Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# 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

ARTICLE INFO

*Article history:*
Received 10 July 2015
Accepted 12 January 2016
Available online xxxx

*Keywords:*
Probabilistic programs
Concurrency
Rely-guarantee
Program verification
Program semantics
Kleene algebra
Event structures

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.



## 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 concurrency in compositional fashion [1]. This paper extends Jones' calculus to the quantitative correctness of probabilistic concurrent programs.

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.

☆ This research was supported by an iMQRES from Macquarie University, the ARC Discovery Grant DP1092464 and the EPSRC Grant EP/J003727/1.
* Corresponding author.
*E-mail addresses:* annabelle.mciver@mq.edu.au (A. McIver), tahiry.rabehaja@mq.edu.au (T. Rabehaja), g.struth@sheffield.ac.uk (G. Struth).

http://dx.doi.org/10.1016/j.tcs.2016.01.016
0304-3975/

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 work-around 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 Sections 2 and 5–6. Sections 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.

Most of the results in this paper are supported by proof sketches. Complete and detailed proofs can be found in the full arXiv version [9].

## 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 the appendix of the full version [9].

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. [10]).

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 $(s_1, s_1'), (s_2, s_2') \ldots (s_k, s_k')$, 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 [11] 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