depending only on $\Sigma \restriction_{\mathrm{FV}(\Phi)}$ can be expressed by saying that $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)}$ is conditionally independent of the input state $\Sigma$ conditional on $\Sigma \restriction_{\mathrm{FV}(\Phi)}$. In order to formulate this property precisely, the random variables involving $\Sigma$ need to be composed with the sample space morphism $q_{C,\Sigma} : \Omega_{C,\Sigma} \twoheadrightarrow \Omega$, so that all random variables lie over the same sample space $\Omega_{C,\Sigma}$.

**Theorem 6.3 (Relative tightness)** *Suppose $\{\Phi\}C\{\Psi\}$ is partially correct, $\Sigma \models \Phi$ and the random configuration $C, \Sigma$ is terminating, then*

$$\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp \Sigma \circ q_{C,\Sigma} \mid \Sigma \restriction_{\mathrm{FV}(\Phi)} \circ q_{C,\Sigma}\,. \tag{18}$$

Before giving the somewhat involved proof of Theorem 6.3, we show how the soundness of the frame rule follows as a consequence of relative tightness.

**Corollary 6.4 (Soundness of the frame rule)** *The frame rule is sound for both partial correctness and total correctness.*

**Proof.** We first consider partial correctness. Suppose $\{\Phi\}C\{\Psi\}$ is partially correct. Suppose also that $\Theta$ is such that $\mathrm{FV}(\Theta) \cap \mathrm{MV}(C) = \emptyset$. We need to show that the specification $\{\Phi * \Theta\}C\{\Psi * \Theta\}$ is partially correct.

Suppose then that $\Sigma \models \Phi * \Theta$. That is, we have

$$\Sigma \models \Phi \tag{19}$$
$$\Sigma \models \Theta \tag{20}$$
$$\Sigma_{\mathrm{FV}(\Phi)} \perp\!\!\!\perp \Sigma_{\mathrm{FV}(\Theta)} \tag{21}$$

By (19) and the partial correctness of $\{\Phi\}C\{\Psi\}$, it is immediate that $C, \Sigma$ is fault-free. Assume that $C, \Sigma$ is terminating. We need to show that $\mathrm{T}_{C,\Sigma} \models \Psi * \Theta$.

By (19) and the partial correctness of $\{\Phi\}C\{\Psi\}$, we have $\mathrm{T}_{C,\Sigma} \models \Psi$.

Since $\mathrm{FV}(\Theta) \cap \mathrm{MV}(C) = \emptyset$, we have, by Lemma 6.2, that

$$\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)} = \Sigma \restriction_{\mathrm{FV}(\Theta)} \circ\ q_{C,\Sigma}\,. \tag{22}$$

From (20), it follows that $\Sigma \restriction_{\mathrm{FV}(\Theta)} \models \Theta$, by (15), hence $\Sigma \restriction_{\mathrm{FV}(\Theta)} \circ q_{C,\Sigma} \models \Theta$ by (SA3), i.e., $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)} \models \Theta$. It then follows from (SA1) that $\mathrm{T}_{C,\Sigma} \models \Theta$.

Finally, we show that $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp \mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)}$. By (21), we have $\Sigma \restriction_{\mathrm{FV}(\Phi)} \circ q_{C,\Sigma} \perp\!\!\!\perp \Sigma \restriction_{\mathrm{FV}(\Theta)} \circ q_{C,\Sigma}$; equivalently, by (22) (and the symmetry of independence)

$$\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)} \perp\!\!\!\perp \Sigma \restriction_{\mathrm{FV}(\Phi)} \circ\ q_{C,\Sigma}\,. \tag{23}$$

By Theorem 6.3, we have $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp \Sigma \circ q_{C,\Sigma} \mid \Sigma \restriction_{\mathrm{FV}(\Phi)} \circ\ q_{C,\Sigma}$. It thus follows from (3) and (1), using the function $\sigma \mapsto \sigma \restriction_{\mathrm{FV}(\Theta)} : \mathsf{State} \to \mathsf{State}$, that $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp \Sigma \restriction_{\mathrm{FV}(\Theta)} \circ q_{C,\Sigma} \mid \Sigma \restriction_{\mathrm{FV}(\Phi)} \circ\ q_{C,\Sigma}$ ; i.e., by (22),

$$\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp \mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)} \mid \Sigma \restriction_{\mathrm{FV}(\Phi)} \circ\ q_{C,\Sigma}\,. \tag{24}$$

