Probabilistic Rely-guarantee Calculus

Annabelle McIver[a] Tahiry Rabehaja[a] Georg Struth[b]

[a]*Department of Computing, Macquarie University, Australia*

[b]*Department of Computer Science, University of Sheffield, United Kingdom*

Abstract

Jones' rely-guarantee calculus for shared variable concurrency is extended to include probabilistic behaviours. We use an algebraic approach that is based on a combination of probabilistic Kleene algebra with concurrent Kleene algebra. Soundness of the algebra is shown relative to a general probabilistic event structure semantics. The main contribution of this paper is a collection of rely-guarantee rules built on top of that semantics. In particular, we show how to obtain bounds on probabilities of correctness by deriving quantitative extensions of rely-guarantee rules. The use of these rules is illustrated by a detailed verification of a simple probabilistic concurrent program: a faulty Eratosthenes sieve.

Key words: probabilistic programs, concurrency, rely-guarantee, program verification, program semantics, Kleene algebra, event structures.

1 Introduction

The rigorous study of concurrent systems remains a difficult task due to the intricate interactions and interferences between their components. A formal framework for concurrent systems ultimately depends on the kind of concurrency considered. Jones' rely-guarantee calculus provides a mathematical foundation for proving the correctness of programs with shared variables concurrencyin compositional fashion [1]. This paper extends Jones' calculus to the quantitative correctness of probabilistic concurrent programs.

This research was supported by an iMQRES from Macquarie University, the ARC Discovery Grant DP1092464 and the EPSRC Grant EP/J003727/1.

Email addresses: `annabelle.mciver@mq.edu.au` (Annabelle McIver), `tahiry.rabehaja@mq.edu.au` (Tahiry Rabehaja), `g.struth@sheffield.ac.uk` (Georg Struth).

Probabilistic programs have become popular due to their ability to express quantitative rather than limited qualitative properties. Probabilities are particularly important for protocols that rely on the unpredictability of probabilistic choices. The sequential probabilistic semantics, originating with Kozen [2] and Jones [3], have been extended with nondeterminism [4,5], to yield methods for quantitative reasoning based on partial orders.

We aim to obtain similar methods for reasoning in compositional ways about probabilistic programs with shared variable concurrency. In algebraic approaches, compositionality arises quite naturally through congruence or monotonicity properties of algebraic operations such as sequential and concurrent composition or probabilistic choice.

It is well known that compositional reasoning is nontrivial both for concurrent and for sequential probabilistic systems. In the concurrent case, the obvious source of non-compositionality is communication or interaction between components. In the rely-guarantee approach, interference conditions are imposed between individual components and their environment in order to achieve compositionality. Rely conditions account for the global effect of the environment's interference with a component; guarantee conditions express the effect of a particular component on the environment. Compositionality is then obtained by considering rely conditions within components and guarantee conditions within the environment.

In the presence of probabilistic behaviours, a problem of congruence (and hence non-compositionality) arises when considering the natural extension of trace-based semantics to probabilistic automata [6], where a standard workaround is to define a partial order based on simulations.

In this paper, we define a similar construct to achieve compositionality. However, simulation-based equivalences are usually too discriminating for program verification. Therefore, we also use a weaker semantics that is essentially based on sequential behaviours. Such a technique has been motivated elsewhere [7], where the sequential order is usually not a congruence. Therefore, the simulation-based order is used for properties requiring composition while the second order provides a tool that captures the sequential behaviours of the system.

Concurrent Kleene algebra [8,7] provides an algebraic account of Jones' rely-guarantee framework. Algebras provide an abstract view of a program by focusing more on control flows rather than data flows. All the rely-guarantee rules described in [8,7] were derived by equational reasoning from a finite set of algebraic axioms. Often, the verification of these axioms on an intended semantics is easier than proving the inference rules directly in that semantics. Moreover, every structure satisfying these laws will automatically incorporate a direct interpretation of the rely-guarantee rules, as well as additional rules that can be used for program refinement. Therefore, we also adopt an algebraic approach to the quantitative extension of rely-guarantee, that is, we establish some basic algebraic properties of a concrete event structure model and derive the rely-guarantee rules by algebraic reasoning.

In summary, the main contribution of this paper is the development of a mathematical foundation for *probabilistic rely-guarantee calculi*. The inference rules are expressed algebraically, and we illustrate their use on an example based on the Sieve of Eratosthenes which incorporates a probability of failure. We also outline two rules that provide probabilistic lower bounds for the correctness of the concurrent execution of multiple components.

A short summary of the algebraic approach to rely-guarantee calculus and the extension to probabilistic programs are found respectively in Section 2 and 5-6. Section 3 and 4 are devoted to the construction of a denotational model for probabilistic concurrent programs. Section 7 closes this paper with a detailed verification of the faulty Eratosthenes sieve.

2 Non-probabilistic rely-guarantee calculus

The rely-guarantee approach, originally put forward by Jones [1], is a compositional method for developing and verifying large concurrent systems. An algebraic formulation of the approach has been proposed recently in the context of concurrent Kleene algebras [8]. In a nutshell, a bi-Kleene algebra is an algebraic structure $(K, +, \cdot, \|, 0, 1, ^*, ^{(*)})$ such that $(K, +, \cdot, 0, 1, ^*)$ is a Kleene algebra and $(K, +, \|, 0, 1, ^{(*)})$ is a commutative Kleene algebra. The axioms of Kleene algebra and related structures are in Appendix A .

Intuitively, the set K models the actions a system can take; the operation $(+)$ corresponds to the nondeterministic choice between actions, $(\cdot)$ to their sequential composition and $(\|)$ to their parallel or concurrent composition. The constant 0, the unit of addition, models the abortive action, 1, the unit of sequential and concurrent composition, the ineffective action `skip`. The operation $(^*)$ is a sequential finite iteration of actions; the corresponding parallel finite iteration operation $(^{(*)})$ is not considered further in this article. Two standard models of bi-Kleene algebras are languages, with $(+)$ interpreted as language union, $(\cdot)$ as language product, $(\|)$ as shuffle product, 0 as the empty language, 1 as the empty word language and $(^*)$ as the Kleene star, and pomset languages under operations similarly to those in Section 4.3 below (cf. [9]).

Language-style models with interleaving or shuffle also form the standard semantics of rely-guarantee calculi. In that context, traces are typically of the form $(s_1, s'_1), (s_2, s'_2) \dots (s_k, s'_k)$, where the s_i and s'_i denote states of a system, pairs (s_i, s'_i) correspond to internal transitions of a component, and fragments $s'_i), (s_{i+1}$ to transitions caused by interferences of the environment. Behaviours of a concurrent system are associated with sets of such traces.

With semantics for concurrency in mind, a generalised encoding of the validity of Hoare triples becomes useful:

$$\{P\}S\{Q\} \Leftrightarrow P{\cdot}S \leq Q,$$

where $P \leq Q \Leftrightarrow P{\cup}Q = Q$. It has been proposed originally by Tarlecki [10] for sequential programs with a relational semantics. In contrast to Hoare's standard approach, where P and Q are assertions and S a program, all three elements are now allowed to be programs. In the context of traces, $\{P\}S\{Q\}$ holds if all traces that are initially in P and then in S are also in Q. This comprises situations where program P models traces ending in a set of states p (a precondition) and Q models traces ending in a set of states q (a postcondition). The Hoare triple then holds if all traces establishing precondition p can be extended by program S to traces establishing postcondition q, whenever y terminates, as in the standard interpretation. We freely write $\{p\}S\{q\}$ in such cases. It turns out that all the inference rules of Hoare logic except the assignment rule can be derived in the setting of Kleene algebra [8].

For concurrency applications, the algebraic encoding of Hoare triples has been expanded to Jones quintuples $\{P\ R\}S\{G\ Q\}$, also written $R, G \vdash \{P\}S\{Q\}$, with respect to rely conditions R and guarantee conditions G [8]. The basic intuition is as follows. A rely condition R is understood as a special program that constrains the behaviour of a component S by executing it in parallel as $R\|S$. This is consistent with the above trace interpretation where parallel composition is interpreted as shuffle and gaps in traces correspond to interferences by the environment. Typical properties of relies are $1 \leq R$ (where 1 is `skip`) and $R^* = R{\cdot}R = R\|R = R$. Moreover, relies distribute over nondeterministic choices as well as sequential and concurrent compositions: $R\|(S{+}T) = R\|S{+}R\|T$, $R\|(S{\cdot}T) = (R\|S){\cdot}(R\|T)$ and $R\|(S\|T) = (R\|S)\|(R\|T)$, hence they apply to all subcomponents of a given component [7]. A guarantee G of a given component S is only constrained by the fact that it should include all behaviours of S, that is, $S \leq G$.

Consequently, a Jones quintuple is valid if the component S constrained by the rely satisfies the Hoare triple—the relationship between precondition and postcondition—and the guarantee includes all behaviours of S [8]:

$$\{P\ R\}S\{G\ Q\} \Leftrightarrow \{P\}R\|S\{Q\} \wedge S \leq G. \qquad (1)$$

The rules of Hoare logic without the assignment axiom are still derivable from the axioms of bi-Kleene algebra, when Hoare triples are replaced by Jones quintuples [8]. To derive the standard rely-guarantee concurrency rule, one can expand bi-Kleene algebra by a meet operation ($\sqcap$) and assume that $(K, +, \sqcap)$ forms a distributive lattice [7]. Then

$$\frac{\{P\ R\}S\{G\ Q\} \quad \{P\ R'\}S'\{G'\ Q'\} \quad G \leq R' \quad G' \leq R}{\{P\ R{\sqcap}R'\}S\|S'\{G{+}G'\ Q{\sqcap}Q',\}}. \tag{2}$$

This inference rule demonstrates how the rely-guarantee specifications of components can be composed into a rely-guarantee specification of a larger system. If S and S' satisfy the premises, then $S\|S'$ satisfies both postconditions Q and Q' when run in an environment satisfying both relies R and R'. Moreover, $S\|S'$ guarantees either of G or G'.

Deriving these inference rules from the algebraic axioms mentioned makes them sound with respect to all models of these axioms, including trace-based semantics with parallel composition interpreted as interleaving, and true-concurrency semantics such as pomset languages and the event structures considered in this article. Without the algebraic layer, Dingel [11] and Coleman and Jones [12] have already proved the soundness of rely-guarantee rules with respect to trace-based semantics, more precisely *Aczel traces* [13]. This paper follows previous algebraic developments, but for probabilistic programs.

In Section 5, we provide a suitable extension of the rely-guarantee formalism, in particular Rule (2), to probabilistic concurrent programs. The soundness of such a formalism is shown relative to a semantic space that allows sequential probabilistic programs to include concurrent behaviours.

3 Sequential probabilistic programs

We start by giving a brief summary of the denotation of sequential probabilistic programs using the powerdomain construction of McIver and Morgan [5]. All probabilistic programs are considered to have a finite state space denoted by Ω. A distribution over the set Ω is a function $\mu{:}\Omega{\to}[0, 1]$ such that $\sum_{s\in\Omega} \mu(s){=}1$. The set of distributions over Ω is denoted by $\mathbb{D}\Omega$. Since Ω is a finite set, we identify a distribution with the associated measure. For every $\mu{\in}\mathbb{D}\Omega$ and $O{\subseteq}\Omega$, we write $\mu(O){=}\sum_{s\in O} \mu(s)$. An example of distribution is the point distribution δ_s, centred at the state $s{\in}\Omega$, such that

$$\delta_s(s') = \begin{cases} 1 & \text{if } s{=}s', \\ 0 & \text{otherwise.} \end{cases}$$

A (nondeterministic) probabilistic program r modelled as a map of type $\Omega{\to}\mathbb{P}\mathbb{D}\Omega$ such that $r(s)$ is a non-empty, topologically closed and convex subset of $\mathbb{D}\Omega$ for every state $s{\in}\Omega$. The set $\mathbb{D}\Omega$ is a topological sub-space of the finite product $\mathbb{R}^\Omega$ (endowed with the usual product topology), and the topological closure is considered with respect to the induced topology on Ω [1] . We denote by $\mathbb{H}_1\Omega$ the set of probabilistic programs that terminate almost certainly. Notice that the set $\mathbb{D}\Omega$ contains only distributions instead of the subdistributions considered by McIver and Morgan [5]. Therefore, we only model nondeterministic programs that are terminating with probability 1.

Programs in $\mathbb{H}_1\Omega$ are ordered by pointwise inclusion, i.e. $r \sqsubseteq_{\mathbb{H}} r'$ if for every $s{\in}\Omega$, $r(s) \subseteq r(s')$. A program r is deterministic if, for every s, $r(s) = \{\mu_s\}$ (i.e. a singleton) for some distribution $\mu_s{\in}\mathbb{D}\Omega$. The set of deterministic programs is denoted by $\mathbb{J}_1\Omega$ (as in Jones' spaces [3]). If $f{\in}\mathbb{J}_1\Omega$ is a deterministic program such that $f(s) = \{\mu_s\}$, then we usually just write $f(s) = \mu_s$. A particularly useful example of a probabilistic deterministic program is the ineffectual program `skip`, which we denote by δ. Thus $\delta(s) = \{\delta_s\}$.

Let $p \in [0,1]$. The probabilistic combination of two probabilistic programs r and r' is defined as ([5, Def. 5.4.5])

$$(r{\oplus_p}r')(s) = \{\mu{\oplus_p}\mu' \mid \mu{\in}r(s) \wedge \mu'{\in}r'(s)\}, \tag{3}$$

where $(\mu{\oplus_p}\mu')(s) = (1{-}p)\mu(s){+}p\mu'(s)$ for every state $s{\in}\Omega$. Thus, the program r (resp. r') is executed with probability $1{-}p$ (resp. p).

Nondeterminism is obtained as the set of all probabilistic choices ([5, Def. 5.4.6]), that is,

$$(r{+}r')(s) = \cup_{p\in[0,1]}(r{\oplus_p}r')(s). \tag{4}$$

The sequential composition of r by r' is defined as ([5, Def. 5.4.7]):

$$(r{\cdot}r')(s) = \{\, f{\star}\mu |\, f{\in}\mathbb{J}_1\Omega \wedge \mu{\in}r(s) \wedge f \sqsubseteq_{\mathbb{H}} r'\} \tag{5}$$

where

$$(f{\star}\mu)(s') = \sum_{s''\in\Omega} f(s'')(s')\mu(s'')$$

for every state $s'{\in}\Omega$.

For $r, r'{\in}\mathbb{H}_1\Omega$, the binary Kleene star $r{*}r'$ is the least fixed point of the function $f_{r,r'}(X) = r'{+}r{\cdot}X$ in $\mathbb{H}_1\Omega$. It has been shown in [5] that the function $r' \mapsto r{\cdot}r'$ is continuous —it preserves directed suprema. Notice that a topological closure is sometimes needed to ensure that we obtain an element of $\mathbb{H}_1\Omega$. Hence, the Kleene star $r{*}r'$ is the program such that $r{*}r'(s) = \overline{\cup_n f^n_{r,r'}(\bot)(s)}$, where $\overline{A}$ is the topological closure of the set $A \subseteq \mathbb{D}\Omega$ and the constant $\bot$ is

[1] These healthiness conditions are set out and fully explained in the work of McIver and Morgan [5].

defined, as usual, such that $r''{\cdot}\bot{=}\bot{\cdot}r''{=}\bot$, $\bot{+}r'' = r''$ and $\bot \sqsubseteq_{\mathbb{H}} r''$ for every $r''{\in}\mathbb{H}_1\Omega{\cup}\{\bot\}$.

