Non-Deterministic Automata for Efficient Verification of Infinite-State Systems

Pierre Ganty* Laurent Van Begin †
{pganty, lvbegin}@ulb.ac.be
Université Libre de Bruxelles
Boulevard du triomphe, c.p. 212
1050 Bruxelles, Belgium

Many systems like multi-threaded programs and communication protocols are very complex to devise, hence error prone. In our setting, those systems are formally modeled and then abstracted — applying a so-called *counting abstraction* — into infinite state *counter systems*, see for instance [5]. A counter system is an automaton manipulating (possibly) unbounded counters. Consider the example of Figure 1(a) that shows a system composed of one server and n clients (where n is undefined) running in parallel and communicating through synchronization labels (*rendez-vous*). The counter system of Figure 1(b) is obtained by applying a counting abstraction on the system of Figure 1(a); i.e. the clients are abstracted into two counters x_1 and x_2 that define the number of clients that are in the location 1 and in the location 2, respectively.

Then, given a counter system $\mathcal{M}$, we apply *Model checking* methods to verify automatically that some important properties hold on $\mathcal{M}$. Those automatic methods manipulate (possibly) infinite sets of configurations of counter systems, hence a suited representation must be defined to handle such sets.

We focus on a particular class of sets that can be described by formulas of the so-called $\mathcal{I}$-logic defined by the following grammar:

$$\Phi ::= x_i \in [a, b] \mid \Phi \wedge \Phi \mid \Phi \vee \Phi$$
$$\text{where } a \in \mathbb{N},\ b \in \mathbb{N} \cup \{+\infty\}$$

Each variable x_i is associated to a counter and each atom $x_i \in [a, b]$ constraints the values of x_i within the interval $[a, b]$. Boolean connectors have the usual semantics and the set of configurations represented by the $\mathcal{I}$-formula Φ is the set $[\![\Phi]\!]$ of all the configurations satisfying Φ.

Given a counter machine $\mathcal{M}$, a set of initial configurations $[\![\Phi_I]\!]$ and a set of target configurations $[\![\Phi_F]\!]$, an important verification problem is to decide if there exists a sequence of transitions of $\mathcal{M}$ that leads to a configuration in $[\![\Phi_F]\!]$ from a configuration in $[\![\Phi_I]\!]$. This problem is called the *reachability problem*.

It turns out that in the general case the reachability problem is undecidable unless the formula Φ_F has all of its atoms $x_i \in [a, b]$ such that $b = +\infty$ and the counter system $\mathcal{M}$ is *monotonic* [3].

When applying model checking methods to solve the reachability problem, formulas are maintained in a disjunctive form to ease their manipulation. However, we have to face off the *symbolic state explosion problem*. This problem corresponds to a blow up of the size of the formulas. To overcome it we introduce a symbolic representation of $\mathcal{I}$-formulas called *Interval Sharing Tree (IST)* [4]. ISTs are directed acyclic graph-based data structures. The nodes of an IST are partitioned into layers, each corresponding to a single variable of the formula. Each node defines an atom $x_i \in [a, b]$ such that $[a, b]$ is the label of the node and x_i is defined by the layer to which the node belongs. In an IST, each path defines a conjunction of atoms. Figure 1(c) shows an IST with two layers such that to the first (resp. second) one we associate the variable x_1 (resp. x_2). The path passing through the nodes labeled with the intervals $[0, 1]$ and $[3, 4]$ defines the constraint $x_1 \in [0, 1] \wedge x_2 \in [3, 4]$. The set of configurations defined by an IST S is the set of configurations satisfying at least one formula associated to a path of S.

Given a $\mathcal{I}$-formula Φ in disjunctive form and the IST S_Φ whose language coincides with $[\![\Phi]\!]$, the salient features of S_Φ are:

- the size of S_Φ may be exponentially smaller than the size of Φ. Here $|S_\Phi|$ is the number of nodes plus the number of edges of S_Φ and $|\Phi|$ is the number of atoms in Φ.

- S_Φ is a non-deterministic automaton[1] whereas

*This author is funded by the FRIA grant.
†This author is funded by a "First Europe" grant EPH3310300R0012 of the Walloon Region.

[1] by non-deterministic we mean that one configuration accepted by S_Φ can be matched by several paths.

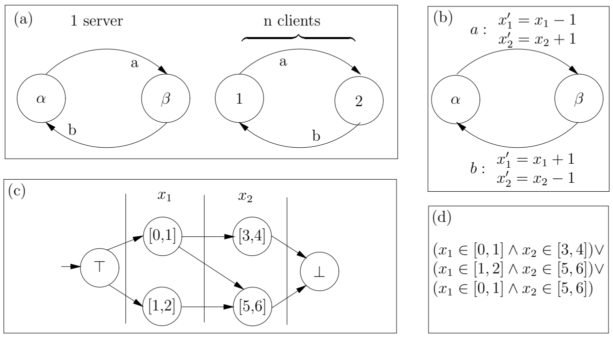

Figure 1: (a) a server and n clients running in parallel ($n \geq 1$); (b) the model obtained by applying a counting abstraction from (a); (c) an IST and (d) a $\mathcal{I}$-formula represented by the IST shown in (c).

other approaches facing to the state explosion problem traditionally use deterministic automata (*e.g.* BDD's like data structure).

We implemented a prototype that uses ISTs and some heuristics (see [1]) for the verification of reachability properties on extended Petri nets, a monotonic sub class of counter systems (see [2] for more details). We applied it on a large set of Petri net examples and counting abstractions of multi-threaded Java programs. Some results that we obtained are reported in Table 1. The column C, T and Ex gives, respectively, the number of counters, transitions and the execution times for each of our examples. The last column R says if the reachability properties to test hold ($\surd$) or not ($\times$). Those examples are explained more exhaustively in [2, 5].

Case study	C	T	Ex	R
Swimming Pool	7	6	$\backsimeq$ 0s	$\surd$
Manufacture_1	7	6	20s	$\surd$
Manufacture_2	25	14	0.1s	$\surd$
$^{\text{Java}}$I/D	32	28	1.5s	$\surd$
$^{\text{Java}}$P/C_1	44	37	4.2s	$\surd$
$^{\text{Java}}$P/C_2	20	16	$\backsimeq$ 0s	$\times$
$^{\text{Java}}$dl	48	42	14.2s	$\times$

Table 1: Results obtained on a AMD 900MHz with 1GB.

References

[1] G. Delzanno, J.-F. Raskin, and L. Van Begin. Attacking Symbolic State Explosion. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102 of *LNCS*, pages 298–310. Springer, 2001.

[2] G. Delzanno, J-F. Raskin, and L. Van Begin. Towards the Automated Verification of Multithreaded Java Programs. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2002)*, volume 2280 of *LNCS*, pages 173–187. Springer, 2002.

[3] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[4] P. Ganty. Algorithmes et Structures de Données Efficaces Pour la Manipulation de Contraintes Sur les Intervalles. Master's thesis, Université Libre de Bruxelles, 2002.

[5] L. Van Begin. *Efficient Verification of Counting Abstractions for Parametric systems.* PhD thesis, Université Libre de Bruxelles, Belgium, 2003.