Applying (4) to (23) and (24), we get

$$\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp (\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)},\ \Sigma \restriction_{\mathrm{FV}(\Phi)} \circ\ q_{C,\Sigma})\,.$$

Finally, by another application of (1), using first projection $(\tau, \sigma) \mapsto \tau$ as the function, we obtain $\mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Psi)} \perp\!\!\!\perp \mathrm{T}_{C,\Sigma} \restriction_{\mathrm{FV}(\Theta)}$, as required.

The soundness of the frame rule for total correctness follows directly from soundness for partial correctness, because the termination of $C, \Sigma$, which is the only additional property needed to to show the total correctness of $\{\Phi * \Theta\}\, C\, \{\Psi * \Theta\}$, is immediate from the assumption that $\{\Phi\}\, C\, \{\Psi\}$ is totally correct. □

We now turn to the proof of Theorem 6.3. In preparation, we need a simple lemma about the operational semantics. Given $\sigma, \sigma' \in \mathsf{State}$, we define $\langle\sigma\rangle\sigma' \in \mathsf{State}$ (the *masking of* $\sigma$ *by* $\sigma'$) by:

$$\langle\sigma\rangle\sigma'(X) \ :=\ \begin{cases} \sigma'(X) & \text{if } X \in \mathsf{Dom}(\sigma') \\ \sigma(X) & \text{otherwise}\,. \end{cases}$$

It is immediate from the definition that $\sigma' \sqsubseteq \langle\sigma\rangle\sigma'$. Also we have:

$$\sigma' \sqsubseteq \sigma \implies \langle\sigma\rangle\sigma' = \sigma\,. \tag{25}$$

**Lemma 6.5 (Masking lemma)** *For any state $\sigma$, configuration $C_0, \sigma_0$ and terminal path $\pi \in \mathcal{TP}_{C_0,\sigma_0}$ of the form (9), we have $\langle\sigma\rangle\pi \in \mathcal{TP}_{C_0,\langle\sigma\rangle\sigma_0}$, where the masked path $\langle\sigma\rangle\pi$ is defined by*

$$\langle\sigma\rangle\pi \;:=\; C_0, \langle\sigma\rangle\sigma_0 \xrightarrow{L_0} C_1, \langle\sigma\rangle\sigma_1 \xrightarrow{L_1} \cdots\; C_{n-1}, \langle\sigma\rangle\sigma_{n-1} \xrightarrow{L_{n-1}} \langle\sigma\rangle\tau\,,$$

*Similarly, for every infinite path $\zeta \in \mathcal{TP}_{C_0,\sigma_0}$ of the form (12), we have $\langle\sigma\rangle\zeta \in \mathcal{IP}_{C_0,\langle\sigma\rangle\sigma_0}$, where the masked path $\langle\sigma\rangle\zeta$ is defined by*

$$\langle\sigma\rangle\zeta \;:=\; C_0, \langle\sigma\rangle\sigma_0 \xrightarrow{L_0} C_1, \langle\sigma\rangle\sigma_1 \xrightarrow{L_1} C_2, \langle\sigma\rangle\sigma_2 \xrightarrow{L_2} \cdots\,,$$

Although we omit the straightforward proof, we remark that the lemma holds because the inequality $\sigma' \sqsubseteq \langle\sigma\rangle\sigma'$ means that transitions of the form $C', \sigma' \xrightarrow{L} C'', \sigma''$ and $C', \sigma' \xrightarrow{L} \tau'$ are preserved by masking; that is, they give rise to transitions $C', \langle\sigma\rangle\sigma' \xrightarrow{L} C'', \langle\sigma\rangle\sigma''$ and $C', \langle\sigma\rangle\sigma' \xrightarrow{L} \langle\sigma\rangle\tau'$ respectively. The same does not apply to error transitions $C', \sigma' \longrightarrow \mathsf{fault}$, because the masked state $\sigma$ may be defined on the variable whose undefinedness in $\sigma'$ triggers the fault.