We introduce tests, which are used for conditional constructs, following the idea adopted in various algebras of programs. We define a test to be a map $b : \Omega \to \mathbb{PD}\Omega$ such that $b(s) \subseteq \{\delta_s\}$. Indeed, an "if statement" is modelled algebraically as $b{\cdot}r{+}(\neg b){\cdot}r'$ where $(\neg b)(s) = \emptyset$ if the test underlying b holds at state s and it is $\{\delta_s\}$ otherwise. The sub-expression $b{\cdot}r(s)$ still evaluates to $\emptyset$ if $b(s)$ is empty, but care should be taken to avoid expressions such as $r{\cdot}b$ (if f is a deterministic refinement of b, then $f(s'')(s')$ may have no meaning if $b(s''){=}\emptyset$). A test that is always false can be identified with $\bot$.

We denote by $\overline{\mathbb{H}}_1\Omega$ the set of tests together with the set of probabilistic programs. The refinement order $\sqsubseteq_{\mathbb{H}}$ is extended to $\overline{\mathbb{H}}_1\Omega$ in a straightforward manner. For every test b, we have $b \sqsubseteq_{\mathbb{H}} \delta$; hence, we refer to tests as *subidentities*. Every elements of $\overline{\mathbb{H}}_1\Omega$ are called programs, unless otherwise specified.

4 An event structures model for probabilistic concurrent programs

The set $\mathbb{H}_1\Omega$ of probabilistic programs provides a full semantics for program constructs such as (probabilistic) assignments, probabilistic choices, conditionals and while loops that terminate almost surely. Unfortunately, it is impossible to define the concurrent composition of two sequential programs as an operation on $\mathbb{H}_1\Omega$ because the result would always be a sequential program. Thus we are forced to look for a more general framework in order to formally model concurrency. Fortunately, there are several suitable mathematical models that allows the formal verification of programs with concurrent behaviours. A powerful example that accounts for true concurrency are Winskel's *event structures* [14,15]. In this section, we outline a denotational semantics for probabilistic concurrent programs based on Langerak's bundle event structures [16], which have been extended successfully to quantitative features [17,18,19]. This construction is necessary to ensure the soundness of the extended rely-guarantee formalism.

A bundle event structure comprises events ranging over some set E of *events* as its fundamental objects. Intuitively, an event is an occurrence of an action at a certain moment in time. Thus an action can be repeated, but each of its occurrences is associated with a unique event. Events are (partially) ordered by a causality relation which we denote by $\mapsto$: if an event e'' causally depends on either e or e' (i.e. $\{e, e'\}{\mapsto}e''$) then either e or e' must have happened before e'' can happen or is *enabled*. The relationship between e and e' is called *conflict*, written $e\#e'$, because both events cannot occur simultaneously.

In general, the conflict relation # is a binary relation on E. Given two subsets $x, x' \subseteq E$, the predicate $x\#x'$ holds iff for every $(e, e')\in x\times x'$ such that $e\neq e'$, we have $e\#e'$.

Definition 4.1 *A quintuple* $\mathcal{E} = (E, \mapsto, \#, \lambda, \mathbf{\Phi})$ *is a* bundle event structure with internal probability *(i.e. an ipBES) if*

- $\#$ *is an irreflexive symmetric binary relation on* E*, called* conflict relation*.*
- $\mapsto\subseteq \mathbb{P}E\times E$ *is a* bundle relation*, i.e. if* $x\mapsto e$ *for some* $x \subseteq E$ *and* $e\in E$*, then* $x\#x$*.*
- $\lambda{:}E\to\overline{\mathbb{H}}_1\Omega$*, i.e. it labels events with (atomic) probabilistic programs.*
- $\mathbf{\Phi} \subseteq \mathbb{P}E$ *such that* $x\#x$ *holds for every* $x\in\mathbf{\Phi}$*.*

The finite state space Ω *of the programs used as labels is fixed.*

The intuition behind this definition is that events are occurrences of atomic program fragments, i.e. they can happen without interferences from an environment. Hence, we need to distinguish all atomic program fragments when translating a program into a bundle event structure. Atomic programs can be achieved by creating a construct that forces atomicity. Examples of such a technique include "atomic brackets" [20]. In this paper, we always state which actions are atomic rather than using such a device.

Given an ipBES $\mathcal{E}$, a *finite trace* of $\mathcal{E}$ is a sequence of events $e_1e_2\dots e_n$ such that for all different $1 \leq i, j \leq n$, $\neg(e_i\#e_j)$ and if $j{=}i{+}1$ then there exists an $x \subseteq E$ such that $x\mapsto e_j$ and $e_i\in x$ [16,18,21]. In other words, a trace is safe (an event may occur only when it is enabled) and is conflict free. The set of all finite traces of $\mathcal{E}$ is denoted by $\mathcal{T}(\mathcal{E})$. The set of maximal traces of $\mathcal{E}$ (w.r.t the prefix ordering) is denoted $\mathcal{T}_{\max}(\mathcal{E})$. We simply write $\mathcal{T}$ (resp. $\mathcal{T}_{\max}$) instead of $\mathcal{T}(\mathcal{E})$ (resp. $\mathcal{T}_{\max}(\mathcal{E})$) when no confusion may arise.

The aim of this section is to elaborate two relationships between the sets of traces of given event structures. The first comparison is based on a sequential reduction using schedulers; the second one is simulation. We will show that the sequential comparison is strictly weaker than the simulation relation.

4.1 Schedulers on ipBES

As in the case of automata, we define schedulers on ipBES in order to obtain a sequential equivalence on bundle event structures with internal probability. Intuitively, a scheduler reduces an ipBES to a element of $\overline{\mathbb{H}}_1\Omega$. While the technicalities of the schedulers we define in this paper is tailored towards a rely-guarantee reasoning, there might be relationships with previous works [22,23] where schedulers (and associated testing theories) are restricted in order achieve a broader class observationally equivalent processes.

A *subdistribution* is a map $\mu : \Omega \to [0,1]$ such that $\sum_{s\in\Omega} \mu(s) \leq 1$. The set of subdistributions over Ω is denoted by $\mathbb{D}_{\leq 1}\Omega$.

Definition 4.2 *A* scheduler σ *on an ipBES* $\mathcal{E}$ *is a map*

$$\sigma{:}\mathcal{T}{\to}[(E{\times}\Omega){\rightharpoonup}\mathbb{D}_{\leq 1}\Omega]$$

such that for all $\alpha{\in}\mathcal{T}$*:*

(1) $\mathrm{dom}(\sigma(\alpha)) = \{(e,s) \mid \alpha e{\in}\mathcal{T} \wedge s{\in}\Omega\}$,
(2) there exists a function $w{:}E{\times}\Omega{\to}[0,1]$ *such that, for every* $(e,s){\in}\mathrm{dom}(\sigma(\alpha))$, $\sigma(\alpha)(e,s) = w(e,s)\mu$ *for some* $\mu{\in}\lambda(e)(s)$*.*
(3) for every $s{\in}\Omega$*, we have* $\sum_{(e,s)\in\mathrm{dom}(\sigma(\alpha))} w(e,s) = 1$,
(4) for every $(e,s){\in}\mathrm{dom}(\sigma(\alpha))$*, if* $\lambda(e)(s){=}\emptyset$*, then* $w(e,s){=}0$ *and* $\sigma(\alpha)(e,s){=}0$ *(the subdistribution that evaluates to* 0 *everywhere).*

The set of all schedulers on $\mathcal{E}$ *is denoted by* $\mathbf{Sched}(\mathcal{E})$*.*

Property 1 says that we may schedule an event provided it does not depend on unscheduled events.

Property 2 states that, given a trace α, the scheduler will resolve the nondeterminism between events enabled after α byusing the weight function w. This may include immediate conflicts or interleavings of concurrent events. Moreover, the scheduler has access to the current program state when resolving that nondeterminism. This means that $w(e,s)$ is the probability that the event e is scheduled, knowing that the program state is s. If the event e is successfully scheduled, then the scheduler performs a last choice of distribution, say μ from $\lambda(e)(s)$, to generate the next state of the program.

Property 3 ensures that when the state s is known, then the choice between the events, enabled after the trace α, is indeed probabilistic.

Property 4 says that a scheduler is forced to choose events whose labels do not evaluate to the empty set at the current state of the program. This is particularly important when the program contains conditionals and the label of an event is a test. A scheduler is forced to choose the branch whose test holds. If two tests hold at state s, then a branch is chosen probabilistically using the weight function w.

The motivation behind Property 4 is to ensure that, for every trace α such that $\mathrm{dom}(\sigma(\alpha)){\neq}\emptyset$, and every state $s{\in}\Omega$, we have

$$\sum_{(e,s)\in\mathrm{dom}(\sigma(\alpha))} \sigma(\alpha)(e,s){\in}\mathbb{D}\Omega,$$

hence that sum is indeed a distribution. To ensure that a scheduler satisfying that condition can be constructed, we restrict ourselves to *feasible* event structures. Given an element $r{\in}\overline{\mathbb{H}}_1\Omega$, we write $\mathrm{dom}(r){=}\{s \mid r(s){\neq}\emptyset\}$.

Definition 4.3 *An ipBES $\mathcal{E}$ is* feasible *if for every trace $\alpha{\in}\mathcal{T}\backslash\mathcal{T}_{\max}$, we have $\cup_{\alpha e\in\mathcal{T}}\mathrm{dom}(\lambda(e)){=}\Omega$.*

A consequence of this assumption is that an "if clause" always needs to have a corresponding "else clause".

Example 4.4 *Let us consider the program $r{\cdot}(\delta{+}r)$. In this program, r is atomic deterministic (such as an assignment to a variable) and the associated event structure has three events:*

$$\mathcal{E} = (\{e_r, e'_r, e_\delta\}, \{\{e_r\}{\mapsto}e_\delta, \{e_r\}{\mapsto}e'_r\}, \{e_\delta \# e_{r_2}\}, \{(e_r, r), (e'_r, r), (e_\delta, \delta)\}, \mathbf{\Phi}),$$

where $\mathbf{\Phi} = \{\{e'_r, e_\delta\}\}$ (see Sec. 4.3 for an inductive construction of ipBES from primitive blocks). This event structure is feasible and a scheduler σ on $\mathcal{E}$ is characterised by a weight function $w{:}\{e_r, e'_r, e_\delta\}{\times}\Omega{\to}[0,1]$ resolving the choice $\delta{+}r$. In fact, for every fixed state $s{\in}\Omega$, we have $\sigma(e_r)(e_\delta, s){=}w(e_\delta, s)\delta_s$ and $\sigma(e_r)(e'_r, s){=}w(e'_r, s)r(s)$ and $w(e_\delta, s){+}w(e'_r, s){=}1$.

4.2 Generating sequential probabilistic programs from ipBES and schedulers

Similar to the case of probabilistic automata [6], our scheduler resolves branching as encoded in the conflict relation of an event structure. In addition, a scheduler also "flattens" concurrency into interleaving by choosing an enabled event according to the associated weight function. The flattening of concurrent behaviours is sound because actions labelling events are assumed atomic and we are using schedulers to generate sequential behaviours from an ipBES. True concurrency is accounted for in Sec. 4.4.

Let $\sigma{\in}\mathbf{Sched}(\mathcal{E})$ and $s{\in}\Omega$ be an initial state. We inductively construct a sequence of functions φ_n that map a trace in $\mathcal{T}$ to a subdistribution on Ω according to σ and s. Intuitively, if $\alpha{\in}\mathcal{T}$, then $\varphi_n(\alpha){\in}\mathbb{D}_{\leq 1}\Omega$ is the sequential composition of the n-first probabilistic actions labelling events in α applied to the initial state s. This yields a subdistribution because α is weighted with respect to the scheduler σ. The sequence of partial functions $\varphi_n{:}\mathcal{T}{\rightharpoonup}\mathbb{D}_{\leq 1}\Omega$ is the *computation sequence* of $\mathcal{E}$ with respect to σ from initial state s.

Formally, for each $n{\in}\mathbb{N}$, we have $\mathrm{dom}(\varphi_n){=}\cup_{k\leq n}\mathcal{T}_k$, where $\mathcal{T}_n$ is the set of traces of length n and

(1) $\varphi_0(\emptyset){=}\delta_s$,where s is the initial state,

(2) if $\alpha e \in \mathcal{T}_{n+1}$ then

$$\varphi_{n+1}(\alpha e)(s) = \sum_{t\in\Omega} [\sigma(\alpha)(e,t)(s)] \varphi_n(\alpha)(t)$$

and $\varphi_{n+1}(\alpha e) = \varphi_n(\alpha e)$ otherwise.

To emphasises that this computation function refers to a specific initial state $t\in\Omega$ we sometimes write $\varphi_{n,t}$ instead of φ_n.

The *complete run* of $\mathcal{E}$ with respect to σ is the limit φ of that sequence, i.e. $\varphi = \cup_n \varphi_n$, which exists because φ_n defines a sequence of partial functions such that φ_n is the restriction of φ_{n+1} to $\mathrm{dom}(\varphi_n)$. Since we consider finite traces only, we have $\mathrm{dom}(\varphi) = \mathcal{T}$. The *sequential behaviour* of $\mathcal{E}$ with respect to σ from the initial state s is defined by the sum

$$\sigma_s(\mathcal{E}) = \sum_{\alpha\in\mathcal{T}_{\max}} \varphi(\alpha).$$

Proposition 4.5 *For every bundle event structure $\mathcal{E}$, scheduler $\sigma \in \mathbf{Sched}(\mathcal{E})$ and initial state s, $\sigma_s(\mathcal{E})$ is a subdistribution.*

Proof. Let φ be the complete run of $\mathcal{E}$ with respect to a given scheduler σ. We show by induction on n that

$$\mu_n(\Omega) = \sum_{\alpha\in\mathcal{T}_n \cup (\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n))} \varphi(\alpha)(\Omega) = \sum_{t\in\Omega} \sum_{\alpha\in\mathcal{T}_n \cup (\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n))} \varphi(\alpha)(t) = 1.$$

For the base case $n=0$, we have $\mu_0(\Omega) = \varphi(\emptyset)(\Omega) = \delta_s(\Omega) = 1$, where s is the initial state. Assume the induction hypothesis $\mu_n(\Omega) = 1$. We have

$$\begin{aligned}
\mu_{n+1}(\Omega) &= \sum_{\alpha\in\mathcal{T}_{n+1} \cup (\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_{n+1}))} \varphi(\alpha)(\Omega) \\
&=^{\dagger} \sum_{\alpha\in\mathcal{T}_{n+1}} \varphi(\alpha)(\Omega) + \sum_{\alpha\in\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n)} \varphi(\alpha)(\Omega) \\
&= \sum_{\alpha e\in\mathcal{T}_{n+1}} \sum_{t\in\Omega} \sigma(\alpha)(e,t)(\Omega)\varphi(\alpha)(t) + \sum_{\alpha\in\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n)} \varphi(\alpha)(\Omega) \\
&= \sum_{\alpha\in\mathcal{T}_n\setminus\mathcal{T}_{\max}} \sum_{\alpha e\in\mathcal{T}} \sum_{t\in\Omega} \sigma(\alpha)(e,t)(\Omega)\varphi(\alpha)(t) + \sum_{\alpha\in\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n)} \varphi(\alpha)(\Omega) \\
&= \sum_{\alpha\in\mathcal{T}_n\setminus\mathcal{T}_{\max}} \left[\sum_{(e,t)\in \mathrm{dom}(\sigma(\alpha))} \sigma(\alpha)(e,t)(\Omega) \right] \varphi(\alpha)(t) + \sum_{\alpha\in\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n)} \varphi(\alpha)(\Omega) \\
&=^{\ddagger} \sum_{\alpha\in\mathcal{T}_n\setminus\mathcal{T}_{\max}} \varphi(\alpha)(\Omega) + \sum_{\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n)} \varphi(\alpha)(\Omega) \\
&= \mu_n(\Omega) = 1.
\end{aligned}$$

(†) Follows from $\mathcal{T}_{n+1} \cup (\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_{n+1})) = \mathcal{T}_{n+1} \cup (\mathcal{T}_{\max}\cap \mathrm{dom}(\varphi_n))$ and the fact that the second union is disjoint.

(‡) The square-bracketed term equals 1 because of Properties 2 and 3 of the scheduler σ.

