Bounded Reachability Checking with Process Semantics*

Keijo Heljanko

Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O. Box 5400, FIN-02015 HUT, Finland
Keijo.Heljanko@hut.fi

Abstract. Bounded model checking has been recently introduced as an efficient verification method for reactive systems. In this work we apply bounded model checking to asynchronous systems. More specifically, we translate the bounded reachability problem for 1-safe Petri nets into constrained Boolean circuit satisfiability. We consider three semantics: process, step, and interleaving semantics. We show that process semantics has often the best performance for bounded reachability checking.

1 Introduction

Bounded model checking has been proposed as a verification method for reactive systems. The main idea is to look for counterexamples which are shorter than some fixed length for a given property. If a counterexample can be found, then the property does not hold for the system. If no counterexample can be found using this bound, usually the result is inconclusive. The decision procedure most often used in bounded model checking is propositional satisfiability. Given the transition relation of the reactive system to be model checked, the property, and the bound $n$, the transition relation and property are “unrolled” $n$ times to obtain a propositional formula which is satisfiable iff there is a counterexample with bound $n$. The implementation ideas are quite similar to procedures used in AI planning.

In this work we apply bounded model checking to asynchronous systems. More specifically, we translate the bounded reachability problem for 1-safe Petri nets into constrained Boolean circuit satisfiability. This work can be seen as a continuation of the work done in [9]. There we discuss using the step and interleaving semantics for bounded reachability, while the formalism into which the problem is translated being logic programs with stable model semantics. The main contribution of this paper is that we show that the so called process semantics of Petri nets [12] can be used to improve the efficiency of bounded model checking. Namely, also the process semantics can be efficiently encoded into constrained Boolean circuits.

* The financial support of the Academy of Finland (Projects 43963 and 47754), and Tekniikan Edistämissäätiö foundation are gratefully acknowledged.
As an additional contribution we report on an implementation called punroll, which uses the BCSat constrained Boolean circuit satisfiability checker to check whether the generated constrained circuit is satisfiable, thus solving the bounded reachability problem.

The structure of the rest of the paper is the following. First we introduce Petri nets and the three different semantics in Sect. 2. Then we shortly introduce constrained Boolean circuits in Sect. 3, and in Sect. 4 show how the bounded reachability problem can be translated into them. After that we discuss our implementation and experiments in Sect. 5, and finish with conclusions in Sect. 6.

2 Petri Nets

We will now introduce Petri nets. A net is a triple \((P, T, F)\), where \(P\) and \(T\) are disjoint sets of places and transitions, respectively, and \(F\) is a function \((P × T) ∪ (T × P) → \{0, 1\})\). Places and transitions are generically called nodes. If \(F(x, y) = 1\) then we say that there is an arc from \(x\) to \(y\). The preset of a node \(x\), denoted by \(\bullet x\), is the set \(\{y ∈ P ∪ T | F(y, x) = 1\}\). The postset of \(x\), denoted by \(x^*\), is the set \(\{y ∈ P ∪ T | F(x, y) = 1\}\). In this paper we consider only finite nets in which every transition has a nonempty preset and a nonempty postset. A marking of a net \((P, T, F)\) is a mapping \(P → \mathbb{N}\) (where \(\mathbb{N}\) denotes the natural numbers including 0). We identify a marking \(M\) with the multiset containing \(M(p)\) copies of \(p\) for every \(p ∈ P\). For instance, if \(P = \{p_1, p_2\}\) and \(M(p_1) = 1, M(p_2) = 2\), we write \(M = \{p_1, p_2, p_2\}\). A 4-tuple \(Σ = (P, T, F, M_0)\) is a net system if \((P, T, F)\) is a net and \(M_0\) is a marking of \((P, T, F)\) (called the initial marking of \(Σ\)). We will use as a running example the net system in Fig. 1.

2.1 Step Semantics

To save some space, we define the behavior of a net system through step semantics. The (usual) interleaving semantics will then be defined later based on this more general concept.

A step is a non-empty set of transitions \(S ⊆ T\). We denote a step by \([S]\). A marking \(M\) enables a step \(S\) if for all \(p ∈ P\) it holds that \(M(p) ≥ ∑_{t ∈ S} F(p, t)\). If the step \(S\) is enabled at \(M\), then it can fire or occur, and its occurrence leads to a new marking \(M’\) defined as \(M’(p) = M(p) + ∑_{t ∈ S} (F(t, p) − F(p, t))\) for every place \(p ∈ P\). We denote this firing of a step by \(M[S]M’\).

A (possibly empty) sequence of steps \(σ = [S_0][S_1]⋯[S_{n−1}]\) is a step execution of the net system \(Σ = (P, T, F, M_0)\) if there exist markings \(M_1, M_2, \ldots, M_n\) such that \(M_0[S_0]M_1[S_1]⋯M_{n−1}[S_{n−1}]M_n\). The marking reached by the occurrence of \(σ\) is \(M_n\). A marking \(M\) is a reachable marking if there exists a step execution \(σ\) such \(M\) is reached by the occurrence of \(σ\). A marking \(M\) is reachable with bound \(n\) if there exists a step execution \(σ\) consisting of (exactly) \(n\) steps.

\(^1\) We only consider a class of nets where the transitions cannot be self-concurrent.

Therefore a set suffices and multisets, i.e., bags are not needed.