**Proof of Theorem 6.3.** Suppose $\{\Phi\}C\{\Psi\}$ is partially correct, $\Sigma \models \Phi$ and $C, \Sigma$ is terminating. We have to show that

$$\mathrm{T}_{C,\Sigma}\restriction_{\mathrm{FV}(\Psi)} \;\perp\!\!\!\perp\; \Sigma \circ q_{C,\Sigma} \;\mid\; \Sigma\restriction_{\mathrm{FV}(\Phi)} \circ q_{C,\Sigma}\,.$$

For notational simplicity, define $T := \mathrm{T}_{C,\Sigma}\restriction_{\mathrm{FV}(\Psi)}$, $S := \Sigma \circ q_{C,\Sigma}$ and $R := \Sigma\restriction_{\mathrm{FV}(\Phi)} \circ q_{C,\Sigma}$. Combining (3) and (2), we need to show that, for every $\rho \in \mathsf{Supp}(R)$ and every $\sigma, \sigma' \in \mathsf{Supp}(S_{|R=\rho})$, it holds that $p_{T|(S|R=\rho)=\sigma} = p_{T|(S|R=\rho)=\sigma'}$. By a straightforward property of iterated conditioning, $p_{T|(S|R=\rho)=\sigma} = p_{T|(S,R)=(\sigma,\rho)}$. Also, by definition of $R$ and $S$, for all $\sigma, \rho \in \mathsf{State}$,

$$\rho \in \mathsf{Supp}(R) \wedge \sigma \in \mathsf{Supp}(S_{|R=\rho}) \iff \sigma \in \mathsf{Supp}(S) \wedge \rho = \sigma\restriction_{\mathrm{FV}(\Phi)}\,.$$

So $\sigma$ determines $\rho$, and hence $p_{T|(S,R)=(\sigma,\rho)} = p_{T|(S,R)=(\sigma,\sigma\restriction_{\mathrm{FV}(\Phi)})} = p_{T|S=\sigma}$. Since $\mathsf{Supp}(S) = \mathsf{Supp}(\Sigma)$, it is therefore enough to show:

$$\forall \sigma, \sigma' \in \mathsf{Supp}(\Sigma),\quad \sigma\restriction_{\mathrm{FV}(\Phi)} = \sigma'\restriction_{\mathrm{FV}(\Phi)} \implies p_{T|S=\sigma} = p_{T|S=\sigma'}\,. \tag{26}$$

We approach this in several steps.

Consider any $\sigma \in \mathsf{Supp}(\Sigma)$. Our first goal is to show that $C, \sigma\restriction_{\mathrm{FV}(\Phi)}$ is terminating. By Lemma 6.5, every infinite execution path $\zeta \in \mathcal{IP}_{C,\,\sigma\restriction_{\mathrm{FV}(\Phi)}}$, gives us a corresponding infinite path $\langle\sigma\rangle\zeta \in \mathcal{IP}_{C,\langle\sigma\rangle(\sigma\restriction_{\mathrm{FV}(\Phi)})}$. Then $\langle\sigma\rangle\zeta \in \mathcal{IP}_{C,\sigma}$, because $\langle\sigma\rangle(\sigma\restriction_{\mathrm{FV}(\Phi)}) = \sigma$, by (25). That is, every path in $\mathcal{IP}_{C,\,\sigma\restriction_{\mathrm{FV}(\Phi)}}$ has a matching path in $\mathcal{IP}_{C,\sigma}$. So, by the formula for divergence probabilities (13), we have

$$\mathsf{Pr}[C, \sigma\restriction_{\mathrm{FV}(\Phi)} \text{ diverges}] \;\le\; \mathsf{Pr}[C, \sigma \text{ diverges}]\,.$$