Therefore, each partial computation φ_n can be seen as a probability distribution $\varphi_n(-)(\Omega)$ supported on $\mathcal{T}_n \cup (\mathcal{T}_{\max} \cap \mathrm{dom}(\varphi_n))$. Hence, the limit is a subdistribution $\varphi(-)(\Omega)$ on $\mathcal{T}_{\max}$. It does not necessarily add up to 1 because elements of $\mathcal{T}_{\max}$ are finite maximal traces only and non-termination will decrease that quantity (we assume that the empty sum is 0. This occurs when there are no maximal traces). $\square$

Given a state $t \in \Omega$, $\sigma_s(\mathcal{E})(t)$ is the probability that the concurrent probabilistic program denoted by $\mathcal{E}$ terminates in state t when conflicts (resp. concurrent events) are resolved (resp. interleaved) according to the scheduler σ. Since we consider terminating programs only, we denote by $\mathbf{Sched}_1(\mathcal{E})$ the set of schedulers of $\mathcal{E}$ such that, for every initial state s, $\sigma_s(\mathcal{E})$ is a distribution. A scheduler in $\mathbf{Sched}_1(\mathcal{E})$ generates a sequential behaviour that terminates almost surely. This leads to our definition of a bracket $[\![\ ]\!]$ that transform each feasible ipBES to an element of $\mathbb{H}_1\Omega$:

$$[\![\mathcal{E}]\!](s) = \overline{\mathrm{conv}\{\sigma_s(\mathcal{E}) \mid \sigma \in \mathbf{Sched}_1(\mathcal{E})\}}$$

where $\mathrm{conv}(A)$ (resp. $\overline{A}$) is the convex (resp. topological) closure of the set of distributions A in $\mathbb{R}^\Omega$.

Definition 4.6 *Let $\mathcal{E}, \mathcal{F}$ be two feasible event structures. We say that $\mathcal{E}$ (sequentially) refines $\mathcal{F}$, denoted by $\mathcal{E} \sqsubseteq \mathcal{F}$, if $[\![\mathcal{E}]\!] \sqsubseteq_{\mathbb{H}} [\![\mathcal{F}]\!]$ holds in $\mathbb{H}_1\Omega$.*

The relation $\sqsubseteq$ is a preorder on ipBES. Whilst this order is not a congruence, it is used to specify the desired sequential properties of a feasible event structure $\mathcal{E}$ with $\mathbf{Sched}_1(\mathcal{E}) \neq \emptyset$. We will show that feasibility and non-emptiness of $\mathbf{Sched}_1$ are preserved by the regular operations of the next section (Props 4.8 and 4.15).

4.3 Regular operations on ipBES

This section provides interpretations of the operations $(+, \cdot, *, \|)$ and constants $0, 1$ on event structures with disjoint sets of events. These definitions allow the inductive translation of program texts into event structure objects.

- The algebraic constant 1 is interpreted as $(e, \emptyset, \emptyset, \{(e, \delta)\}, \{e\})$.
- The algebraic constant 0 is interpreted as $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$.
- Each atomic action $r \in \overline{\mathbb{H}}_1\Omega$ is associated with $(\{e\}, \emptyset, \emptyset, \{(e, r)\}, \{e\})$. This event structure is again denoted by r.

- The nondeterministic choice between the event structures $\mathcal{E}$ and $\mathcal{F}$ is constructed as

$$\mathcal{E}+\mathcal{F} = (E \cup F, \#_{\mathcal{E}+\mathcal{F}}, \mapsto_{\mathcal{E}} \cup \mapsto_{\mathcal{F}}, \lambda_{\mathcal{E}} \cup \lambda_{\mathcal{F}}, \{x \cup y \mid x \in \mathbf{\Phi}_{\mathcal{E}} \wedge y \in \mathbf{\Phi}_{\mathcal{F}}\})$$

  where $\#_{\mathcal{E}+\mathcal{F}} = [\cup_{x \in \mathbf{\Phi}_{\mathcal{E}} \wedge y \mathbf{\Phi}_{\mathcal{F}}} \mathrm{sym}(x \times y)] \cup \#_{\mathcal{E}} \cup \#_{\mathcal{F}} \cup \mathrm{sym}(\mathbf{in}(\mathcal{E}) \times \mathbf{in}(\mathcal{F}))$ and sym is the symmetric closure of a relation on $E \cup F$. The square-bracketed set ensures that every final event in $\mathcal{E}$ is in conflict with every final event in $\mathcal{F}$. This ensures that, if $z \in \mathbf{\Phi}_{\mathcal{E}+\mathcal{F}}$, then $z \# z$.
- The sequential composition of $\mathcal{E}$ by $\mathcal{F}$ is

$$\mathcal{E} \cdot \mathcal{F} = (E \cup F, \#_{\mathcal{E}} \cup \#_{\mathcal{F}}, \mapsto_{\mathcal{E}} \cup \mapsto_{\mathcal{F}} \cup \{x \mapsto e \mid e \in \mathbf{in}(\mathcal{F}) \wedge x \in \mathbf{\Phi}_{\mathcal{E}}\}, \lambda_{\mathcal{E}} \cup \lambda_{\mathcal{F}}, \mathbf{\Phi}_{\mathcal{F}}).$$

- The concurrent composition of $\mathcal{E}$ and $\mathcal{F}$ is

$$\mathcal{E} \| \mathcal{F} = (E \cup F, \#_{\mathcal{E}} \cup \#_{\mathcal{F}}, \mapsto_{\mathcal{E}} \cup \mapsto_{\mathcal{F}}, \lambda_{\mathcal{E}} \cup \lambda_{\mathcal{F}}, \mathbf{\Phi}_{\mathcal{E}} \cup \mathbf{\Phi}_{\mathcal{F}}).$$

- The binary Kleene star of $\mathcal{E}$ and $\mathcal{F}$ is the supremum of the sequence

$$\mathcal{F}, \mathcal{F}+\mathcal{E} \cdot \mathcal{F}, \mathcal{F}+\mathcal{E} \cdot (\mathcal{F}+\mathcal{E} \cdot \mathcal{F}), \ldots$$

  of bundle event structures with respect to the ω-complete sub-BES order [24].

Example 4.7 *Let us consider the sequential programs $r, \delta \in \mathbb{H}_1\Omega$. A concurrent program that is skipping or running r in parallel with itself is algebraically denoted by $(r\|r)+1$. The construction of the associated event structure starts from the innermost operation $(r\|r)$, assuming that each occurrence of the atomic action r is associated with an event from $\{e_r, e'_r\}$. Thus*

$$\mathcal{E}_{r\|r} = (\{e_r, e'_r\}, \emptyset, \emptyset, \underbrace{\{(e_r, r), (e'_r, r)\}}_{\lambda_{r\|r}}, \{\{e_r\}, \{e'_r\}\}).$$

We can now construct the nondeterministic choice between $r\|r$ and δ as

$$\mathcal{E}_{(r\|r)+1} = (\{e_r, e'_r, e_\delta\}, \{e_r \# e_\delta, e'_r \# e_\delta\}, \emptyset, \lambda_{r\|r} \cup \{(e_\delta, \delta)\}, \{\{\epsilon, e_\delta\} \mid \epsilon \in \{e_r, e'_r\}\}).$$

In this example, we have $e_r \# e_\delta$ and $e'_r \# e_\delta$ but e_r and e'_r are concurrent.

For every bundle event structure $\mathcal{E}$, $0+\mathcal{E}=\mathcal{E}$, $0 \cdot \mathcal{E}=\mathcal{E} \cdot 0=\mathcal{E}$, and in particular, $0 \cdot 1=1$. The constant 0 was only introduced to have a bottom element on the set of bundle event structures with internal probabilities. It ensures that we can compute the Kleene star inductively from the least element. Moreover, 0 will disappear in mixed expressions because of these properties.

We now show that the operations $(+)$ and $(\cdot)$ are preserved by the map $[\![\ ]\!]$. The case of the binary Kleene star $(*)$ is proven in Prop. 4.15.

Proposition 4.8 *For $\mathcal{E},\mathcal{F}$ non-zero, feasible and terminating event structures, we have $[\![\mathcal{E}{+}\mathcal{F}]\!] = [\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!]$ and $[\![\mathcal{E}{\cdot}\mathcal{F}]\!] = [\![\mathcal{E}]\!]{\cdot}[\![\mathcal{F}]\!]$.*

Proof. For the case of nondeterminism (+), let $s{\in}\Omega$ be the initial state and $\mu{\in}[\![\mathcal{E}{+}\mathcal{F}]\!](s)$. Let us firstly assume that $\mu{=}\sigma_s(\mathcal{E})$ for some $\sigma{\in}\mathbf{Sched}_1(\mathcal{E}{+}\mathcal{F})$. By definition of the sum $\mathcal{E}{+}\mathcal{F}$, the set of events E and F are disjoints, so we can define two schedulers $\sigma^{\mathcal{E}}{\in}\mathbf{Sched}_1(\mathcal{E})$ and $\sigma^{\mathcal{F}}{\in}\mathbf{Sched}_1(\mathcal{F})$ as follows. Let $\alpha{\in}\mathcal{T}(\mathcal{E}{+}\mathcal{F})$ and $(e,t){\in}\mathrm{dom}(\sigma(\alpha))$, we define

$$\sigma^{\mathcal{E}}(\alpha)(e,t) = \begin{cases} \sigma(\alpha)(e,t) & \text{if } \alpha{\in}\mathcal{T}(\mathcal{E})\backslash\{\emptyset\}, \\ \frac{\sigma(\emptyset)(e,t)}{p_t^{\mathcal{E}}} & \text{if } \alpha{=}\emptyset. \end{cases}$$

where $p_t^{\mathcal{E}}{=}\sum_{e'\in\mathbf{in}(\mathcal{E})} w(e,t)$, w is the weight function associated to σ at the trace $\emptyset$ and s is the initial state. The real number $p_t^{\mathcal{E}}$ is just a normalisation constant required by Property 3 in the definition of schedulers. [2] The scheduler $\sigma^{\mathcal{F}}$ is similarly defined. It follows directly from these definition of $\sigma^{\mathcal{E}}$ and $\sigma^{\mathcal{F}}$ that $\sigma(\emptyset)(e,t){=}p_t^{\mathcal{E}}\sigma(\emptyset)(e,t){+}p_t^{\mathcal{F}}\sigma(\emptyset)(e,t)$ where $p_t^{\mathcal{E}}{+}p_t^{\mathcal{F}}{=}1$ because of Property 3. Hence, $\sigma_s(\mathcal{E}){=}p_s^{\mathcal{E}}\sigma_s^{\mathcal{E}}(\mathcal{E}){+}p_s^{\mathcal{F}}\sigma_s^{\mathcal{F}}(\mathcal{F})$ i.e. $\sigma_s(\mathcal{E}){\in}[\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!]$. Since $[\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!]$ is convex and topologically closed, we deduce that $[\![\mathcal{E}{+}\mathcal{F}]\!](s) \subseteq ([\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!])(s)$.

For the converse inclusion $([\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!])(s){\subseteq}[\![\mathcal{E}{+}\mathcal{F}]\!](s)$, notice that $\overline{\mathrm{conv}(A)} = \mathrm{conv}(\overline{A})$ holds for every subset $A{\subseteq}\mathbb{R}^{\Omega}$. If we write $A{=}\{\sigma_s(\mathcal{E}) \mid \sigma{\in}\mathbf{Sched}_1(\mathcal{E})\}$ and $B{=}\{\sigma_s(\mathcal{F}) \mid \sigma{\in}\mathbf{Sched}_1(\mathcal{F})\}$, then

$$([\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!])(s) = \overline{\mathrm{conv}(\overline{\mathrm{conv}(A)}{\cup}\overline{\mathrm{conv}(B)})} = \overline{\mathrm{conv}(A{\cup}B)}.$$

But it is clear that $A{\subseteq}[\![\mathcal{E}{+}\mathcal{F}]\!](s)$ (a scheduler that does not choose $\mathcal{F}$ is possible because $\mathcal{E}$ is feasible) and $B{\subseteq}[\![\mathcal{E}{+}\mathcal{F}]\!](s)$. Therefore, $([\![\mathcal{E}]\!]{+}[\![\mathcal{F}]\!])(s) = \overline{\mathrm{conv}(A{\cup}B)}{\subseteq}[\![\mathcal{E}{+}\mathcal{F}]\!](s)$ because the last set is convex and topologically closed.

The sequential composition is proven using a similar reasoning. Let $\mathcal{E},\mathcal{F}$ be two bundle event structures satisfying the hypothesis, and $\mu{\in}[\![\mathcal{E}{\cdot}\mathcal{F}]\!](s)$ for some initial state $s{\in}\Omega$.

The proof of $[\![\mathcal{E}{\cdot}\mathcal{F}]\!](s) \subseteq [\![\mathcal{E}]\!]{\cdot}[\![\mathcal{F}]\!](s)$ goes as follows. Firstly, let us assume that there is a scheduler σ on $\mathcal{E}{\cdot}\mathcal{F}$ such that $\mu{=}\sigma_s(\mathcal{E}{\cdot}\mathcal{F})$. Since schedulers are inductively constructed, there exists $\sigma^{\mathcal{E}}{\in}\mathbf{Sched}(\mathcal{E})$ and $\sigma^{\mathcal{F}}{\in}\mathbf{Sched}(\mathcal{F})$ such that

$$\sigma(\alpha)(e,t) = \begin{cases} \sigma^{\mathcal{E}}(\alpha)(e,t) & \text{if } \alpha e{\in}\mathcal{T}(\mathcal{E}), \\ \sigma^{\mathcal{F}}(\alpha'')(e,t) & \text{if } \alpha{=}\alpha'\alpha'' \text{ and } (\alpha',\alpha''){\in}\mathcal{T}_{\max}(\mathcal{E}){\times}\mathcal{T}(\mathcal{F}). \end{cases}$$

[2] If $p_t^{\mathcal{E}}{=}0$, then $\sigma{\in}\mathbf{Sched}_1(\mathcal{F})$.

Let us denote by φ_n and $\varphi_n^{\mathcal{E}}$ (resp. $\varphi_{n,t}^{\mathcal{F}}$)[3] the computation sequences associated to the respective schedulers σ and $\sigma^{\mathcal{E}}$ (resp. $\sigma^{\mathcal{F}}$) from the initial state s (resp. t). It follows directly that $\varphi_n(\alpha)=\varphi_n^{\mathcal{E}}(\alpha)$ for every $\alpha\in\mathcal{T}_n(\mathcal{E})$. If $\alpha'\in\mathcal{T}_{\max}(\mathcal{E})\cap\mathcal{T}_n(\mathcal{E})$ and $e\in\mathbf{in}(\mathcal{F})$ then, for every state $u\in\Omega$,

$$\varphi_{n+1}(\alpha' e)(u) = \sum_{t\in\Omega} \sigma^{\mathcal{F}}(\emptyset)(e,t)(u)\varphi^{\mathcal{E}}(\alpha')(t).$$

Similarly, we have

$$\begin{aligned}\varphi_{n+1}(\alpha' e e')(u) &= \sum_{t'\in\Omega} \sigma^{\mathcal{F}}(e)(e,t')(u)\left[\sum_{t\in\Omega}\sigma^{\mathcal{F}}(\emptyset)(e,t)(t')\varphi^{\mathcal{E}}(\alpha')(t)\right]\\ &= \sum_{t\in\Omega}\left[\sum_{t'\in\Omega}\sigma^{\mathcal{F}}(e)(e,t')(u)\sigma^{\mathcal{F}}(\emptyset)(e,t)(t')\right]\varphi^{\mathcal{E}}(\alpha')(t)\\ &= \sum_{t\in\Omega}\varphi_{2,t}^{\mathcal{F}}(u)\varphi^{\mathcal{E}}(\alpha')(t).\end{aligned}$$

By simple induction on the length of α'', we deduce that

$$\varphi(\alpha'\alpha'')(u) = \sum_{t\in\Omega}\varphi_t^{\mathcal{F}}(\alpha'')(u)\varphi^{\mathcal{E}}(\alpha')(t),$$

where $\varphi_t^{\mathcal{F}}$ is the complete run obtained from the sequence $\varphi_{n,t}^{\mathcal{F}}$. It follows by definition of the sequential composition on $\mathbb{H}_1\Omega$ (Eqn. (5)) that

$$\sigma_s(\mathcal{E})(u) = \sum_{t\in\Omega}\sigma_t^{\mathcal{F}}(\mathcal{F})(u)\sigma_s^{\mathcal{E}}(\mathcal{E})(t)\in[\![\mathcal{E}]\!]\cdot[\![\mathcal{F}]\!](s)$$

for every state $u\in\Omega$. Secondly, since $[\![\mathcal{E}]\!]\cdot[\![\mathcal{F}]\!](s)$ is upclosed and topologically closed, we deduce that $[\![\mathcal{E}\cdot\mathcal{F}]\!](s)\subseteq[\![\mathcal{E}]\!]\cdot[\![\mathcal{F}]\!](s)$.

