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

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Local abstraction refinement for probabilistic timed programs

Klaus Dräger [a,*], Marta Kwiatkowska [a], David Parker [b], Hongyang Qu [a]

[a] Department of Computer Science, University of Oxford, Oxford, UK
[b] School of Computer Science, University of Birmingham, Birmingham, UK

ARTICLE INFO

Keywords:
Probabilistic verification
Abstraction refinement

ABSTRACT

We consider models of programs that incorporate probability, dense real-time and data. We present a new abstraction refinement method for computing minimum and maximum reachability probabilities for such models. Our approach uses strictly local refinement steps to reduce both the size of abstractions generated and the complexity of operations needed, in comparison to previous approaches of this kind. We implement the techniques and evaluate them on a selection of large case studies, including some infinite-state probabilistic real-time models, demonstrating improvements over existing tools in several cases.

1. Introduction

Abstraction refinement is a highly successful approach to the verification of complex infinite-state systems. The basic idea is to construct a sequence of increasingly precise abstractions of the system to be verified, with each abstraction typically over-approximating its behaviour. Successive abstractions are constructed through a process of refinement which terminates once the abstraction is precise enough to verify the desired property of the system under analysis. Abstraction refinement techniques have also been used to verify probabilistic systems [6,11,13,7], including those with real-time characteristics [16,17,8] and continuous variables [25]. Frequently, though, practical implementations of these techniques are hindered by the high complexity of both the abstractions involved and the operations needed to construct and refine them.

In this paper, we target the verification of programs whose behaviour incorporates both probabilistic and real-time aspects, and which include the manipulation of (potentially infinite) data variables. We analyse systems modelled as *probabilistic timed programs* (PTPs) [17], whose semantics are defined as infinite-state Markov decision processes (MDPs). We introduce an abstraction refinement procedure for computing minimum and maximum reachability probabilities in PTPs. As in [6,11], we use an MDP-based abstraction. This provides *outer* bounds on reachability probabilities (i.e., a lower bound on the minimum probability or an upper bound on the maximum). In addition, we compute dual, *inner* bounds, based on a stepwise concretisation of adversaries of this abstract MDP, yielding upper and lower bounds on minimum and maximum probabilities, respectively. Concretisation is also used, for example, in [11], for untimed models. The key difference in our work is that we aim to keep the abstraction small by using *local* refinement and simplification operations, so as to reduce the need for expensive operations such as Craig interpolation.

At the core of our approach is a refinement loop that repeatedly attempts to construct a concrete adversary of the PTP. This is based on the exploration of the part of the state space on which the current abstract adversary can be concretised. In each exploration step, we may encounter an inconsistency, in which case we derive a refinement operation and restart. Otherwise, we numerically solve the constructed adversary, giving inner bounds on the desired probability values. The refinement loop terminates once the difference between upper and lower bounds is smaller than a specified threshold ε.

* Corresponding author.
E-mail address: klaus.draeger@cs.ox.ac.uk (K. Dräger).

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.07.013

Please cite this article in press as: K. Dräger et al., Local abstraction refinement for probabilistic timed programs, Theoretical Computer Science (2013), http://dx.doi.org/10.1016/j.tcs.2013.07.013

We implement our abstraction refinement approach, deploy it on various large case studies, and compare to the probabilistic verification tools PRISM [18], PASS [9] and FORTUNA [3], illustrating improved performance in many cases. We are also able to verify probabilistic timed programs containing both real-time behaviour and infinite data variables, which these tools cannot handle.

1.1. Related work

Abstraction refinement for MDPs and related models is an active research field. In [6], techniques were proposed for abstracting MDPs using the notion of probabilistic simulation. Building on the same approach to abstraction, [11] developed a probabilistic version of the classic counterexample-guided abstraction refinement (CEGAR) method, which was then implemented in the tool PASS [9]. This verifies a probability-bounded reachability property using a refinement scheme based on probabilistic counterexamples and Craig interpolation. In contrast to the implementation of [6], PASS uses predicate abstraction, allowing it to analyse infinite-state models. More recent work [15] proposes an alternative probabilistic CEGAR technique using stochastic tree counterexamples; this applies to finite-state MDPs, on which properties are specified using simulation rather than reachability. However, all three methods [6,11,15] were applied to discrete-time models (MDPs), whereas our approach generalises to models with real-time behaviour. We provide an experimental comparison with PASS, for MDP models, in Section 4.