Since $C, \Sigma$ is terminating and $\sigma \in \mathsf{Supp}(\Sigma)$, it holds that $C, \sigma$ is terminating, hence $\mathsf{Pr}[C, \sigma \text{ diverges}] = 0$. Therefore $\mathsf{Pr}[C, \sigma\restriction_{\mathrm{FV}(\Phi)} \text{ diverges}] = 0$. Because $\Sigma \models \Phi$, it holds that $\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Phi$. So, by the safety guarantee from the partial correctness of $\{\Phi\}C\{\Psi\}$, the random configuration $C, \Sigma\restriction_{\mathrm{FV}(\Phi)}$ is fault-free. Since $\sigma\restriction_{\mathrm{FV}(\Phi)} \in \mathsf{Supp}(\Sigma\restriction_{\mathrm{FV}(\Phi)})$, it holds that $C, \sigma\restriction_{\mathrm{FV}(\Phi)}$ is fault-free. Hence, by (14) and the discussion following it, $\mathsf{Pr}[C, \sigma\restriction_{\mathrm{FV}(\Phi)} \text{ terminates}] = 1$. That is, $C, \sigma\restriction_{\mathrm{FV}(\Phi)}$ is terminating.

Since both $C, \sigma$ and $C, \sigma\restriction_{\mathrm{FV}(\Phi)}$ are terminating, they induce probability distributions $p_{C,\sigma}$ and $p_{C,\,\sigma\restriction_{\mathrm{FV}(\Phi)}}$ on their terminal states. Our next goal is to relate these two distributions. By Lemma 6.5, every terminal path $\pi \in \mathcal{TP}_{C,\,\sigma\restriction_{\mathrm{FV}(\Phi)}}$, induces a corresponding terminal path $\langle\sigma\rangle\pi \in \mathcal{TP}_{C,\sigma}$ with $\mathsf{Pr}(\langle\sigma\rangle\pi) = \mathsf{Pr}(\pi)$. Since $C, \sigma\restriction_{\mathrm{FV}(\Phi)}$ is terminating, the probabilities $\mathsf{Pr}(\pi)$ of all $\pi \in \mathcal{TP}_{C,\,\sigma\restriction_{\mathrm{FV}(\Phi)}}$ add up to 1, hence so do

all probabilities $\mathsf{Pr}(\langle\sigma\rangle\pi)$. Thus every terminal path in $\mathcal{TP}_{C,\sigma}$ is necessarily of the form $\langle\sigma\rangle\pi$ for some $\pi \in \mathcal{TP}_{C,\sigma\restriction_{\mathrm{FV}(\Phi)}}$. It follows that the induced probabilities on terminal states, $p_{C,\sigma}$ and $p_{C,\sigma\restriction_{\mathrm{FV}(\Phi)}}$, are related to each other by the pushforward property

$$p_{C,\sigma} \;=\; (\tau \mapsto \langle\sigma\rangle\tau)_!\,(p_{C,\sigma\restriction_{\mathrm{FV}(\Phi)}})\,. \tag{27}$$

We next apply the partial correctness of $\{\Phi\}C\{\Psi\}$ to the random state $\Sigma\restriction_{\mathrm{FV}(\Phi)}$. We have already noted that $\Sigma\restriction_{\mathrm{FV}(\Phi)} \models \Phi$. We have also shown that $C, \sigma\restriction_{\mathrm{FV}(\Phi)}$ is terminating for every $\sigma \in \mathsf{Supp}(\Sigma)$. Since every $\sigma' \in \mathsf{Supp}(\Sigma\restriction_{\mathrm{FV}(\Phi)})$ is of the form $\sigma\restriction_{\mathrm{FV}(\Phi)}$ for $\sigma \in \mathsf{Supp}(\Sigma)$, it follows that the random configuration $C, \Sigma\restriction_{\mathrm{FV}(\Phi)}$ is terminating. So $\mathrm{T}_{C,\,\Sigma\restriction_{\mathrm{FV}(\Phi)}} \models \Psi$ holds, by the partial correctness of $\{\Phi\}C\{\Psi\}$. It follows that the random state $\mathrm{T}_{C,\,\Sigma\restriction_{\mathrm{FV}(\Phi)}}$ is $\mathrm{FV}(\Psi)$-total.