Conversely, if $\mu\in[\![\mathcal{E}]\!]\cdot[\![\mathcal{F}]\!](s)$, then either $\mu(u) = \sum_{t\in\Omega}\sigma_t^{\mathcal{F}}(\mathcal{F})(u)\sigma_s^{\mathcal{E}}(\mathcal{E})(t)$ or μ is in the closure of the set of these distributions. Either way, the closure properties of $[\![\mathcal{E}\cdot\mathcal{F}]\!](s)$ implies that $[\![\mathcal{E}]\!]\cdot[\![\mathcal{F}]\!](s)\subseteq[\![\mathcal{E}\cdot\mathcal{F}]\!](s)$. $\square$

4.4 Simulation for ipBES

The partial order defined in Definition 4.6 compares the sequential behaviours of two systems. However, it suffers from a congruence problem, i.e. there exist programs $\mathcal{E},\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{E}\sqsubseteq\mathcal{F}$ but $\mathcal{E}\|\mathcal{G}\not\sqsubseteq\mathcal{F}\|\mathcal{G}$. A known technique for achieving congruence is to construct an order based on simulations, which is the subject of this section. We use a similar technique in this subsection.

[3] Remind that $\varphi_{n,t}$ is the computation function computed given the initial state t.

We say that a trace α is *weakly maximal* if it is maximal or there exist some events $e_1, \ldots, e_n$ such that $\alpha e_1 \cdots e_n \in \mathcal{T}_{\max}$ and $\delta \sqsubseteq_{\mathbb{H}} \lambda(e_i)$ for every $1 \leq i \leq n$.

Definition 4.9 *A function $f{:}\mathcal{T}(\mathcal{E}) \rightarrow \mathcal{T}(\mathcal{F})$ is called a* t-simulation *if the following conditions hold:*

- *if $f(\emptyset) = \emptyset$ and $f^{-1}(\beta)$ is a finite set for every $\beta \in \mathcal{T}(\mathcal{F})$,*
- *if $\alpha e \in \mathcal{T}(\mathcal{E})$ then either:*
  · *$f(\alpha e) = f(\alpha)$ and $\lambda(e) \sqsubseteq_{\mathbb{H}} \delta$ holds in $\mathbb{H}_1\Omega$,*
  · *or there exists an event e' of $\mathcal{F}$ such that $\lambda(e) \sqsubseteq_{\mathbb{H}} \lambda(e')$ and $f(\alpha e) = f(\alpha)e'$.*
- *if αe is maximal in $\mathcal{T}(\mathcal{E})$ then $f(\alpha e) = f(\alpha)e'$, for some e' (with $\lambda(e) \sqsubseteq_{\mathbb{H}} \lambda(e')$), and $f(\alpha e)$ is weakly maximal in $\mathcal{T}(\mathcal{F})$* [4] *.*

We say that $\mathcal{E}$ is simulated by $\mathcal{F}$, written $\mathcal{E} \sqsubseteq_{\mathrm{sim}} \mathcal{F}$, if there exists a simulation from $\mathcal{E}$ to $\mathcal{F}$. The equivalence generated by this preorder is denoted $\equiv_{sim}$.

The notion of t-simulation has been designed to simulate event structures correctly in the presence of tests. For instance, given a test b, the simulation $\delta \sqsubseteq_{\mathrm{sim}} (b + \neg b)$ fails because a t-simulation is a total function and it does not allow the removal of "internal" events labelled with subidentities during a refinement step. The finiteness condition on $f^{-1}(\beta)$ ensures that we do not refine a terminating specification with a diverging implementation. Without that constraint, we would be able to write the refinement

$$\texttt{if } (0{=}1) \texttt{ then } s{:=}0 \texttt{ else}[\texttt{if } (0{=}1) \texttt{ then } s{:=}0 \texttt{ else}[\ldots]] \sqsubseteq_{\mathrm{sim}} s{:=}0.$$

However, this should not hold because the left hand sides is a non-terminating program and cannot refine the terminating assignment s :=0.

A t-simulation is used to compare bundle event structures without looking in details at the labels of events. It can be seen as a refinement order on the higher level structure of a concurrent program. Once a sequential behaviour has to be checked, we use the previously defined functional equivalence on event structures with internal probabilities.

Example 4.10 *Consider a program variable x of type Boolean (with value 0 or 1). A t-simulation from $(x{=}1) + (x{\neq}1) \cdot (x{:=}1)$ to $1 + (x{:=}0 \sqcap 1)$* [5] *is given by*

[4] If $f(\alpha)$ is maximal then α is necessarily maximal.
[5] $x{:=}0 \sqcap 1$ is an atomic nondeterministic-assignment, it cannot be interfered with.

the dotted arrow in the following diagram:

$$\begin{tikzcd}
 & \emptyset \arrow[dl] \arrow[dr] \arrow[rrrr, dotted] & & & & \emptyset \arrow[dl] \arrow[dr] & \\
e_{x=1} \arrow[rrrr, dotted] & & e_{x\neq 1} \arrow[d] \arrow[urrr, dotted] & & e_{\delta} & & e_{x:=0\sqcap 1} \\
 & & e_{x\neq 1}e_{x:=1} \arrow[urrrr, dotted] & & & &
\end{tikzcd}$$

This t-simulation refines two nondeterministic choices, one at the program structure level and the other at the atomic level.

Proposition 4.11 *The t-simulation relation $\sqsubseteq_{\mathrm{sim}}$ is a preorder.*

Proof. Reflexivity follows from the identity function and transitivity is obtained by composing t-simulations which will generate a new t-simulation. Notice that care should be taken with respect to the third property of a t-simulation. If $f{:}\mathcal{T}(\mathcal{E}){\to}\mathcal{T}(\mathcal{F})$, $g{:}\mathcal{T}(\mathcal{F}){\to}\mathcal{T}(\mathcal{G})$ are t-simulations, $\alpha e{\in}\mathcal{T}_{\max}(\mathcal{E})$ and $\lambda(e){\sqsubseteq_{\mathbb{H}}}\delta$, then $f(\alpha e){=}f(\alpha)e'$ for some e' of $\mathcal{F}$ such that $\lambda(e){\sqsubseteq_{\mathbb{H}}}\lambda(e')$. If $\lambda(e'){\sqsubseteq_{\mathbb{H}}}\delta$, then it is possible that $g(f(\alpha)e'){=}g(f(\alpha))$. However, since $f(\alpha)e'$ is weakly maximal, $g(f(\alpha)e')$ is also weakly maximal and we can find an event $e''{\in}G$ such that $f(\alpha)e''$ is weakly maximal and $\lambda(e'){\sqsubseteq_{\mathbb{H}}}\lambda(e'')$. We then map αe to $g(f(\alpha))e''$ in the t-simulation from $\mathcal{E}$ to $\mathcal{F}$. □

Proposition 4.12 *If $\mathcal{E},\mathcal{F},\mathcal{G}$ are ipBES, then*

$$\mathcal{E}\|\mathcal{F} \equiv_{\mathrm{sim}} \mathcal{F}\|\mathcal{E}, \tag{6}$$
$$\mathcal{E}\|(\mathcal{F}\|\mathcal{G}) \equiv_{\mathrm{sim}} (\mathcal{E}\|\mathcal{F})\|\mathcal{G}, \tag{7}$$
$$\mathcal{E}{*}\mathcal{F} \equiv_{\mathrm{sim}} \mathcal{F}{+}\mathcal{E}{\cdot}(\mathcal{E}{*}\mathcal{F}), \tag{8}$$
$$\mathcal{E} \sqsubseteq_{\mathrm{sim}} \mathcal{F} \Rightarrow \mathcal{G}{+}\mathcal{E} \sqsubseteq_{\mathrm{sim}} \mathcal{G}{+}\mathcal{F}, \tag{9}$$
$$\mathcal{E} \sqsubseteq_{\mathrm{sim}} \mathcal{F} \Rightarrow \mathcal{G}{\cdot}\mathcal{E} \sqsubseteq_{\mathrm{sim}} \mathcal{G}{\cdot}\mathcal{F}, \tag{10}$$
$$\mathcal{E} \sqsubseteq_{\mathrm{sim}} \mathcal{F} \Rightarrow \mathcal{E}\|\mathcal{G} \sqsubseteq_{\mathrm{sim}} \mathcal{F}\|\mathcal{G}. \tag{11}$$

Proof. The constructions $\mathcal{E}\|\mathcal{F}$ and $\mathcal{F}\|\mathcal{E}$ result in the same event structure and similarly for the associativity.

The Unfold Eqn. (8) is clear because the left and right hand side event structures are exactly the same up to renaming of events.

Implication (9) follows by considering the function $\mathrm{id}_{\mathcal{T}(\mathcal{G})}{\cup}f{:}\mathcal{T}(\mathcal{G}{+}\mathcal{E}){\to}\mathcal{T}(\mathcal{G}{+}\mathcal{F})$. It is indeed a function because the sets of events G and E (resp. F) are disjoint. The property of a t-simulation follows directly because the set of traces $\mathcal{T}(\mathcal{G}{+}\mathcal{E})$ is the disjoint union $\mathcal{T}(\mathcal{G}){\cup}\mathcal{T}(\mathcal{E})$ (similarly for $\mathcal{G}{+}\mathcal{F}$).

For case of sequential composition (10), let f be a t-simulation from $\mathcal{E}$ to $\mathcal{F}$.

It is clear that the function $g{:}\mathcal{T}(\mathcal{G}{\cdot}\mathcal{E}){\to}\mathcal{T}(\mathcal{G}{\cdot}\mathcal{F})$, such that $g(\alpha){=}\alpha|_G f(\alpha|_E)$ is a t-simulation.

For the Implication (11), let $f{:}\mathcal{T}(\mathcal{E}){\to}\mathcal{T}(\mathcal{F})$ be a t-simulation. Let us construct a t-simulation $g{:}\mathcal{T}(\mathcal{E}\|\mathcal{G}){\to}\mathcal{T}(\mathcal{F}\|\mathcal{G})$ inductively. We set $g(\emptyset){=}\emptyset$. Let $\alpha{\in}\mathcal{T}(\mathcal{E}\|\mathcal{G})$ and $e{\in}E{\cup}G$ such that αe is a trace of $\mathcal{E}\|\mathcal{G}$. We write $\alpha|_E$ the restriction of α to the events occurring in $\mathcal{E}$. The inductive definition of g is:

$$g(\alpha e) = \begin{cases} g(\alpha)e \text{ if } e{\in}G, \\ \ g(\alpha) \ \text{ if } e{\in}E \text{ and } f(\alpha|_E e){=}f(\alpha|_E), \\ g(\alpha)e' \text{ if } e{\in}E \text{ and } f(\alpha|_E e){=}f(\alpha|_E)e'. \end{cases}$$

Since the set of events of $\mathcal{E}$ and $\mathcal{G}$ are disjoint, the cases in the above definition of g are disjoint. That is, g is indeed a function and it satisfies the second property of a t-simulation. The last property is clear because if αe is maximal in $\mathcal{T}(\mathcal{E}\|\mathcal{G})$, then either $\alpha|_E$ is maximal in $\mathcal{E}$ and $\alpha|_G e$ is maximal in $\mathcal{T}(\mathcal{G})$, or $\alpha|_E e$ is maximal in $\mathcal{T}(\mathcal{E})$ and $\alpha|_G$ is maximal in $\mathcal{T}(\mathcal{G})$. In both cases, $g(\alpha e){=}g(\alpha)e'$ for some $e'{\in}E{\cup}G$ and $g(\alpha e)$ is weakly maximal in $\mathcal{T}(\mathcal{F}\|\mathcal{G})$. $\square$

We now state the main result of this section, which is the backbone of our probabilistic rely-guarantee calculus.

Theorem 4.13 *Let $\mathcal{E}$ and $\mathcal{F}$ be feasible and terminating ipBES. Then $\mathcal{E} \sqsubseteq_{\rm sim} \mathcal{F}$ implies $\mathcal{E} \sqsubseteq \mathcal{F}$.*

Proof. Let f be a t-simulation from $\mathcal{E}$ to $\mathcal{F}$, $s{\in}\Omega$ be the initial state, $\sigma{\in}\mathbf{Sched}_1(\mathcal{E})$ and φ is the complete run of σ on $\mathcal{E}$ from s. We have to generate a scheduler $\tau{\in}\mathbf{Sched}_1(\mathcal{F})$ such that the measures $\sigma_s(\mathcal{E})$ and $\tau_s(\mathcal{F})$ are equal i.e. they produce the same value for every state $u{\in}\Omega$.

For every $\beta{\in}\mathcal{T}(\mathcal{F})$, we define $f^{-1}_{\min}(\beta)$ to be the set of minimal traces in $f^{-1}(\beta)$, that is,

$$f^{-1}_{\min}(\beta) = \{\alpha \mid \forall e{\in}E : \alpha{=}\alpha' e{\in}f^{-1}(\beta) \Rightarrow \alpha'{\notin}f^{-1}(\beta)\}.$$

We now construct the scheduler τ. Let $\beta{\in}\mathcal{T}(\mathcal{F})$. We consider two cases:

- If $f^{-1}(\beta) = \emptyset$ then we set $\tau(\beta)(e,t) = 0{\in}\mathbb{D}_{\le 1}\Omega$, except for some particular maximal traces that are handled in (†) below.
- Otherwise, given a state $t{\in}\Omega$, we define a normalisation factor

$$C_{\beta,t} = \sum_{\alpha \in f^{-1}_{\min}(\beta)} \varphi(\alpha)(t),$$

and we set [6]

$$\tau(\beta)(e,t) = \frac{1}{C_{\beta,t}}\left(\sum_{\alpha\in f_{\min}^{-1}(\beta)} \varphi(\alpha)(t) \sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)} \prod_{i=1}^{k} w_{i-1}(e_i,t)\mu_k\right)$$

where $w_{i-1}(e_i,t)$ is the weight function such that $\sigma(\alpha e_1\cdots e_{i-1})(e_i,t) = w_{i-1}(e_i)\mu$, and $\mu\in\lambda(e_i)$ (if $\lambda(e_i)(t)$ is empty then $w_{i-1}(e_i,t)=0$). The distribution μ_k is chosen by σ from $\lambda(e_k)(t)$, when scheduling e_k.

