Automated Verification of Infinite State Concurrent Systems

Piotr Dembiński\(^1\), Wojciech Penczek\(^1\), and Agata Półrola\(^2\)

\(^1\) Institute of Computer Science, PAS, Ordonia 21, 01-237 Warsaw, Poland
\{piotr, penczek\}@ipipan.waw.pl

\(^2\) Faculty of Mathematics, University of Lodz, Banacha 22, 90-238 Lodz, Poland
polrola@math.uni.lodz.pl

Abstract. The paper shows how to use partitioning techniques to generate abstract state spaces (models) preserving similarity and injectivity. Since these relations are weaker than bisimilarity and surjectivity, our algorithms generate smaller models. This method can be applied for improving several existing partitioning algorithms used for generating finite models of concurrent programs, Time Petri Nets and Timed Automata.

1 Introduction

Automated verification of concurrent programs is usually translated to the problem of checking whether a finite state graph (model) corresponding to a program satisfies a given property (formula). The complexity of checking this strongly depends on the size of the model. Moreover, in many cases (as for timed systems) state spaces are infinite. Therefore, one tries to construct models with a reduced, finite number of states, preserving properties to be verified. As it is not always known how to generate minimal models for arbitrary formulas, models preserving whole subclasses of CTL\(^*\) are generated instead. This is done by identifying the equivalence (on models) preserving a selected subclass of the logic and then generating a minimal model equivalent to the original one. Standard minimization algorithms are based on bisimulation (preserving whole CTL\(^*\)), similarity (preserving ACTL\(^*\)), and surjectivity (preserving LTL). In this paper we show that the standard minimization algorithms can be adapted for generating smaller models based on simulation (preserving ACTL\(^*\)) and injectivity (preserving LTL).

2 Related Work

The original partitioning algorithm for generating minimal bisimulating models has been introduced in [BFH\(^+\)92] and then extended to timed systems in [ACD\(^+\)92a, ACD\(^+\)92b, TY96]. Algorithms for computing similarity relations of the states of labelled graphs have been considered in [HHK95] and [BG00]. These algorithms could be also exploited for defining finite simulating models, but would give unnecessarily large state spaces in comparison to ours. This
follows from the fact that we do not require our simulating models to be quotient structures of the simulation relation defined on the states of the original (concrete) model. Instead, we exploit the notion of similarity between models, defined in [GL91,GKP92]. For all examples we have considered the simulating models generated by the algorithm of [HHK95,BG00] would be equal to the bisimulating models.

3 Models and Their Reducts

A Kripke structure is a triple $K = (S,s_0,\rightarrow)$, where $S$ is a set of states, $s_0 \in S$ is an initial state, and $\rightarrow \subseteq S \times L \times S$ is a labelled total transition relation for some fixed set of labels $L$. A model is a pair $M = (K,V)$, where $V$ is a valuation, i.e., $V : S \rightarrow 2^{PV}$ for some fixed set of propositional variables $PV$.

Definition 1. A graph $G = (W,w_0,\rightarrow)$ is a set-graph over $K$ and $L$ if: each node $w \in W$ is a set of states of $S$ and $s_0 \in w_0 \in W$; $(\forall w_1, w_2 \in W)(\forall a \in L)$ $w_1 \xrightarrow{a} w_2$ iff $(\exists s \in w_1)(\exists s' \in w_2)$ s.t. $s \xrightarrow{a} s'$. A set-graph is complete iff $(\forall s \in S)(\exists w \in W) s \in w$.

Usually, two types of set-graphs are considered [TY96,ACD+92a,ACD+92b]:

Definition 2 (bisimulating and surjective set-graphs).

- A set-graph $G$ is bisimulating iff
  $$(\forall w_1, w_2 \in W)(\forall a \in L) \text{ if } w_1 \xrightarrow{a} w_2 \text{ then } (\forall s \in w_1)(\exists s' \in w_2) \text{ s.t. } s \xrightarrow{a} s'$$

- A set-graph $G$ is surjective iff
  $$(\forall w_1, w_2 \in W)(\forall a \in L) \text{ if } w_1 \xrightarrow{a} w_2 \text{ then } (\forall s \in w_1)(\exists s' \in w_2) \text{ s.t. } s \xrightarrow{a} s'$$

Bisimulating set-graphs preserve branching time properties of Kripke structures, whereas finite surjective set-graphs preserve reachability and, with an additional condition (see FTP defined in the next section), also linear time properties.

4 Temporal Logics: CTL*, ACTL*, and LTL

Syntax and semantics of CTL*. The set of formulas of CTL* can be defined inductively as follows:

$$\varphi := p \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X\varphi \mid A\varphi \mid \Until(\varphi,\varphi) \mid \Until_{\infty}(\varphi,\varphi)$$

The language of CTL* consists of all the state formulas, i.e., the formulas of the form $p$ for $p \in PV$, $A\varphi$ for each $\varphi$, and $\neg \varphi$, $\varphi \land \psi$ and $\varphi \lor \psi$ for state formulas $\varphi$, $\psi$. Let $\pi = s_0a_0s_1 \ldots (s_i \xrightarrow{a_i} s_{i+1}$, for all $i \geq 0$) be an infinite path in a model $M$ and $\pi_i$ the suffix $s_ia_is_{i+1} \ldots$ of $\pi$. The validity ($\models$) of the CTL* formulas is defined as follows:

- Proposition $p$ is valid in $s$ if $p \in V(s)$,
- A state formula $\varphi$ is valid for a path $\pi$ if it is valid in its starting state,
- $A\varphi$ is valid in $s$ if $\varphi$ is valid for all paths starting at $s$, 