We now finally turn to (26). What we actually prove is

$$\forall\sigma \in \mathsf{Supp}(\Sigma),\quad p_{T|S=\sigma} \;=\; (\tau' \mapsto \tau'\restriction_{\mathrm{FV}(\Psi)})_!\,(p_{C,\,\sigma\restriction_{\mathrm{FV}(\Phi)}})\,, \tag{28}$$

from which (26) follows, since the right-hand side of (28) depends only on $\sigma\restriction_{\mathrm{FV}(\Phi)}$. To establish (28), consider any $\sigma \in \mathsf{Supp}(\Sigma)$. We have $T = \mathrm{T}_{C,\Sigma}\restriction_{\mathrm{FV}(\Psi)} = (\tau' \mapsto \tau'\restriction_{\mathrm{FV}(\Psi)}) \circ \mathrm{T}_{C,\Sigma}$. Also, by the definition of $p_{C,\sigma}$, it holds that $p_{\mathrm{T}_{C,\Sigma}|S=\sigma} = p_{C,\sigma}$. We can thus establish (28) by

$$\begin{aligned}
p_{T|S=\sigma} &= p_{((\tau'\mapsto\tau'\restriction_{\mathrm{FV}(\Psi)})\circ\mathrm{T}_{C,\Sigma})|S=\sigma} \\
&= (\tau' \mapsto \tau'\restriction_{\mathrm{FV}(\Psi)})_!\,(p_{\mathrm{T}_{C,\Sigma}|S=\sigma}) \\
&= (\tau' \mapsto \tau'\restriction_{\mathrm{FV}(\Psi)})_!\,(p_{C,\sigma}) \\
&= (\tau' \mapsto \tau'\restriction_{\mathrm{FV}(\Psi)})_!\,(\tau \mapsto \langle\sigma\rangle\tau)_!\,(p_{C,\sigma\restriction_{\mathrm{FV}(\Phi)}}) && \text{by (27)} \\
&= (\tau \mapsto \tau\restriction_{\mathrm{FV}(\Psi)})_!\,(p_{C,\sigma\restriction_{\mathrm{FV}(\Phi)}})\,,
\end{aligned}$$

where the last step uses the fact, established earlier, that $\mathrm{T}_{C,\,\Sigma\restriction_{\mathrm{FV}(\Phi)}}$ is $\mathrm{FV}(\Psi)$-total. Indeed, this means that every $\tau \in \mathsf{Supp}(p_{C,\sigma\restriction_{\mathrm{FV}(\Phi)}})$ is $\mathrm{FV}(\Psi)$-total. Hence, $\tau\restriction_{\mathrm{FV}(\Psi)} = (\langle\sigma\rangle\tau)\restriction_{\mathrm{FV}(\Psi)}$ holds for such $\tau$, since the masking operation does not change the value of variables defined in $\tau$. □

We end this section with a very simple example illustrating how the safety aspect of partial correctness is essential to the validity of both relative tightness and the frame rule. Consider the specification

$$\{\top\}\; X := X \bmod 2\; \{[X = 0 \vee X = 1]\}\,, \tag{29}$$

where $\top$ is the true assertion. Recall that $[X = 0 \vee X = 1]$ says that the stated property holds with probability 1. Under our definition of partial correctness (Definition 5.1), this specification is not partially correct because it fails the safety guarantee: the empty random state satisfies the precondition, but execution from the empty state faults. If, however, the safety guarantee is dropped from the definition of partial correctness, then the above specification becomes correct. In any random state $\Sigma$ from which the program $X{:=}X \bmod 2$ terminates, the postcondition $\{[X = 0 \vee X = 1]\}$ is indeed true.

We observe that considering (29) as correct violates relative tightness. Let $\Sigma$ be a random state with distribution (expressed as a convex sum of outputs weighted by probabilities)

$$p_\Sigma \;=\; \frac{1}{2}\cdot[X \mapsto 0] + \frac{1}{2}\cdot[X \mapsto 1]\,.$$

Obviously $\Sigma$ satisfies the precondition $\top$. Moreover $X{:=}X \bmod 2$ terminates, when run from $\Sigma$, in the random state

$$\mathrm{T} \;=\; \Sigma \circ q_{X:=X \bmod 2,\,\Sigma}\,,$$

whose distribution is

$$p_{\mathrm{T}} \;=\; \frac{1}{2}\cdot[X\mapsto 0] + \frac{1}{2}\cdot[X\mapsto 1]\,.$$

Since $\mathrm{FV}(\top)=\emptyset$, and $\mathrm{FV}([X=0\vee X=1])=\{X\}$, the property of relative tightness (18) asserts that $\mathrm{T}\perp\!\!\!\perp \Sigma\circ q_{X:=X\ \mathsf{mod}\ 2,\,\Sigma}$, i.e., $\mathrm{T}\perp\!\!\!\perp \mathrm{T}$, which is patently not the case.