Firstly, we show that τ is indeed a scheduler on $\mathcal{F}$. The Property(1) of Definition 4.2 is clear. Let us show the other properties. Let $\beta e\in\mathcal{T}(\mathcal{E})$ and let $W{:}E\times\Omega\to\mathbb{R}$ be the weight function such that

$$W(e,t) = \frac{1}{C_{\beta,t}} \sum_{\alpha\in f_{\min}^{-1}(\beta)} \varphi(\alpha)(t) \sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)} \prod_{i=1}^{k} w_{i-1}(e_i,t).$$

Indeed, $\mu=\frac{\tau(\beta)(e,t)}{W(e,t)}$ is in $\lambda(e)(t)$ [7] because $\lambda(e)(t)$ is convex and for each $\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)$ and $\mu_k\in\lambda(e_k)(t)\subseteq\lambda(e)(t)$. Hence $\tau(\beta)(e) = W_e f_e$ and τ satisfies the Property (2) of Def. 4.2. As for Property (3), let $s\in\Omega$ and let us compute the quantity

$$V(t) = \sum_{(e,t)\in\mathrm{dom}(\tau(\beta))} W(e,t),$$

for a fixed $t\in\Omega$. Let us write $\mathrm{dom}(\beta) = \{e \mid \beta e\in\mathcal{T}(\mathcal{F})\}$.

$$\begin{aligned}
V(t) &= \sum_{(e,t)\in\mathrm{dom}(\tau(\beta))} \frac{1}{C_{\beta,t}} \sum_{\alpha\in f_{\min}^{-1}(\beta)} \varphi(\alpha)(t) \sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)} \prod_{i=1}^{k} w_{i-1}(e_i,t)\\
&= \frac{1}{C_{\beta,t}} \sum_{\alpha\in f_{\min}^{-1}(\beta)} \varphi(\alpha)(t) \sum_{(e,t)\in\mathrm{dom}(\tau(\beta))} \sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)} \prod_{i=1}^{k} w_{i-1}(e_i,t)\\
&= \frac{1}{C_{\beta,t}} \sum_{\alpha\in f_{\min}^{-1}(\beta)} \varphi(\alpha)(t) \sum_{\alpha e_1\cdots e_k\in\cup_{e\in\mathrm{dom}(\beta)} f_{\min}^{-1}(\beta e)} \prod_{i=1}^{k} w_{i-1}(e_i,t).
\end{aligned}$$

From the second to the third expression, the two rightmost sums were merged into a single one because $f_{\min}^{-1}(\beta e)\cap f^{-1}(\beta e') = \emptyset$ (f is a function). It follows

[6] Notice if $C_{\beta,t}=0$ for some $t\in\Omega$ then $\varphi(\alpha)(t)=0$ for every $\alpha\in f_{\min}^{-1}(\beta)$. In other words, none of these α will be scheduled at all. Hence, β need not be scheduled either.

[7] The case $W(e,t)=0$ can be adapted easily because the numerator in the definition of $\tau(\beta)(e)$ is also 0. For instance, we can assume that $\frac{0}{0}=1$.

$$\begin{tikzcd}
 & \alpha \arrow[dl, "w_0(e'_1,t)"'] \arrow[dr, "w_0(e_1,t)"] \arrow[rrr, dotted, two heads] & & & \beta \arrow[d, "W(e,t)"] \\
\alpha e'_1 \arrow[rrrr, dotted, two heads, bend right=10] & & \alpha e_1 \arrow[dl, "w_1(e'_2,t)"'] \arrow[dr, "w_1(e_2,t)"] \arrow[urr, dotted, two heads, bend right=10] \arrow[rr, dotted, two heads] & & e \\
 & \alpha e_1 e'_2 \arrow[urrr, dotted, two heads, bend right=10] & & \alpha e_1 e_2 \arrow[ur, dotted, two heads, bend right=10] &
\end{tikzcd}$$

We have $V(t) = w_0(e'_1, t) + w_0(e_1, t)w_1(e'_2, t)w_0(e_1, t)w_1(e_2, t) = 1$ because $w_1(e'_2, t) + w_1(e_2, t) = 1$ and $w_0(e'_1, t) + w_0(e_1, t) = 1$ (Def. 4.2 Property (3)).

Fig. 1. An example showing that $V(t) = 1$.

from Property (3), applied on the weight $w_{i-1}(e_i, t)$ of σ, that

$$\sum_{\alpha e_1 \cdots e_k \in \cup_{e \in \mathrm{dom}(\beta)} f^{-1}_{\min}(\beta e)} \prod_{i=1}^{k} w_{i-1}(e_i, s) = 1$$

and hence $V = 1$ (c.f. Figure 1 for a concrete example). The last Property (4) of Def. 4.2 is clear because if $\lambda(e)(t) = \emptyset$, then the coefficient of $\sigma(\alpha e_1 \cdots e_{k-1})(e_k, t)$ is 0 because $\lambda(e_k)(t) = \emptyset$. Hence, the product is also 0.

Secondly, let ψ be the complete run of $\mathcal{F}$ with respect to τ. We now show by induction on β that

$$\psi(\beta) = \sum_{\alpha \in f^{-1}_{\min}(\beta)} \varphi(\alpha) = C_{\beta,t}, \tag{12}$$

where the empty sum evaluates to the identically zero distribution. The base case is clear because $\psi(\emptyset){=}\delta_s{=}\phi(\emptyset)$ where s is the initial state. Let us assume the above identity for $\beta{\in}\mathcal{T}(\mathcal{F})$ and let $e{\in}F$ such that $\beta e{=}\mathcal{T}(\mathcal{E})$ and

$f_{\min}^{-1}(\beta e)\neq\emptyset$. By definition of ψ, if $u\in\Omega$, we have:

$$\begin{aligned}
\psi(\beta e)(u) &= \sum_{t\in\Omega}\frac{1}{C_{\beta,t}}\sum_{\alpha\in f_{\min}^{-1}(\beta)}\varphi(\alpha)(t)\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\prod_{i=1}^{k} w_{i-1}(e_i,t)\mu_k(u)\psi(\beta)(t)\\
&=\sum_{t\in\Omega}\sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\prod_{i=1}^{k} w_{i-1}(e_i,t)\mu_k(u)\varphi(\alpha)(t)\\
&=\sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\sum_{t\in\Omega}\prod_{i=1}^{k} w_{i-1}(e_i,t)\mu_k(u)\varphi(\alpha)(t)\\
&=\sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\sum_{t\in\Omega}\sum_{t'\in\Omega} w_0(e_1,t')\delta_{t'}(t)\left[\prod_{i=2}^{k} w_{i-1}(e_i,t)\mu_k(u)\right]\varphi(\alpha)(t')\\
&=\sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\sum_{t\in\Omega}\prod_{i=2}^{k} w_{i-1}(e_i,t)\mu_k(u)\varphi(\alpha e_1)(t)\\
&=\sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\sum_{t\in\Omega}\sum_{t'\in\Omega} w_1(e_2,t')\delta_{t'}(t)\left[\prod_{i=3}^{k} w_{i-1}(e_i,t)\mu_k(u)\right]\varphi(\alpha e_1)(t')\\
&=\cdots.
\end{aligned}$$

By continuing the above reasoning for all e_i (induction), $i\leq k-1$, we obtain

$$\begin{aligned}
\psi(\beta e)(u) &= \sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\sum_{t\in\Omega} w_{k-1}(e_k,t)\mu_k(u)\varphi(\alpha e_1\cdots e_{k-1})(t)\\
&=\sum_{\alpha\in f_{\min}^{-1}(\beta)}\sum_{\alpha e_1\cdots e_k\in f_{\min}^{-1}(\beta e)}\varphi(\alpha e_1\cdots e_k)(u).
\end{aligned}$$

Hence,

$$\psi(\beta e)(u)=\sum_{\alpha'\in f_{\min}^{-1}(\beta e)}\varphi(\alpha')(u).$$

(†) We finally compute the sum $\tau_s(\mathcal{F})=\sum_{\beta\in\mathcal{T}_{\max}(\mathcal{F})}\psi(\beta)$. Notice firstly that τ may not schedule some traces of $\mathcal{F}$. In particular, the third property in the definition of simulation implies that a maximal element of $\mathcal{T}(\mathcal{E})$ may be mapped to a weakly maximal element of $\mathcal{T}(\mathcal{F})$. Hence, we need to extend the scheduler τ so that it is non-zero for exactly one maximal element from that weakly maximal trace. More precisely, if $\beta'=f(\alpha)$ is weakly maximal for some maximal trace $\alpha\in\mathcal{T}_{\max}(\mathcal{E})$, then there exists a sequence $e_1,\ldots,e_n$ such that $\beta=\beta' e_1\cdots e_n\in\mathcal{T}_{\max}(\mathcal{F})$ and $\delta\sqsubseteq_{\mathbb{H}}\lambda(e_i)$. We extend τ such that $\tau(\beta' e_1\cdots e_i)(e_{i+1},t)=\delta_t$. This implies that $\psi(\beta)(t)=\psi(\beta')(t)$. The other case is that β is maximal and belongs to the image of f. In both cases, we have

$$\psi(\beta)(t)=\sum_{\alpha\in A_\beta}\varphi(\alpha)(t),$$

where $A_\beta=f_{\min}^{-1}(\beta)$ if β is in the image of f, or $A_\beta=f_{\min}^{-1}(\beta')$ if there is such a β' as above, otherwise, $A_\beta=\emptyset$. Thus, A_β contains maximal traces only (if it is not empty). Since, f is a total function, the set $\{A_\beta \mid \beta{\in}\mathcal{T}_{\max}(\mathcal{F})\}$ is a partition of $\mathcal{T}_{\max}(\mathcal{E})$ and we have

$$\sum_{\beta\in\mathcal{T}_{\max}(\mathcal{E})} \psi(\beta)(t) = \sum_{\beta\in\mathcal{T}_{\max}(\mathcal{E})} \sum_{\alpha\in A_\beta} \varphi(\alpha)(t) = \sum_{\alpha\in\mathcal{T}_{\max}(\mathcal{E})} \varphi(\alpha)(t),$$

i.e. we obtain $\tau_s(\mathcal{F}) = \sigma_s(\mathcal{E})$. □

Example 4.14 *Reconsider the t-simulation of Example 4.10. By definition, the unique scheduler σ on $(x{=}1){+}(x{\neq}1){\cdot}(x{:=}1)$ satisfies:*

- $\sigma(\emptyset)(e_{x=t}, s) = w(e_{x=t}, s)\delta_t$ *where* $w(e_{x=t}, s) = \begin{cases} 1 & \textit{if } s{=}t, \\ 0 & \textit{otherwise.} \end{cases}$
- $\sigma(e_{x=0})(e_{x:=1}, s) = \delta_1$, *for* $s{\in}\{0, 1\}$.

The corresponding scheduler τ on $1{+}(x{:=}0{\sqcap}1)$, constructed (as per the proof of Thm. 4.13) from σ using the illustrated t-simulation, satisfies:

- $\sigma(\emptyset)(e_\delta, 1) = w(e_{x=1}, 1)\delta_1 = \delta_1$ *and* $\sigma(\emptyset)(e_\delta, 0) = 0$,
- $\sigma(\emptyset)(e_{x:=0\sqcap1}, 1) = 0$ *and* $\sigma(\emptyset)(e_{x:=0\sqcap1}, 0) = w(e_{x=0}, 0)\delta_1 = \delta_1$.

Since $(x{=}1){+}(x{\neq}1){\cdot}(x{:=}1)$ is sequentially equivalent to $x{:=}1$, we can see that the scheduler τ on $1{+}(x{:=}0{\sqcap}1)$ forces the final value of x to be 1 by resolving $(+)$ and $(\sqcap)$ as they were resolved in the program $(x{=}1){+}(x{\neq}1){\cdot}(x{:=}1)$.

We now show that the binary Kleene star is preserved by the semantics map.

Proposition 4.15 *For every non-zero, feasible and terminating event structure $\mathcal{E}$ and $\mathcal{F}$, we have $[\![\mathcal{E}{*}\mathcal{F}]\!] = [\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!]$.*

Proof. For the binary Kleene product, since $[\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!]$ is the least fixed point of $f(X) = [\![\mathcal{F}]\!]{+}[\![\mathcal{E}]\!]{\cdot}X$ in $\mathbb{H}_1\Omega$[8], and $\mathcal{E}{*}\mathcal{F}$ satisfies

$$\mathcal{F}{+}\mathcal{E}{\cdot}(\mathcal{E}{*}\mathcal{F}) \equiv_{\text{sim}} \mathcal{E}{*}\mathcal{F}$$

by construction of the sequences of bundle event structures defining $\mathcal{E}{*}\mathcal{F}$. Therefore, Thm. 4.13 and Prop. 4.8 imply that $[\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!] \sqsubseteq_{\mathbb{H}} [\![\mathcal{E}{*}\mathcal{F}]\!]$.

Conversely, let $\mu{\in}[\![\mathcal{E}{*}\mathcal{F}]\!](s)$ for some initial state $s{\in}\Omega$. As in the case of Prop. 4.8, we assume that μ is computed from a scheduler σ on $\mathcal{E}{*}\mathcal{F}$. We construct a sequence of schedulers σ_n that "converges" to σ as follows. We set σ_0 to be any element of $\mathbf{Sched}_1(\mathcal{F})$, $\sigma_1(\alpha){=}\sigma(\alpha)$ if α is a trace of $\mathcal{F}$ or $\mathcal{E}$, otherwise, we set $\sigma_1(\alpha'\alpha''){=}\sigma_0(\alpha'')$ where $\alpha'{\in}\mathcal{T}_{\max}(\mathcal{E})$ (notice that σ_0 is applied to a different copy of $\mathcal{F}$ but this is not important as event names can be

[8] Notice that the least fixed point is in $\mathbb{H}_1\Omega$ but not $\overline{\mathbb{H}}_1\Omega$. The reason is that $[\![\mathcal{E}]\!]$ and $[\![\mathcal{F}]\!]$ are elements of $\mathbb{H}_1\Omega$ because of feasibility and termination.

abstracted.). Inductively, we define

$$\sigma_n(\alpha) = \begin{cases} \sigma(\alpha) & \text{if } \alpha{\in}\mathcal{T}(\underbrace{\mathcal{F}{+}\mathcal{E}{\cdot}(\ldots\mathcal{E}{\cdot}(\mathcal{F}{+}\mathcal{E}))}_{n \text{ occurrences of } \mathcal{E}}), \\ \sigma_0(\alpha|_F) & \text{otherwise.} \end{cases}$$

Again, σ_0 is applied to the $n{+}1^{\text{th}}$ copy of $\mathcal{F}$. Indeed, we have

$$\sigma_n{\in}\mathbf{Sched}_1(\underbrace{\mathcal{F}{+}\mathcal{E}{\cdot}(\cdots\mathcal{E}{\cdot}(\mathcal{F}{+}\mathcal{E}{\cdot}\mathcal{F}))}_{n \text{ occurrences of } \mathcal{E}})$$

by construction. On the one hand, the sequence of distributions $\sigma_{n,s}(\mathcal{E})$ forms a subset of $[\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!](s)$. On the other hand, let $u{\in}\Omega$ and let us denote

$$\mathcal{T}_{\leq n} = \mathcal{T}(\underbrace{\mathcal{F}{+}\mathcal{E}{\cdot}(\ldots\mathcal{E}{\cdot}(\mathcal{F}{+}\mathcal{E}{\cdot}\mathcal{F}))}_{n \text{ occurrences of } \mathcal{E}}).$$

If we denote by φ_n the complete run of σ_n on $\mathcal{E}{*}\mathcal{F}$, then we have

