Performance Evaluation 73 (2014) 110–132

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Operational versus weakest pre-expectation semantics for the probabilistic guarded command language

Friedrich Gretz [a,b,*], Joost-Pieter Katoen [a], Annabelle McIver [b]

[a] RWTH Aachen University, Aachen, Germany

[b] Macquarie University, Sydney, Australia

ARTICLE INFO

Article history:
Available online 17 December 2013

Keywords:
Expectation transformer semantics
Operational semantics
Markov decision process
Expected rewards

ABSTRACT

This paper proposes a simple operational semantics of pGCL, Dijkstra's guarded command language extended with probabilistic choice, and relates this to pGCL's *wp*-semantics by McIver and Morgan. Parametric Markov decision processes whose state rewards depend on the post-expectation at hand are used as the operational model. We show that the *weakest pre-expectation* of a pGCL-program w.r.t. a post-expectation corresponds to the *expected cumulative reward* to reach a terminal state in the parametric MDP associated to the program. In a similar way, we show a correspondence between weakest *liberal* pre-expectations and *liberal* expected cumulative rewards. The verification of probabilistic programs using *wp*-semantics and operational semantics is illustrated using a simple running example.

1. Introduction

Formal semantics of programming languages has been the subject of intense research in computer science for several decades. Various approaches have been developed for the description of program semantics. Structured operational semantics defines the meaning of a program by means of an abstract machine where states correspond to program configurations (typically consisting of a program counter and a variable valuation) and transitions model the evolution of a program by executing statements. Program executions are then the possible runs of the abstract machine. Denotational semantics maps a program onto a mathematical object that describes for instance its input–output behaviour. Finally, axiomatic semantics provides the program semantics in an indirect manner by describing its properties. A prominent example of the latter are Hoare triples in which annotations, written in predicate logic, are associated to control points of the program.

The semantics of Dijkstra's seminal guarded command language [1] from the seventies is given in terms of weakest preconditions. It is in fact a predicate transformer semantics, i.e. a total function between two predicates on the state of a program. The predicate transformer $E = wp(P, F)$ for program P and postcondition F yields the weakest precondition E on the initial state of P ensuring that the execution of P terminates in a final state satisfying F. There is a direct relation with axiomatic semantics: the Hoare triple $\langle E\rangle\ P\ \langle F\rangle$ holds for total correctness if and only if $E \Rightarrow wp(P, F)$. The weakest *liberal* precondition $wlp(P, F)$ yields the weakest precondition for which P either does not terminate or establishes F. It does not ensure termination and corresponds to Hoare logic in partial correctness. Although providing an operational semantics for the guarded command language is rather straightforward, it was not until the early nineties that Lukkien [2,3] provided a formal connection between the predicate transformer semantics and the notion of a computation.

* Corresponding author at: Macquarie University, Sydney, Australia.
E-mail addresses: friedrich.gretz@students.mq.edu.au, fgretz@cs.rwth-aachen.de (F. Gretz), katoen@cs.rwth-aachen.de (J.-P. Katoen), annabelle.mciver@mq.edu.au (A. McIver).

0166-5316/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.peva.2013.11.004

Qualitative annotations in predicate calculus are often insufficient for probabilistic programs as they cannot express quantities such as expectations over program variables. To that end, McIver and Morgan [4] generalised the methods of Dijkstra and Hoare to probabilistic programs by making the annotations real-valued expressions – referred to as expectations – in the program variables. Expectations are the quantitative analogue of predicates. This yields an expectation transformer semantics of the probabilistic guarded command language (pGCL, for short), an extension of Dijkstra's language with a probabilistic choice operator. An expectation transformer is a total function between two expectations on the state of a program. The expectation transformer $e = wp(P, f)$ for pGCL-program P and post-expectation f over final states yields the least expected value e on P's initial state ensuring that P's execution terminates with a value f. The annotation $\langle e \rangle\ P\ \langle f \rangle$ holds for total correctness if and only if $e \leq wp(P, f)$, where $\leq$ is to be interpreted in a point-wise manner. The weakest *liberal* pre-expectation $wlp(P, f)$ yields the least expectation for which P either does not terminate or establishes f. It does not ensure termination and corresponds to partial correctness.