For a similar reason, the frame rule also fails if (29) is considered as correct. Consider the specification

$$\{\top * [Y=0\vee Y=1]\}\ X:=X\ \mathsf{mod}\ 2\ \{[X=0\vee X=1]*[Y=0\vee Y=1]\}\,, \tag{30}$$

Let $\Sigma$ be a random state with distribution

$$p_{\Sigma} \;:=\; \frac{1}{2}\cdot[X\mapsto 0, Y\mapsto 0] + \frac{1}{2}\cdot[X\mapsto 1, Y\mapsto 1]$$

Then $\Sigma$ satisfies the precondition $\top * [Y=0\vee Y=1]$. The command $X{:=}X\ \mathsf{mod}\ 2$ again terminates, when run from $\Sigma$, in a random state T with $p_{\mathrm{T}}=p_{\Sigma}$. Because the vaues of $X$ and $Y$ are correlated in the distribution $p_{\mathrm{T}}$, it is not the case that $\mathrm{T}\models [X=0\vee X=1]*[Y=0\vee Y=1]$. So (30) does not hold.

Let us use the same example to compare our frame rule (Figure 3) with the frame rule from the original probabilistic separation logic [3]. The translation of specification (29) into the assertion logic of [3] produces a correct specification according to their interpretation. However, the frame rule from [3] is not applicable, because (29) fails the side-condition that the specification's precondition must imply that the starting state is defined on variables whose initial values are read by the program. It is this side-condition that prevents the incorrect specification (30) from being derived. The frame rule of [3] also has another side-condition, which places a restriction on the variables that are allowed to appear in the specification's postcondition. Neither of the two side-conditions under discussion appears in the frame rule of the present paper (Fig. 3), because the safety guarantee and the derived property of relative tightness render them unnecessary.

Our proof of soundness for the frame rule depends on the property of relative tightness (Theorem 6.3). An analogue of this property is crucial to the proof of soundness for the frame rule in [3]; specifically, a "soundness" property for classes of variables, which says that behaviour of a program $C$ can be described as a probabilistic map from variables whose initial values are read by $C$ to variables that are written to by $C$ (Lemma 6(3) of *op. cit.*). The analogy is that this probabilistic map establishes that a portion of the final state (the write variables) is independent of the input state conditional on the projection of the latter to the read variables. A major difference, however, is that our relative tightness is a derived property of *specifications*, relating to portions of the input and output states determined by the pre- and postcondition respectively; whereas the analogous property in [3] is based on a taxonomy of program variables. One other difference is that our results apply to the full pwhile language with general unbounded while loops. This additional generality necessitates the termination argument that appears in our proof of relative tightness.

## 7 Further work

Having established the soundness of the frame rule, the most pressing question is whether the framework introduced in this paper can serve as the basis for a practicable probabilistic separation logic for verifying interesting probabilistic programs. Given our departure from the standard, resource-monoid-based formulation of probabilistic separation logic, and our relaxation of the restrictions on imperative programs that are built into the design of the proof systems in [3,2] (the special treatment of deterministic variables, and the restrictions to deterministic loop guards and to bounded loops), there is no *a priori* reason to expect that a reasonable such proof system exists. Nevertheless, we are optimistic that one does.

As a first step in this direction, in his undergraduate project work [5], the first author has checked that the example verifications of cryptographic protocols from [3] do transfer to our semantic framework. This has been carried out in a setting in which the aforementioned restrictions on programs have been retained, using proof rules for partial correctness closely modelled on those in *op. cit.*

$$\frac{\{\Phi\}\ \mathsf{if}\ B\ \mathsf{then}\ C\ \mathsf{else}\ \mathsf{skip}\ \{\Phi\}}{\{\Phi\}\ \mathsf{while}\ B\ \mathsf{do}\ C\ \{[\neg B] \wedge \Phi\}}\ \Phi \text{ topologically closed}$$