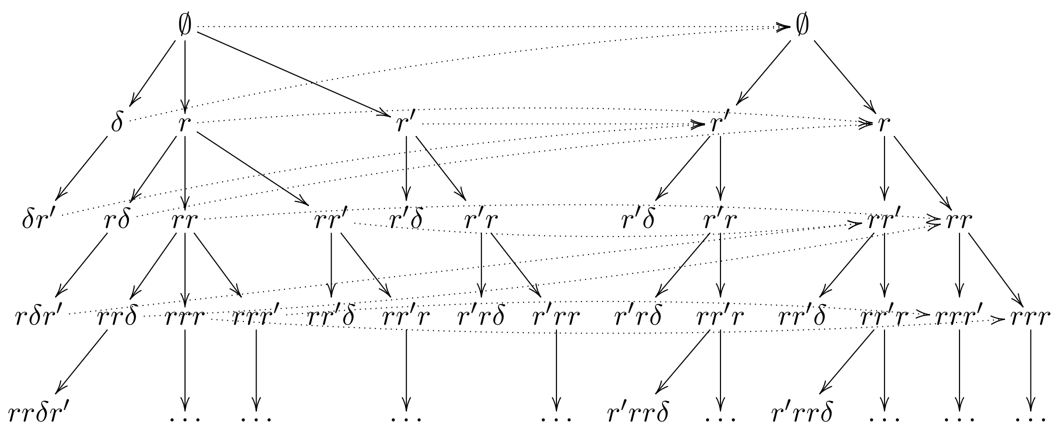
$$\begin{aligned} |\sigma_s(\mathcal{E})(u) - \sigma_{n,s}(\alpha)(u)| &= \left| \sum_{\alpha\in\mathcal{T}_{\max}(\mathcal{E}*\mathcal{F})} \varphi(\alpha)(u) - \sum_{\alpha\in\mathcal{T}_n\cap\mathcal{T}_{\max}(\mathcal{E}*\mathcal{F})} \varphi_n(\alpha)(u) \right| \\ &= \left| \sum_{\alpha\in\mathcal{T}_{\max}(\mathcal{E}*\mathcal{F})\setminus\mathcal{T}_{\leq n-1}} (\varphi(\alpha)(u) - \varphi_n(\alpha)(u)) \right| \\ &\leq \sum_{\alpha\in\mathcal{T}_{\max}(\mathcal{E}*\mathcal{F})\setminus\mathcal{T}_{\leq n-1}} |\varphi(\alpha)(u) - \varphi_n(\alpha)(u)|\,. \end{aligned}$$

The set $\mathcal{T}_{\max}(\mathcal{E}{*}\mathcal{F})\setminus\mathcal{T}_{\leq n-1}$ shrinks, when n increases, because every finite trace of $\mathcal{E}{*}\mathcal{F}$ belongs to some set $\mathcal{T}_{\leq k}$. Therefore, the last sum above is decreasing to 0. Hence, since Ω is a finite set, the sequence $\sigma_{n,s}(\mathcal{E}{*}\mathcal{F})$ converges (pointwise) to $\sigma_s(\mathcal{E}{*}\mathcal{F})$ in $\mathbb{D}\Omega$. Since $[\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!](s)$ is topologically closed, we deduce that $\sigma_s(\mathcal{E}){\in}[\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!](s)$. Therefore, $[\![\mathcal{E}{*}\mathcal{F}]\!] \sqsubseteq_{\mathbb{H}} [\![\mathcal{E}]\!]{*}[\![\mathcal{F}]\!]$. $\square$

Proposition 4.16 *Let $r, r'{\in}\mathbb{H}_1\Omega$ be two atomic programs and let $\mathcal{E}, \mathcal{F}$ be two bundle event structures with internal probability, then*

$$\begin{aligned} r^*\|r^* &\sqsubseteq_{\text{sim}} r^*, && (13)\\ r^*\|r' &\sqsubseteq_{\text{sim}} r{*}(r'{\cdot}r^*), && (14)\\ r^*\|(b{\cdot}\mathcal{E}{+}c{\cdot}\mathcal{F}) &\sqsubseteq_{\text{sim}} r{*}(b{\cdot}(r^*\|\mathcal{E}){+}c{\cdot}(r^*\|\mathcal{E})), && (15)\\ r^*\|(r'{\cdot}\mathcal{E}) &\sqsubseteq_{\text{sim}} r{*}(r'{\cdot}(r^*\|\mathcal{E})), && (16) \end{aligned}$$

where $r^* = r{*}1$.



The "obvious" arrows, such as an arrow from $r'\delta$ to $r'\delta$, have been left out to keep the picture clear.

Fig. 2. The t-simulation from $r^*\|r'$ to $r{*}(r'{\cdot}r^*)$.

PROOF. Let us denote by e_1 and e_2 (resp. e) the events that are labelled by δ in the event structure associated to $r^*\|r^*$ (resp. r^*). Given a trace α of $r^*\|r^*$ that does not contain any of the e_is, we denote by α' unique trace corresponding to α in r^* (i.e. with the same number of events labelled by r).

A t-simulation from $r^*\|r^*$ to r^* is obtained by considering a function f such that

$$f(\alpha) = \begin{cases} (\alpha\backslash\{e_1, e_2\})' \text{ if } e_1{\notin}\alpha \text{ or } e_2{\notin}\alpha, \\ (\alpha\backslash\{e_1, e_2\})'e \text{ if } e_1, e_2{\in}\alpha. \end{cases}$$

The t-simulation (14) is constructed as follows. Let us abstract the event names, i.e. r^k would be a trace where each r is the label of a unique event. Every trace of $r^*\|r'$ is a prefix of $r^m r' r^n \delta$ or $r^m \delta r'$, for some $m, n \geq 0$. Every prefix of either trace corresponds to a unique trace of $r{*}(r'{\cdot}r^*)$. For instance, the maximal trace $r^m\delta r'$ is associated to the weakly maximal trace $r^m r'$ of $r{*}(r'{\cdot}r^*)$. Figure 2 shows an explicit construction of the t-simulation.

The Simulation (15)is similar. Every trace of $r^*\|(b{\cdot}\mathcal{E}{+}c{\cdot}\mathcal{F})$ is a prefix of $r^m b\alpha$ or $r^m c\beta$ or $r^m\delta b\gamma$ or $r^m\delta c\zeta$, where $\alpha{\in}\mathcal{T}(r^*\|\mathcal{E})$, $\beta{\in}\mathcal{T}(r^*\|\mathcal{F})$, $\gamma{\in}\mathcal{T}(\mathcal{E})$, $\zeta{\in}\mathcal{F}$ and $n \geq 0$. Again, prefixes of the first two traces correspond to a unique trace of $r{*}(b{\cdot}(r^*\|\mathcal{E}){+}c{\cdot}(r^*\|\mathcal{F}))$. The maximal trace $r^m\delta b\gamma$ is again mapped to the weakly maximal trace $r^m b\gamma$. Similarly for the fourth case. This indeed results in a t-simulation.

The Simulation (16) is constructed as follows. Every trace of $r^*\|(r'{\cdot}\mathcal{E})$ is a prefix of $r^m r'\alpha$ or $r^m\delta r'\beta$ for some trace $\alpha{\in}\mathcal{T}(r^*\|\mathcal{E})$ and $\beta{\in}\mathcal{T}(\mathcal{E})$. We continue as in the previous case. □

Prop. 4.16 is used mainly to interleave the right operand r^* systematically with the internal structure of $\mathcal{E}$, while preserving the simulation order. More precisely, these equations are applied to generate algebraic proofs for the reduction of one expression into another, where the occurrence of $\|$ is pushed deeper into the sub-expressions (and possibly removed).

5 Probabilistic rely-guarantee conditions

Our first task towards the extension of the rely-guarantee method to probabilistic systems is to provide a suitable definition of a rely condition that contains sufficient quantitative information about the environment and the components of a system.

From a relational point of view, as in Jones' thesis [1], a guarantee condition expresses a constraint between a state and its successor by running the relation as a nondeterministic program. Therefore, it is important to know whether some action is executed atomically or whether it is split into smaller components. For instance, when run in the same environment, a probabilistic choice between $x{:=}x{+}1$ and $x{:=}x{-}1$ produced from an `if...then...else` clause may behave differently from an atomic probabilistic assignment that assigns $x{+}1$ and $x{-}1$ to x with the exact same probability.

Without probability, a common example of a guarantee condition for a given program is the reflexive transitive closure with respect to ($\|$) of the union of all atomic actions in that program [25] which completely captures all possible "effects" of the program. Such a closure property plays a crucial role in the algebraic proof of Rule 2 is achieved through Prop. 4.16. This construction was introduced by Jones [1] and later refined by others [11,20,25].

Non-probabilistic rely-guarantee conditions usually take the form ρ^* for some binary relation ρ, defined on the state space of the studied program. The transitive closure of ρ with respect to the relational composition $(\cdot)$ is usually a desirable property. To obtain a probabilistic guarantee condition from a relation $\rho{\subseteq}\Omega{\times}\Omega$, we construct a probabilistic program $r{\in}\mathbb{H}_1\Omega$ such that

$$r(s) = \{\mu{\in}\mathbb{D}\Omega \mid \mu(\{s' \mid (s,s'){\notin}\rho\}) = 0\}.$$

Equivalently, r is the convex closure of ρ. The following proposition then follows from that construction.

Proposition 5.1 *If a relation $\rho{\subseteq}\Omega{\times}\Omega$ is transitive, then the convex closure r of ρ satisfies $r{\cdot}(r{+}\delta) \sqsubseteq_{\mathbb{H}} r$.*

Proof. Let ρ be a transitive relation, r its associated probabilistic program,

$s{\in}\Omega$ a state and $\mu{\in}[r{\cdot}(r{+}\delta)](s)$. We need to show that $\mu{\in}r(s)$. By definition of the sequential composition $(\cdot)$ (Eqn. (5)), there exists $\nu{\in}r(s)$ and a deterministic program $f \sqsubseteq_{\mathbb{H}} (1{+}r)$ such that $\mu = f{\star}\nu$. Let $u{\in}\Omega$ such that $(s,u){\notin}\rho$, we are going to show that $\mu(u) = 0$. We have:

$$\mu(u) = \sum_{t\in\Omega} f(t)(s)\nu(t) = \sum_{t\in\Omega\wedge(s,t)\in\rho} f(t)(u)\nu(t) = \sum_{t\in\Omega\wedge(s,t)\in\rho\wedge(t,u)\in\rho} f(t)(u)\nu(t).$$

The second equality follows from $\nu(t) = 0$ for every $(s,t){\notin}\rho$. Similarly, the last equality follows from $f(t)(u) = 0$ for $(t,u){\notin}\rho$. The last expression reduces to $\sum_{t\in\Omega\wedge(s,u)\in\rho} f(t)(u)\nu(t)$, by transitivity of ρ, which is an empty sum because $(s,u){\notin}\rho$. Therefore, $\mu(u) = 0$ for every $(u,s){\notin}\rho$, that is $\mu{\in}r(s)$. □

The convex closure of a relation ρ, given in Prop. 5.1, sometimes provides a very general rely condition that is too weak to be useful in the probabilistic case. In practice, a probabilistic assignment is considered atomic and the correctness of many protocols is based on that crucial assumption. Hence the random choice and the writing of the chosen value into a program variable x is assumed to happen instantaneously and no other program can modify x during and in-between these two operations. Thus, probabilistic rely and guarantee conditions need to capture the probabilistic information in such an assignment.

Example 5.2 *Let x be a (integer) program variable with values bounded by* 0 *and n. Let us write $x{:=}\texttt{uniform}(0,n)$ for the program that assigns a random integer between two integers* 0 *and n to the variable x. A probabilistic guarantee condition for that assignment is obtained from the probabilistic program r that satisfies, for every integer $s \in [0,n]$,*

$$r(s) = \left\{\mu \;\middle|\; \mu(\{0,n\}) \geq \frac{1}{n{+}1}\right\}. \tag{17}$$

The condition r specifies the convex set of all probabilistic deterministic programs whose atomic actions establish a state in $\{0,n\}$ with probability at least $\frac{1}{n+1}$. In particular, r is an overspecification of $x{:=}\texttt{uniform}(0,x)$ where the rhs occurrence of x is evaluated to the initial value of x. Since r is transitive, it can prove useful to deduce quantitative properties of $(x{:=}\texttt{uniform}(0,x))^*$.

In practice, constructing a useful transitive probabilistic rely-guarantee condition is difficult, but the standard technique is still valid: the strongest guarantee condition of a given program is the nondeterministic choice of all atomic actions found in that program.

Definition 5.3 *A* probabilistic rely *or* guarantee condition *R is a probabilistic concurrent program such that $R\|R \sqsubseteq_{\mathrm{sim}} R$.*

In particular, the concurrent program $r^* = r{*}1$ is a rely condition because

$$r^*\|r^* \sqsubseteq_{\mathrm{sim}} r^* \tag{18}$$

holds in the event structure model (Prop. 4.16 Eqn. (13)). This illustrates the idea that a rely condition specifies an environment that can stutter or execute a sequence of actions that are bounded by r.

6 Probabilistic rely-guarantee calculus

In this section, we develop the rely-guarantee rules governing programs involving probability and concurrency. An example is given by Rule 2, which allows us check the safety properties of the subsystems and infer the correctness of the whole system in a compositional fashion. We provide a probabilistic version of that rule.

In the previous sections, we have developed the mathematical foundations needed for our interpretation of Hoare triples and *guarantee* relations, namely, the sequential refinement $\sqsubseteq$ and simulation-based order $\sqsubseteq_{\mathrm{sim}}$. Following [25], we only adapt the orders in the algebraic interpretation of rely-guarantee quintuples (Eqn. (1)). That is, validity of probabilistic rely-guarantee quintuples is captured by

$$\{P\ R\}\mathcal{E}\{G\ Q\} \ \Leftrightarrow\ P{\cdot}(R\|\mathcal{E}) \sqsubseteq Q \wedge \mathcal{E} \sqsubseteq_{\mathrm{sim}} G,$$

where $P, \mathcal{E}$ and Q are probabilistic concurrent programs and R and G are rely-guarantee conditions. The first part is seen as a probabilistic instance of the contraction of [7] which specifies the functional behaviour of $R\|\mathcal{E}$ under a precondition P. The second part uses the simulation order which is compositional and very sensitive to the structural properties of the program.

The conditions R and G specify how the component $\mathcal{E}$ interacts with its environment. As we have discussed in the previous section, rely and guarantee conditions are obtained by taking $r^* = r{*}\delta$ for some atomic probabilistic program r. Therefore, $\mathcal{E} \sqsubseteq_{\mathrm{sim}} r^*$ implies that all actions carried by events in $\mathcal{E}$ are either stuttering or satisfying the specification r. This corresponds to the standard approach of Jones [20,1].

The following rules are probabilistic extensions of the related rely-guarantee rules developed in [8,20]. These rules are sound with respect to the event structure semantics of Section 4.

Atomic action: The rely-guarantee rule for an atomic statement r' is provided by the equation

$$r^*\|r' \sqsubseteq_{\mathrm{sim}} r{*}(r'{\cdot}r^*) \tag{19}$$

where r is the rely condition. This equation shows that a (background) program satisfying the rely condition r will not interfere with the low level operations involved in the atomic execution of r'. The programs will be interleaved.

Conditional statement: The rely-guarantee rule for conditional statement is provided by the equation

$$r^*\|(b{\cdot}\mathcal{E}{+}c{\cdot}\mathcal{F}) \sqsubseteq_{\mathrm{sim}} r{*}(b{\cdot}(r^*\|\mathcal{E}){+}c{\cdot}(r^*\|\mathcal{F})). \tag{20}$$

This equation shows how a rely condition r^* distributes through branching structures. The tests b and c are assumed to be atomic and their disjunction is always *true* (this is necessary for feasibility). This assumption may be too strong in general because b may involve the reading of some large data that is too expensive to be performed atomically. However, we may assume that such a reading is done before the guard b is checked and the non-atomic evaluation of the variables involved in b may be assigned to some auxiliary variable that is then checked atomically by b.

Prefixing: the sequential rely-guarantee rule for a probabilistic program expressed using prefixing. We have

$$r^*\|(r'{\cdot}\mathcal{E}) \sqsubseteq_{\mathrm{sim}} r{*}(r'{\cdot}(r^*\|\mathcal{E})). \tag{21}$$

It generalises Rule 19 and tells us that a rely condition r^* distributes through the prefixing operation. In other words, the program r' and $\mathcal{E}$ should tolerate the same rely condition in order to prove any meaningful property of $r{\cdot}\mathcal{E}$. This results from of our interpretation of $\|$ where no synchronisation is assumed.

