Information and Computation 243 (2015) 26–36

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Reachability in two-clock timed automata is PSPACE-complete ☆

John Fearnley [a,*], Marcin Jurdziński [b]

[a] Department of Computer Science, University of Liverpool, UK
[b] Department of Computer Science, University of Warwick, UK

ARTICLE INFO

Article history:
Received 2 September 2013
Available online 11 December 2014

Keywords:
Timed automata
Counter automata
PSPACE-complete

ABSTRACT

Recently, Haase, Ouaknine, and Worrell have shown that reachability in two-clock timed automata is log-space equivalent to reachability in bounded one-counter automata. We show that reachability in bounded one-counter automata is PSPACE-complete.

1. Introduction

Timed automata [1] have been successfully and widely used in the analysis and verification of real time systems. A timed automaton is a non-deterministic finite automaton that is equipped with a number of real-valued *clocks*, which allow the automaton to measure the passage of time.

Perhaps the most fundamental problem for timed automata is the *reachability* problem: given an initial state, can we perform a sequence of transitions in order to reach a specified target state? In their foundational paper on timed automata [1], Alur and Dill showed that this problem is PSPACE-complete. To show hardness for PSPACE, their proof starts with a linear bounded automaton (LBA), which is a non-deterministic Turing machine with a finite tape of length n. They produced a timed automaton with $2n + 1$ clocks, and showed that the timed automaton can reach a specified state if and only if the LBA halts.

However, the work of Alur and Dill did not address the case where the number of clocks is small. This was rectified by Courcoubetis and Yannakakis [2], who showed that reachability in timed automata with only three clocks is still PSPACE-complete. Their proof cleverly encodes the tape of an LBA in a single clock, and then uses the two additional clocks to perform all necessary operations on the encoded tape. In contrast to this, Laroussinie et al. have shown that reachability in one-clock timed automata is complete for NLOGSPACE, and therefore no more difficult than computing reachability in directed graphs [3].

The complexity of reachability in two-clock timed automata has been left open. So far, the best lower bound was given by Laroussinie et al., who gave a proof that the problem is NP-hard via a very natural reduction from subset-sum [3]. Moreover, the problem lies in PSPACE, because reachability in two-clock timed automata is no harder than reachability in three-clock timed automata. However, the PSPACE-hardness proof of Courcoubetis and Yannakakis seems to fundamentally

☆ This work was supported by EPSRC grants EP/H046623/1 *Synthesis and Verification in Markov Game Structures*, EP/L011018/1 *Algorithms for Finding Approximate Nash Equilibria*, and EP/D063191/1 *The Centre for Discrete Mathematics and its Applications (DIMAP)*.

* Corresponding author.
E-mail addresses: john.fearnley@liv.ac.uk (J. Fearnley), marcin.jurdzinski@dcs.warwick.ac.uk (M. Jurdziński).

http://dx.doi.org/10.1016/j.ic.2014.12.004
0890-5401/

require three clocks, and does not naturally extend to the two-clock case. Naves [4] has shown that several extensions to two-clock timed automata lead to PSPACE-completeness, but his work does not advance upon the NP-hardness result for unextended two-clock timed automata.

In a recent paper, Haase et al. have shown a link between reachability in timed automata and reachability in *bounded counter automata* [5]. A bounded counter automaton is a non-deterministic finite automaton equipped with a set of counters, and the transitions of the automaton may add or subtract arbitrary integer constants to the counters. The state space of each counter is bounded by some natural number b, so the counter may only take values in the range $[0, b]$. Moreover, transitions may only be taken if they do not increase or decrease a counter beyond the allowable bounds. This gives these seemingly simple automata a surprising amount of power, because the bounds can be used to implement inequality tests against the counters.

Haase et al. show that reachability in two-clock timed automata is log-space equivalent to reachability in bounded *one*-counter automata. Reachability in bounded one-counter automata has also been studied in the context of one-clock timed automata with energy constraints [6], where it was shown that the problem lies in PSPACE and is NP-hard. It has also been shown that the reachability problem for *unbounded* one-counter automata is NP-complete [7], but the NP membership proof does not seem to generalise to bounded one-counter automata. Haase et al. also showed that reachability in bounded *two*-counter automata is log-space equivalent to reachability in *three*-clock timed automata, and that therefore, for any $k > 1$, reachability in bounded k-counter automata is PSPACE-complete [5].

Our contribution We show that reachability in bounded one-counter automata is PSPACE-complete. Therefore, we resolve the complexity of reachability in two-clock timed automata. Our reduction uses two intermediate steps: *subset-sum games* and *safe counter-stack automata*.

Counter automata are naturally suited for solving subset-sum problems, so our reduction starts with a quantified version of subset-sum, which we call subset-sum games. One interpretation of satisfiability for quantified boolean formulas is to view the problem as a game between an *existential* player and a *universal* player. The players take turns to set their propositions to true or false, and the existential player wins if and only if the boolean formula is satisfied. Subset-sum games follow the same pattern, but apply it to subset-sum: the two players alternate in choosing numbers from sets, and the existential player wins if and only if the chosen numbers sum to a given target. Previous work by Travers can be applied to show that subset-sum games are PSPACE-complete [8].

We reduce subset-sum games to reachability in bounded one-counter automata. However, we will not do this directly. Instead, we introduce safe counter-stack automata, which are able to store multiple counters, but have a stack-like restriction on how these counters may be accessed. These automata are a convenient intermediate step, because having access to multiple counters makes it easier for us to implement subset-sum games. Moreover, the stack based restrictions mean that it is relatively straightforward to show that reachability in safe counter-stack automata is reducible, in logarithmic space, to reachability in bounded one-counter automata, which completes our result.

2. Subset-sum games

A subset-sum game is played between an *existential* player and a *universal* player. The game is specified by a pair (ψ, T), where $T \in \mathbb{N}$, and ψ is a list:

$$\forall\{A_1, B_1\}\exists\{E_1, F_1\} \ldots \forall\{A_n, B_n\}\exists\{E_n, F_n\},$$

where A_i, B_i, E_i, and F_i, are all natural numbers encoded in binary.

The game is played in rounds. In the first round, the universal player chooses an element from $\{A_1, B_1\}$, and the existential player responds by choosing an element from $\{E_1, F_1\}$. In the second round, the universal player chooses an element from $\{A_2, B_2\}$, and the existential player responds by choosing an element from $\{E_2, F_2\}$. This pattern repeats for rounds 3 through n. Thus, at the end of the game, the players will have constructed a sequence of numbers, and the existential player wins if and only if the sum of these numbers is T.

Formally, the set of *plays* of the game is the set:

$$\mathcal{P} = \prod_{1 \leq j \leq n} \{A_j, B_j\} \times \{E_j, F_j\}.$$

A play $P \in \mathcal{P}$ is winning for the existential player if and only if $\sum P = T$.

A strategy for the existential player consists of a list of functions $\mathsf{s} = (s_1, s_2, \ldots, s_n)$, where each function s_i dictates how the existential player should play in the ith round of the game. Thus, each function s_i is of the form:

$$s_i : \prod_{1 \leq j \leq i} \{A_j, B_j\} \to \{E_i, F_i\}.$$

This means that the function s_i maps the first i moves of the universal player to a decision for the existential player in the ith round. Note that the function s_i does not need to take the previous moves of existential player as inputs, because these moves are entirely determined by the previous moves of the universal player and the functions s_j with $j < i$.