Fig. 5. A provisional proof rule for general while loops

Finding a more general proof system, catering for unrestricted programs, is an interesting research goal. We believe that our specification framework provides a promising basis for the development of such a system. For example, Figure 5 presents a plausible partial-correctness proof rule for while loops. It has a side condition that the loop invariant $\Phi$ must be *topologically closed*, meaning that, for every sample space $\Omega$, the set $\{\Sigma : \Omega \to \mathsf{State} \mid \Sigma \models \Phi\}$ is closed in the topology of convergence-in-probability of random variables (which, because we are considering random variables valued in a discrete space $\mathsf{State}$, coincides with almost-sure convergence). Regarding the full system we have in mind, we comment that we envisage a crucial interaction between conditioning modalities $\mathbf{C}_E\,\Phi$ in the logic and conditionals in the programming language, and also a conditional generalisation of the frame rule, allowing frame-rule-like inferences to be applied to conditional independence statements. Such a system is the subject of current investigation.

One final point for investigation is the extent to which our classical logic, based on a partial satisfaction relation, forms a sufficiently expressive alternative to probabilistic separation logic based on partial resource monoids. For example, does it support a natural (and useful) version of the separating implication ("magic wand") connective of separation logic?

## Acknowledgement

We thank the anonymous MFPS reviewers for their helpful comments.

## References

[1] Bao, J., S. Docherty, J. Hsu and A. Silva, *A bunched logic for conditional independence*, in: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–14 (2021).
https://doi.org/10.1109/LICS52264.2021.9470712

[2] Bao, J., M. Gaboardi, J. Hsu and J. Tassarotti, *A separation logic for negative dependence*, Proc. ACM Program. Lang. **6** (2022).
https://doi.org/10.1145/3498719

[3] Barthe, G., J. Hsu and K. Liao, *A probabilistic separation logic*, Proc. ACM Program. Lang. **4** (2019).
https://doi.org/10.1145/3371123

[4] Ishtiaq, S. and P. W. O'Hearn, *Bi as an assertion language for mutable data structures*, SIGPLAN Not. **46**, page 84–96 (2011), ISSN 0362-1340.
https://doi.org/10.1145/1988042.1988050

[5] Jereb, J. I., *Verjetnostna separacijska logika*, Diplomsko delo, FRI, University of Ljubljana (2025). In Slovene.

[6] Kozen, D., *Semantics of probabilistic programs*, Journal of Computer and System Sciences **22**, pages 328–350 (1981), ISSN 0022-0000.
https://doi.org/https://doi.org/10.1016/0022-0000(81)90036-2

[7] Li, J. M., A. Ahmed and S. Holtzen, *Lilac: A modal separation logic for conditional probability*, Proc. ACM Program. Lang. **7** (2023).
https://doi.org/10.1145/3591226

[8] O'Hearn, P., J. Reynolds and H. Yang, *Local reasoning about programs that alter data structures*, in: L. Fribourg, editor, *Computer Science Logic*, pages 1–19, Springer Berlin Heidelberg, Berlin, Heidelberg (2001), ISBN 978-3-540-44802-0.
https://doi.org/10.1007/3-540-44802-0_1

[9] O'Hearn, P. W. and D. J. Pym, *The logic of bunched implications*, The Bulletin of Symbolic Logic **5**, pages 215–244 (1999), ISSN 10798986.
http://www.jstor.org/stable/421090

[10] Pym, D. J., *On bunched predicate logic*, in: *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, LICS '99, page 183, IEEE Computer Society, USA (1999), ISBN 0769501583.
https://doi.org/10.1109/LICS.1999.782614

[11] Pym, D. J., P. W. O'Hearn and H. Yang, *Possible worlds and resources: the semantics of bi*, Theoretical Computer Science **315**, pages 257–305 (2004), ISSN 0304-3975. Mathematical Foundations of Programming Semantics.
https://doi.org/https://doi.org/10.1016/j.tcs.2003.11.020

[12] Yang, H. and P. O'Hearn, *A semantic basis for local reasoning*, in: M. Nielsen and U. Engberg, editors, *Foundations of Software Science and Computation Structures*, pages 402–416, Springer Berlin Heidelberg, Berlin, Heidelberg (2002), ISBN 978-3-540-45931-6.
https://doi.org/10.1007/3-540-45931-6_28