Concurrent execution: in Rule 2, the concurrent composition $\mathcal{E}\|\mathcal{E}'$ requires an environment that satisfies $R{\cap}R'$ to establish the postcondition $Q{\cap}Q'$. However, such an intersection is not readily accessible at the structural level of event structures. Therefore, the most general probabilistic extension of Rule 2 which applies to our algebraic setting is:

$$\frac{\{P\ R\}\mathcal{E}\{G\ Q\} \qquad \{P\ R'\}\mathcal{E}'\{G'\ Q'\} \qquad G \sqsubseteq_{\mathrm{sim}} R' \qquad G' \sqsubseteq_{\mathrm{sim}} R}{\{P\ R''\}\mathcal{E}\|\mathcal{E}'\{G\|G'\ Q\}}, \tag{22}$$

where R'' is a rely condition such that $R'' \sqsubseteq_{\mathrm{sim}} R$ and $R'' \sqsubseteq_{\mathrm{sim}} R'$. The proof of this rule is exactly the same as in [8,21]. In fact, we have $R'' \sqsubseteq_{\mathrm{sim}} R$, $\mathcal{E}' \sqsubseteq_{\mathrm{sim}} R$, $R\|R \sqsubseteq_{\mathrm{sim}} R$, therefore Eqn. (7) and Equational Implication (11) imply

$$R''\|(\mathcal{E}'\|\mathcal{E}) \sqsubseteq_{\mathrm{sim}} R\|(R\|\mathcal{E}) \sqsubseteq_{\mathrm{sim}} R\|\mathcal{E},$$

and we obtain $P{\cdot}R''\|(\mathcal{E}'\|\mathcal{E}) \sqsubseteq_{\mathrm{sim}} P{\cdot}(R\|\mathcal{E})$ by Eqn. (10). It follows from Thm. 4.13 that $P{\cdot}R''\|(\mathcal{E}'\|\mathcal{E}) \sqsubseteq Q$.

The conclusion does not contain any occurrence of Q', but by symmetry, it is also valid if Q' is substituted for Q. The combined rely condition R'' is constructed such that it is below R and R'. Indeed, if R, R' have a greatest lower bound with respect to $\sqsubseteq_{\mathrm{sim}}$, then R'' can be taken as that bound, so that the strengthening of the rely is as week as possible.

The above rule can be specialised by considering rely-guarantee conditions of the form r^*, where r is an atomic probabilistic program. The following rule is expressed in exactly as in the standard case [8]. This is possible because probabilities are internal.

Proposition 6.1 *The following rule is valid in BES:*

$$\frac{\{P\ r_1^*\}\mathcal{E}_1\{g_1^*\ Q_1\} \qquad \{P\ r_2^*\}\mathcal{E}_2\{g_2^*\ Q_2\} \qquad g_1 \sqsubseteq_{\mathbb{H}} r_2 \qquad g_2 \sqsubseteq_{\mathbb{H}} r_1}{\{P\ (r_1 \cap r_2)^*\}\mathcal{E}_1 \| \mathcal{E}_2\{(g_1 + g_2)^*\ Q_1\}}, \tag{23}$$

where $r, r', g, g' \in \mathbb{H}_1\Omega$ and $g+g'$ is the nondeterministic choice on $\mathbb{H}_1\Omega$.

PROOF. This follows from substituting R and G by respectively r^* and g^* in Rule 22. Moreover $g^* \| g'^* \sqsubseteq_{\mathrm{sim}} (g+g')^*$ holds because $(g+g')^* \| (g+g')^* \sqsubseteq_{\mathrm{sim}} (g+g')^*$ (Eqn. (18)). $\square$

Recall that the nondeterministic choice of $\mathbb{H}_1\Omega$ is obtained by the pointwise union followed by the necessary closure properties for the elements of $\mathbb{H}_1\Omega$. The intersection $r \cap r'$ is obtained by pointwise intersection.

Iteration: a while program is modelled by using the binary Kleene star. The idea is to unfold the loop as far as necessary. The conditional and prefix (sequential) cases can then be applied on the unfolded structure to distribute the rely condition. That is, we write

$$r^* \| ((b \cdot \mathcal{E}) * c) \sqsubseteq_{\mathrm{sim}} r * (c \cdot r^* + b \cdot (r^* \| [\mathcal{E} \cdot (b \cdot \mathcal{E} * c)])).$$

If $\mathcal{E}$ is sequential, then r^* can be "interleaved" within the internal structure of $\mathcal{E} \cdot (b \cdot \mathcal{E} * c)$ by applying the prefixing and conditional statement rules.

The sequential correctness is achieved by the usual generation of probability distributions, obtained from terminating sequential behaviours, on the "totally" unfolded event structure (assuming that $\mathcal{E}$ is sequential). The sequential behaviours are usually obtained by interleaving the rely condition r^* through the internal structure of the unfolded loop. A bounded loop, such as a for loop, should be modelled using a sequence of sequential compositions or prefixing.

7 Application: a faulty Eratosthenes sieve

In this section, we show how to use the previously established rely-guarantee rules to verify a probabilistic property of a faulty Eratosthenes sieve, which is a quantitative variant Jones' example [1].

Let $n \geq 2$ be a natural number and $s_0 = \{2, 3, \ldots, n\}$. For each integer i such that $2 \leq i \leq \sqrt{n}$, we consider a program $\mathtt{thd}_i$ that sequentially removes all (strict) multiples of i from the shared set variable s with a fixed probability p. More precisely, each thread $\mathtt{thd}_i$ is implemented as the following program:

```
for(j = 2 to n/i)
    $u_{i,j}$ : skip $\oplus_p$ remove(i*j from s);
```

where `n/i` is the integer division of `n` by `i`. Each $u_{i,j}$ can be seen as a faulty action that removes the product ij from the current value of s with probability p. The state space of each atomic deterministic program $u_{i,j}$ is $\Omega = \{s \mid s{\subseteq}s_0\}$. In $\mathbb{H}_1\Omega$, $u_{i,j}$ is defined by $u_{i,j}(s) = (1{-}p)\delta_s{+}p\delta_{s\setminus\{ij\}}$. The whole system is specified by the concurrent execution

$$\mathtt{thd}_2\|...\|\mathtt{thd}_{\sqrt{n}} = \|_{i=2}^{\sqrt{n}}(u_{i,2}\cdots u_{i,n/i}).$$

where, in the sequel, $\sqrt{n}$ is computed without decimals.

Let $\pi{=}\{2, 3, \ldots, m\}$ be the set of prime numbers in s_0. Our goal is to compute a "good" lower bound probability that the final state is π, after executing the threads $\mathtt{thd}_i$ concurrently, from the initial state s_0.

We denote by $O_{i,j} = \{s \mid ij{\notin}s\} \subseteq \Omega$ and

$$Q_{i,j}(s) = \{\mu{\in}\mathbb{D}\Omega \mid \mu(O_{i,j}){\geq}p \wedge \mu(\{s' \mid s'{\subseteq}s\}){=}1\}$$

a specification of a probabilistic program that removes ij from the state s with at least probability p and does not add anything to it. We define $O_i = \cap_{j=2}^{n/i}O_{i,j}$, $Q_i = Q_{i,2}{\cdot}Q_{i,3}{\cdot}\ldots{\cdot}Q_{i,n/i}$ and r to be the probabilistic program such that $r(s)$ is the convex closure of $\{\delta_{s'} \mid s'{\subseteq}s\}$.

First, we show that every thread $\mathtt{thd}_i$ guarantees r^*. Second, we show that $\mathtt{thd}_i$ establishes Q_i when run in an environment satisfying r, i.e. $r^*\|\mathtt{thd}_i \sqsubseteq Q_i$, using the atomic and prefix rules 19 and 21. Finally, we apply the concurrency rule 23 to deduce that the system $\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i$ establishes all postconditions $Q_2, Q_3, \ldots Q_{\sqrt{n}}$, when run in an environment satisfying r.

Establising $\mathtt{thd}_i \sqsubseteq_{\text{sim}} r^*$ and $r^*\|\mathtt{thd}_i \sqsubseteq Q_i$

On the one hand, it is clear that $u_{i,j} \sqsubseteq_{\mathbb{H}} r$, for every i, j, and thus $\mathtt{thd}_i \sqsubseteq_{\mathrm{sim}} r^*$ follows from the unfold (8). On the other hand, let us show that $r^* \| \mathtt{thd}_i \sqsubseteq Q_i$. Multiple applications of the prefix-case give

$$r^* \| \mathtt{thd}_i \sqsubseteq_{\mathrm{sim}} r{*}(u_{i,2}{\cdot}(r{*}(u_{i,3}{\cdot}(\dots r{*}(u_{i,n/i}{\cdot}r^*))))).$$

Since the right multiplication $X \mapsto X{\cdot}r$, by any program $r{\in}\mathbb{H}_1\Omega$, is the lower adjoint in a Galois connection [5], the fixed point fusion theorem [26] implies

$$r{*}(u_{i,2}{\cdot}(r{*}(u_{i,3}{\cdot}(\dots r{*}(u_{i,n/i}{\cdot}r^*))))) = r^*{\cdot}u_{i,2}{\cdot}r^*{\cdot}u_{i,3}{\cdot}\dots r^*{\cdot}u_{i,n/i}{\cdot}r^*,$$

where the equality is in $\mathbb{H}_1\Omega$. Thus,

$$r^* \| \mathtt{thd}_i \sqsubseteq r^*{\cdot}u_{i,2}{\cdot}r^*{\cdot}u_{i,3}{\cdot}\dots{\cdot}r^*{\cdot}u_{i,n/i}{\cdot}r^*$$

follows from the fact that $\sqsubseteq$ is weaker than $\sqsubseteq_{\mathrm{sim}}$ (Thm. 4.13). The right hand side explicitly states the interleaving of the rely condition r^* in-between the atomic executions in $\mathtt{thd}_i$ as in [20].

Moreover, since r is the probabilistic version of a transitive binary relation, Prop. 5.1 implies that $r{\cdot}(r{+}\delta) \sqsubseteq_{\mathbb{H}} r$. Since $\mathbb{H}_1\Omega$ is a probabilistic Kleene algebra [27], the right induction law of pKA implies $r^* = \delta{+}r$. This reduction of r^* to $\delta{+}r$ illustrates the practical importance of transitive rely conditions. Therefore,

$$r^* \| \mathtt{thd}_i \sqsubseteq (\delta{+}r){\cdot}u_{i,2}{\cdot}(\delta{+}r){\cdot}u_{i,3}{\cdot}\dots(\delta{+}r){\cdot}u_{i,n/i}{\cdot}(\delta{+}r),$$

where the left hand side is a sequential program (thus Prop. 4.8 enables us to use the definition of sequential composition of $\mathbb{H}_1\Omega$) directly . Since $u_{i,j} \sqsubseteq Q_{i,j}$, it remains to show that $(\delta{+}r){\cdot}Q_{i,2}{\cdot}(\delta{+}r){\cdot}Q_{i,3}{\cdot}\dots(\delta{+}r){\cdot}Q_{i,n/i}{\cdot}(\delta{+}r) \sqsubseteq Q_i$.

First we show that $Q_{i,j}{\cdot}(\delta{+}r) \sqsubseteq Q_{i,j}$ and $(\delta{+}r){\cdot}Q_{i,j} \sqsubseteq Q_{i,j}$. Let $s{\in}\Omega$ and $\nu{\in}(Q_{i,j}{\cdot}(\delta{+}r))(s)$. By definition of the sequential composition in $\mathbb{H}_1\Omega$, there exists a probabilistic deterministic program $f \sqsubseteq_{\mathbb{H}} \delta{+}r$ and a distribution $\mu{\in}Q_{i,j}(s)$ such that $\nu(s') = \sum_{t\in\Omega} f(t)(s')\mu(t)$, for every $s'{\in}\Omega$. Therefore,

$$\nu(O_{i,j}{\cap}\{s' \mid s'{\subseteq}s\}) = \sum_{t\in\Omega} f(t)(O_{i,j})\mu(t) = \sum_{t\subseteq s} f(t)(O_{i,j})\mu(t),$$

where the second equality follows from $\mu(\{t \mid t{\not\subseteq}s\}){=}0$, for every $\mu{\in}Q_{i,j}$. We deduce $\sum_{t\subseteq s} f(t)(O_{i,j})\mu(t) \geq p$, i.e. $\nu{\in}Q_{i,j}(s)$, by observing

$$\sum_{t\subseteq s} f(t)(O_{i,j})\mu(t) \geq \sum_{ij\notin t\wedge t\subseteq s} f(t)(O_{i,j})\mu(t) = \mu(O_{i,j}{\cap}\{t \mid t{\subseteq}s\}) \geq p,$$

because $f(t)(O_{i,j}){=}1$ for every t such that $ij{\notin}t$ and $\mu(O_{i,j}{\cap}\{t \mid t{\subseteq}s\}) = \mu(O_{ij})$ for every $\mu{\in}Q_{i,j}(s)$. Consequently, $Q_{i,j}{\cdot}(\delta{+}r) \sqsubseteq Q_{i,j}$. Similarly, we can show that $(\delta{+}r){\cdot}Q_{i,j} \sqsubseteq Q_{i,j}$ and thus $r^* \| \mathtt{thd}_i \sqsubseteq Q_i$.

Establising the property of $r^*\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i$

Applying the rule 23 $\sqrt{n}-1$ times, we obtain, for every Q_j such that $2\leq j\leq\sqrt{n}$,

$$r^*\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i \sqsubseteq Q_j.$$

Inferring a lower bound for the probability of correctness

Unfortunately, Rule 23 does not give any explicit quantitative bound in term of probability for correctness. It does provide quantitative correctness, but all the probabilities are buried in the Q_i.

To obtain an explicit lower bound for the probability of removing all composite numbers, we first study the case of two threads that run concurrently. We know from Rule 23 that $r^*\|\mathtt{thd}_2\|\mathtt{thd}_3 \sqsubseteq Q_2$ and $r^*\|\mathtt{thd}_2\|\mathtt{thd}_3 \sqsubseteq Q_3$. Therefore, for every $\mu\in[\![r^*\|\mathtt{thd}_2\|\mathtt{thd}_3]\!](s_0)$, we have $\mu(O_2) \geq p^{n/2-1}$ and $\mu(O_3) \geq p^{n/3-1}$ because there are $n/2-1$ (resp. $n/3-1$) multiples of 2 (resp. 3) in $[3,n]$ (resp. $[4,n]$). Therefore, $\mu(O_1\cup O_2)+\mu(O_2\cap O_3) = \mu(O_1)+\mu(O_2) \geq p^{n/2-1}+p^{n/3-1}$ and

$$\mu(O_2\cap O_3) \geq p^{n/2-1}+p^{n/3-1}-1. \tag{24}$$

In the construction of the lower bound in Eqn. (24), we have only used the modularity of measures and, therefore, it can be transformed into a more general rely-guarantee rule with explicit probabilities (Prop. 7.1).

Given a subset $O\subseteq\Omega$ and $p\in[0,1]$, we write $[\![\mathcal{E}]\!](s_0)(O) \geq p$ if for every $\mu\in[\![\mathcal{E}]\!](s_0)$ we have $\mu(O) \geq p$.

Proposition 7.1 *For every initial state s_0 and for all subsets $O_1, O_2\subseteq\Omega$,*

$$\frac{[\![r_1^*\|\mathcal{E}_1]\!](s_0)(O_1) \geq p_1 \quad [\![r_2^*\|\mathcal{E}_2]\!](s_0)(O_2) \geq p_2 \quad \mathcal{E}_1 \sqsubseteq_{\text{sim}} g^* \sqsubseteq_{\text{sim}} r_2^* \quad \mathcal{E}_2 \sqsubseteq_{\text{sim}} g'^* \sqsubseteq_{\text{sim}} r_1^*}{[\![(r_1\cap r_2)^*\|\mathcal{E}_1\|\mathcal{E}_2]\!](s_0)(O_1\cap O_2) \geq p_1+p_2-1 \qquad \mathcal{E}_1\|\mathcal{E}_2 \sqsubseteq_{\text{sim}} (g+g')^*}.$$

PROOF. Let $\mu \in [\![(r_1\cap r_2)^*\|\mathcal{E}\|\mathcal{E}_2]\!](s_0)$, we need to show that $\mu(O_1\cap O_2) \geq p_1+p_2-1$ with the above definition of p_1 and p_2.