In [13], a quantitative abstraction refinement technique for MDPs was proposed, using a different form of abstraction based on stochastic games. This computes lower and upper bounds for reachability probabilities, the difference between which determines if further refinement is needed. The framework of [13] was subsequently applied to verification of C programs with probabilistic behaviour [12]. Later extensions to PASS also use game-based abstraction refinement [24]. In recent work [7], the abstraction frameworks of [13,24] were adapted to handle arbitrary abstract domains, illustrating cases where this can outperform predicate abstraction. As for [6,11,15] above, though, these methods [13,12,24,7] all focus on models with a discrete notion of time.

Probabilistic timed automata (PTAs) are a subclass of the probabilistic timed programs (PTPs) that we target in this paper, since only the latter allows arbitrary (infinite) data variables. For PTAs, several verification techniques exist. Most relevant here is [16], which extends the abstraction refinement framework of [13] mentioned above, to PTAs, by using zones to represent abstract states. Other possibilities include the digital clocks discretisation [19] and backwards reachability [21]. The probabilistic model checker PRISM [18] supports verification of PTAs, using either [16] or [19]. In [16], abstraction refinement was shown to outperform the other available techniques. Subsequently, an optimised version of backwards reachability, implemented in the tool FORTUNA [3], was shown to exhibit superior performance on various examples. We compare the performance of our approach to both PRISM and FORTUNA in Section 4.

Several PTA verification tools do support PTAs with data variables, but they are required to be finite. This includes PRISM, discussed above, `mcpta` [10], which translates the modelling language Modest to PRISM using [19], and UPPAAL PRO, which computes maximum reachability probabilities for PTAs by progressively partitioning the state space into sets of zones.

The closest approaches to the one presented here are [17] and [8]. In [17], an extension of the game-based abstraction refinement framework of [13] is defined for PTPs, but not implemented. This defines abstractions as stochastic games, rather than MDPs as in our approach. In recent work [8], PTPs (there called variable-decorated PTAs) are verified using a combination of discretisation via digital clocks [19] and predicate abstraction methods from PASS [24]. Our approach avoids the use of discretisation by using zones and aims to improve efficiency by using local refinement and simplification operations to reduce the size of abstractions. The implementation of [8] is not currently available; we give a brief, indirect comparison of results in Section 4.

2. Preliminaries

We assume a set $\mathcal{V}$ of *variables*, ranging over a domain D defined by a theory T (linear integer arithmetic in our examples). We require satisfiability of quantifier-free formulae in T to be decidable. The set of *assertions* over $\mathcal{V}$, i.e., (conjunctive) quantifier-free formulae in T, is denoted by $Asrt(\mathcal{V})$, and $Val(\mathcal{V})$ is the set of *valuations* of $\mathcal{V}$, i.e., functions $u : \mathcal{V} \to D$. We use $Assn(\mathcal{V})$ for the set of *assignments* over $\mathcal{V}$, given by a term r_x for each $x \in \mathcal{V}$. The result of applying assignment f to a valuation u is $f(u)$, given for each $x \in \mathcal{V}$ by $f(u)(x) = u(r_x)$. Given an assignment f and an assertion φ, the composition $\varphi \circ f$ is defined by $(\varphi \circ f)(u) \equiv \varphi(f(u))$.

For a set S, we use $\mathcal{P}(S)$ to denote the set of subsets of S and $\mathcal{D}(S)$ for the set of discrete probability distributions over S, i.e. finite-support functions $\Delta : S \to [0,1]$ such that $\sum_{s \in S} \Delta(s) = 1$. A distribution $\Delta \in \mathcal{D}(S)$ with support $\{s_1, \ldots, s_n\}$ and $\Delta(s_j) = \lambda_j$ will also be written $\lambda_1 s_1 + \cdots + \lambda_n s_n$.

2.1. Clocks

We use a set $\mathcal{X}$ of *clock* variables to represent the time elapsed since the occurrence of various events. The set of *clock valuations* is $\mathbb{R}_{\geqslant 0}^{\mathcal{X}} = \{v : \mathcal{X} \to \mathbb{R}_{\geqslant 0}\}$. For any clock valuation v and $\delta \in \mathbb{R}_{\geqslant 0}$, the *delayed* valuation $v + \delta$ is defined by $(v + \delta)(x) = v(x) + \delta$ for all $x \in \mathcal{X}$. For a subset $Y \subseteq \mathcal{X}$, the valuation $v[Y{:=}0]$ is obtained by setting all clocks in Y to 0, i.e., $v[Y{:=}0](x)$ is 0 if $x \in Y$ and $v(x)$ otherwise. The valuation $\mathbf{0}$ has all clocks set to 0.

Please cite this article in press as: K. Dräger et al., Local abstraction refinement for probabilistic timed programs, Theoretical Computer Science (2013), http://dx.doi.org/10.1016/j.tcs.2013.07.013