This paper provides a simple operational semantics of pGCL using parametric Markov decision processes (pMDPs), a slight variant of MDPs in which probabilities may be parameterised [5,6]. Our main contribution in this paper is a formal connection between the *wp*- and *wlp*-semantics of pGCL by McIver and Morgan and the operational semantics of pGCL. This provides a clean and insightful relationship between the abstract expectation transformer semantics that has been proven useful for formal reasoning about probabilistic programs, and the notion of a computation in terms of the operational model, a pMDP. In order to establish this connection we equip pMDPs with state rewards that depend on the post-expectation at hand. Intuitively speaking, we decorate a terminal state in the operational model of a program with a reward that corresponds to the value of the post-expectation. All other states are assigned reward zero. We then show that the *weakest pre-expectation* of a pGCL-program P w.r.t. a post-expectation corresponds to the *expected cumulative reward* to reach a terminal state in the pMDP associated to P. In a similar way, we show that weakest *liberal* pre-expectations correspond to *liberal* expected cumulative rewards. The proofs are by induction on the structure of our probabilistic programs using standard results from fixed point theory. This paper thus yields a correspondence theorem that enables us to understand the mathematically involved expectation transformers intuitively using only first principles of Markov decision processes with rewards. In addition, for finite-state programs (or program fragments), our result implies that algorithms for computing expected accumulated rewards in MDPs – for which efficient algorithms and tools based on linear programming exist – can be employed for computing weakest pre-expectations. Finally we recall the notion of probabilistic invariants [4] and apply our correspondence theorem to find an operational characterisation of invariants (which originally are defined in terms of expectation transformers).

1.1. Related work

The MDP semantics of pGCL in this paper bears strong resemblance to the operational semantics of similar languages. To mention a few, Baier et al. [7] provide an MDP semantics of a probabilistic version of Promela, the modelling language of the SPIN model checker. Di Pierro et al. [8] give a semantics to a very similar programming language without non-determinism. The seminal work by Kozen [9] provides two semantics of a deterministic variant of pGCL and shows their correspondence. Kozen interprets probabilistic programs as partial measurable functions on a measurable space, and as continuous linear operators on a Banach space of measures. He et al. [10] provide a mapping from a semantics based on a probabilistic complete partial order which contains non-determinism à la Jones [11] to a semantics which is a mapping from initial states to sets of probability distributions over final states. To our knowledge, our results on relating weakest pre-expectations of pGCL and an operational semantics are novel. Our set-up and results can be considered as a probabilistic analogue of the work by Lukkien [2,3] who provided a formal connection between the predicate transformer semantics of Dijkstra's guarded command language and the operational notion of a computation.

More examples of how to discover and apply invariants when reasoning about probabilistic programs can be found at [12]. There we also describe Prinsys, a tool for semi-automatic invariant generation.

1.2. Structure of this paper

The rest of the paper is divided as follows. In Section 2 we introduce the probabilistic programming language pGCL. Parametric Markov decision processes with rewards are introduced in Section 3. Section 4 recaps the denotational semantics of pGCL [4] and introduces operational semantics for this language. Then the main result is established, namely that the two semantics are equivalent. Section 5 provides an example of reasoning over pGCL programs. Finally, Section 6 introduces invariants and uses our main result to give an operational characterisation for them.

This paper is an extended version of the conference paper [13]. This version contains a generalised version of the proofs of Theorems 23 and 24, a new section on invariants and an Appendix with a new proof for continuity of $wp(P, \cdot)$ and $wlp(P, \cdot)$.

2. Probabilistic programs

Our programming language pGCL [4] is an extension of Dijkstra's guarded command language [1]. Besides a non-deterministic choice operator, denoted [], and a conditional choice, it incorporates a probabilistic choice operator, denoted