Let us define Q_1 to be the (single event) ipBES whose event is labelled by the probabilistic program u_1 such that $u_1(s_0) = \{\mu \mid \mu(O_1)\geq p_1\}$ else $u_1(s) = \mathbb{D}\Omega$ for $s \neq s_0$. Similarly, we define Q_2. Then the premises imply $r_1^*\|\mathcal{E}_1 \sqsubseteq Q_1$ and $r_2^*\|\mathcal{E}_2 \sqsubseteq Q_2$. By Prop. 6.1, we have

$$[\![(r_1\cap r_2)^*\|\mathcal{E}_1\|\mathcal{E}_2]\!] \sqsubseteq_{\mathbb{H}} [\![Q_1]\!] \qquad \text{and} \qquad [\![(r_1\cap r_2)^*\|\mathcal{E}_1\|\mathcal{E}_2]\!] \sqsubseteq_{\mathbb{H}} [\![Q_2]\!].$$

Therefore $\mu(O_1) \geq p_1$ and $\mu(O_2) \geq p_2$. Modularity of finite measures implies that $\mu(O_1\cap O_2)+\mu(O_1\cup O_2) = \mu(O_1)+\mu(O_2) \geq p_1+p_2$. Hence, $\mu(O_1\cap O_2) \geq p_1+p_2-\mu(O_1\cup O_2) \geq p_1+p_2-1$ since $\mu(O_1\cup O_2) \leq 1$.

The simulation $\mathcal{E}_1\|\mathcal{E}_2 \sqsubseteq_{\text{sim}} (g+g')^*$ is also clear from Prop. 6.1. $\square$

We know from the above discussion that

$$[\![r^*\|\mathtt{thd}_2\|\mathtt{thd}_3]\!](s_0)(O_2{\cap}O_3) \geq p^{n/2-1}{+}p^{n/3-1}{-}1.$$

Applying Prop. 7.1 on $\mathtt{thd}_2\|\mathtt{thd}_3$ and $\mathtt{thd}_4$ yields

$$[\![r^*\|\mathtt{thd}_2\|\mathtt{thd}_3\|\mathtt{thd}_4]\!](s_0)(O_2{\cap}O_3{\cap}O_4) \geq p^{n/2-1}{+}p^{n/3-1}{+}p^{n/4-1}{-}2.$$

Thus $\sqrt{n}{-}1$ applications of Prop. 7.1 give

$$[\![r^*\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i]\!](s_0)(\cap_{i=2}^{\sqrt{n}}O_i) \geq \sum_{i=2}^{\sqrt{n}} p^{n/i-1}{-}(\sqrt{n}{-}2) = f(p,n).$$

The lower bound $f(p,n)$ sometimes provides a bad lower-approximation for the probability that the system establishes $\cap_{i=2}^{\sqrt{n}}O_i$. However, it is clear that $\lim_{p\to 1} f(p,n) = f(1,n) = 1$.

In the particular case of $n = 15$, we have $\sqrt{15} = 3$ and we only need to consider $\mathtt{thd}_2$ and $\mathtt{thd}_3$ so that $f(p,15) = p^6{+}p^4{-}1$. The plot of $f(p,15)$ in Fig. 3 shows that $f(p,15)$ gives a positive lower bound when $p \geq 0.868$, the exact probability being $p^{10}{+}4p^9(1{-}p){+}4p^8(1{-}p)^2$.

Refining the lower bound

We can use other internal properties of the system to obtain a better lower bound. It is clear that O_i is an invariant for every $\mathtt{thd}_j$ (for $j{\neq}i$) and that all actions $u_{i,j}$ (sequentially) commute with each other. Thus, we should obtain a better lower bound by noticing that the system is "sequentially better" than the following interleaving: $\mathtt{thd}_2$ removes all (strict) multiples of 2, $\mathtt{thd}_3$ removes all multiples of 3 assuming that all multiples of $\mathrm{lcm}'(2,3)$ (the lowest common multiple of 2 and 3 that is strictly greater than both) have been removed by $\mathtt{thd}_2$, and so on [9]. Thus

$$\begin{aligned}[\![r^*\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i]\!](s_0)(\cap_{i=2}^{\sqrt{n}}O_i) &\geq p^{n/2-1}p^{n/3-1-[n/6]}p^{n/4-1-[n/4-1]}p^{n/5-1-[n/10+n/15-n/30]}\dots\\ &= g(p,n),\end{aligned}$$

where the square-bracketed terms are the numbers of multiples remove by threads with smaller indices. For example, before $\mathtt{thd}_5$ runs, $\mathtt{thd}_2$ removes $n/10$ multiples of $\mathrm{lcm}'(2,5)$, $\mathtt{thd}_3$ removes $n/15{-}n/30$ multiples of $\mathrm{lcm}'(3,5)$ (not multiples of $\mathrm{lcm}'(2,5)$), thus $\mathtt{thd}_5$ removes the remaining $n/5{-}1{-}[n/10{+}n/15{-}n/30]$ multiples of 5. In the particular case of $n = 15$, this yields

$$g(p,15) = p^{15/2-1}p^{15/3-1-15/6} = p^{7-1+5-1-2} = p^8.$$

[9] The probability of removing all composite numbers is usually above that bound because 6 can be removed by either $\mathtt{thd}_2$ or $\mathtt{thd}_3$.

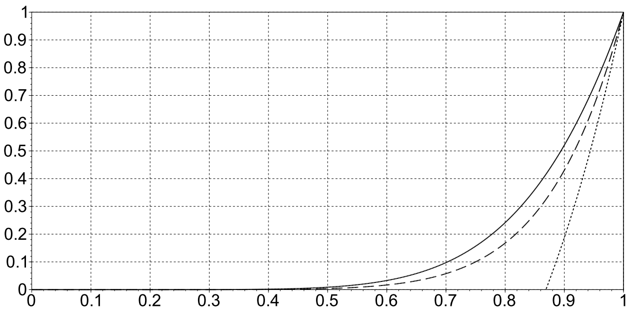


Fig. 3. Comparison of the quantities $f(p, 15)$ (dotted), $g(p, 15)$ (dashed) and the actual probability $p^{10}+4p^9(1-p)+4p^8(1-p)^2$ (solid).

A graphical comparison of f, g and the actual probability is displayed in Figure 3 for $n = 15$.

Establising the property of $\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i$

Finally, notice that $\emptyset \in \cap_{i=2}^{\sqrt{n}} O_i$ which means that $r^* \|_{i=2}^{\sqrt{n}}\mathtt{thd}_i$ can establish $s = \emptyset$ with a positive probability. This issue is resolved by using a stronger guarantee property such as "$u_{i,j}$ never removes i". Therefore, $\|_{i=2}^{\sqrt{n}}\mathtt{thd}_i$ never removes any prime numbers i.e. any element of $\cap_{i=2}^{\sqrt{n}} O_i$, that does not contain all the positive prime numbers below n, occurs with probability 0.

8 Conclusion

We have presented an extension of the rely-guarantee calculus that accounts for probabilistic programs running in a shared variable environment. The rely-guarantee rules are expressed and derived by and large by using the algebraic properties of a bundle event structure semantics for concurrent programs.

In our approach, the specification of a probabilistic concurrent program is expressed with a rely-guarantee quintuple. Each quintuple is defined algebraically through the use of a sequential order $\sqsubseteq$, which captures all possible sequential behaviours when a suitable definition of the concurrency operation $\|$ is given, and a simulation order $\sqsubseteq_{\mathrm{sim}}$, which specifies the level of interference between the specified component and the environment. Various probabilistic rely-guarantee rules have been established and applied on a simple example of a faulty concurrent system. We have also shown some rules that provide explicit quantitative properties, including a lower bound for the probability of correctness. In particular, a better lower-approximation can be derived if further internal properties of the systems are known.

The framework developed in this paper has its current limitations. Firstly, neither the algebra nor the event structure model support non-terminating probabilistic concurrent programs at the moment. That is, the rely-guarantee rules of this paper can only be applied in a partial correctness setting. Secondly, the concrete model is restricted to programs with finite state spaces. We will focus particularly on the first limitation in our future work.

References

[1] C. B. Jones, Development methods for computer programs including a notion of interference, Ph.D. thesis, Oxford University (June 1981).

[2] D. Kozen, Semantics of probabilistic programs, J. Comput. Syst. Sci. 22 (3) (1981) 328–350.

[3] C. Jones, Probabilistic non-determinism, Ph.D. thesis, University of Edinburgh, Scotland, UK (1989).

[4] J. He, K. Seidel, A. McIver, Probabilistic models for the guarded command language, Sci. Comput. Program. 28 (2-3) (1997) 171–192.

[5] A. K. McIver, C. C. Morgan, Abstraction, Refinement And Proof For Probabilistic Systems, SpringerVerlag, 2004.

[6] R. Segala, N. A. Lynch, Probabilistic simulations for probabilistic processes., Nord. J. Comput. 2 (2) (1995) 250–273.

[7] A. Armstrong, V. B. F. Gomes, G. Struth, Algebraic principles for rely-guarantee style concurrency verification tools, in: C. B. Jones, P. Pihlajasaari, J. Sun (Eds.), FM 2014, 2014, pp. 78–93.

[8] C. A. R. Hoare, B. Möller, G. Struth, I. Wehrman, Concurrent Kleene algebra and its foundations, J. Log. Algebr. Program. 80 (6) (2011) 266–296.

[9] S. L. Bloom, Z. Ésik, Free shuffle algebras in language varieties, Theoretical Computer Science 163 (1&2) (1996) 55–98.

[10] A. Tarlecki, A language of specified programs., Sci. Comput. Program. 5 (1) (1985) 59–81.

[11] J. Dingel, A refinement calculus for shared-variable parallel and distributed programming., Formal Asp. Comput. 14 (2) (2002) 123–197.

[12] J. W. Coleman, C. B. Jones, A structural proof of the soundness of Rely-guarantee rules., J. Log. Comput. 17 (4) (2007) 807–841.

[13] F. S. de Boer, U. Hannemann, W. P. de Roever, A compositional proof system for shared variable concurrency., in: J. S. Fitzgerald, C. B. Jones, P. Lucas (Eds.), FME, Vol. 1313 of LNCS, Springer, 1997, pp. 515–532.

[14] G. Winskel, Events in computation, Ph.D. thesis, University of Edinburgh, Scotland, UK (1980).

[15] G. Winskel, Event structures, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.), Advances in Petri Nets, 1986, pp. 325–392.

[16] R. Langerak, Bundle event structures: a non-interleaving semantics for LOTOS, Memoranda informatica, University of Twente, 1992.

[17] J.-P. Katoen, R. Langerak, D. Latella, Modeling systems by probabilistic process algebra: An event structures approach, in: TC6/WG6.1, FORTE '93, North-Holland Publishing Co., The Netherlands, 1994, pp. 253–268.

[18] J. P. Katoen, Quantitative and qualitative extensions of event structures, Ph.D. thesis, University of Twente (1996).

[19] D. Varacca, Probability, nondeterminism and concurrency: two denotational models for probabilistic computation, Ph.D. thesis, University of Aarhus (2003).

[20] I. J. Hayes, C. B. Jones, C. R. J., Refining rely-guarantee thinking, Tech. rep., Newcastle University, United Kingdom (2012).

[21] A. K. McIver, T. M. Rabehaja, G. Struth, Probabilistic concurrent Kleene algebra, in: L. Bortolussi, H. Wiklicky (Eds.), QAPL, Vol. 117 of EPTCS, 2013, pp. 97–115.

[22] S. Georgievska, S. Andova, Retaining the probabilities in probabilistic testing theory, in: L. Ong (Ed.), FOSSACS, Vol. 6014 of LNCS, Springer, 2010, pp. 79–93.

[23] S. Georgievska, S. Andova, Probabilistic may/must testing: retaining probabilities by restricted schedulers, FAC 24 (4-6) (2012) 727–748.

[24] A. K. McIver, T. M. Rabehaja, G. Struth, An event structure model for probabilistic concurrent Kleene algebra., in: K. L. McMillan, A. Middeldorp, A. Voronkov (Eds.), LPAR, Vol. 8312 of LNCS, Springer, 2013, pp. 653–667.

[25] C. A. R. Hoare, B. Möller, G. Struth, I. Wehrman, Concurrent Kleene algebra., in: M. Bravetti, G. Zavattaro (Eds.), CONCUR, Vol. 5710 of LNCS, Springer, 2009, pp. 399–414.

[26] R. Backhouse, Galois connections and fixed point calculus, in: R. Backhouse, R. Crole, J. Gibbons (Eds.), ACM/MPC, Vol. 2297 of LNCS, Springer Berlin Heidelberg, 2002, pp. 89–150.

[27] A. K. McIver, T. Weber, Towards automated proof support for probabilistic distributed systems, in: G. Sutcliffe, A. Voronkov (Eds.), LPAR, Vol. 3835 of LNAI, Springer, 2005, pp. 534–548.

A Axioms of Kleene algebra and related structures

A.1 Idempotent semiring

An *idempotent semiring* is an algebraic structure $(K, +, \cdot, 0, 1)$ such that, for every $x, y, z \in K$, the following axioms hold

$$\begin{aligned}
x + x &= x, && \text{(A.1)}\\
x + y &= y + x, && \text{(A.2)}\\
x + (y + z) &= (x + y) + z, && \text{(A.3)}\\
x + 0 &= x, && \text{(A.4)}\\
x \cdot 1 &= x, && \text{(A.5)}\\
1 \cdot x &= x, && \text{(A.6)}\\
x \cdot (y \cdot z) &= (x \cdot y) \cdot z, && \text{(A.7)}\\
0 \cdot x &= 0, && \text{(A.8)}\\
x \cdot 0 &= 0, && \text{(A.9)}\\
(x + y) \cdot z &= x \cdot z + y \cdot z, && \text{(A.10)}\\
x \cdot y + x \cdot z &= x \cdot (y + z). && \text{(A.11)}
\end{aligned}$$

A.2 Kleene algebra

A *Kleene algebra* is an algebraic structure $(K, +, \cdot, ^*, 0, 1)$ where $(K, +, \cdot, 0, 1)$ is an idempotent semiring and the Kleene star $(^*)$ satisfies Kozen's axioms:

$$\begin{aligned}
x^* &= 1 + x \cdot x^*, && \text{(A.12)}\\
z + x \cdot y \leq y &\Rightarrow x^* \cdot z \leq y, && \text{(A.13)}\\
z + y \cdot x \leq y &\Rightarrow z \cdot x^* \leq y. && \text{(A.14)}
\end{aligned}$$

The induction laws A.13 (resp. A.14) implies that x^* is the least fixed point of $\lambda y.1 + x \cdot y$ (resp. $\lambda y.1 + y \cdot x$).

A.3 Probabilistic Kleene algebra

A *probabilistic Kleene algebra* has the same signature as Kleene algebra but weakens the distributivity law A.11 and the induction rule A.14 to:

$$x{\cdot}y + x{\cdot}z \leq x{\cdot}(y+z), \tag{A.15}$$
$$z + y{\cdot}(x+1) \leq y \Rightarrow z{\cdot}x^* \leq y. \tag{A.16}$$

A.4 Concurrent Kleene algebra

A *concurrent Kleene algebra* is composed of a Kleene algebra $(K, +, \cdot, ^*, 0, 1)$ and a commutative Kleene algebra $(K, +, \|, ^{(*)}, 0, 1)$ (i.e. $\|$ is commutative) linked by the interchange law:

$$(x\|y) \cdot (x'\|y') \leq (x \cdot x')\|(y\|y'). \tag{A.17}$$