FAST: Fast Acceleration of Symbolic Transition systems (ACSD tool presentation)

Sébastien Bardin[1], Alain Finkel[1], Jérôme Leroux[2], and Laure Petrucci[3]

[1] LSV - (CNRS UMR 8643, ENS de Cachan), France, {bardin,finkel}@lsv.ens-cachan.fr
[2] IRISA - (INRIA Rennes), France, jleroux@irisa.fr
[3] LIPN - (CNRS UMR 7030, Univ. Paris 13), France, petrucci@lipn.univ-paris13.fr

Abstract. FAST is a tool for the analysis of large or even infinite systems. This paper describes the underlying theory, the architecture choices that have been made in the tool design. The user must provide a model to analyse, the property to check and a computation policy. Several such policies are proposed as a standard in the package, others can be added by the user. FAST capabilities are compared with those of other tools. A range of case studies from the literature has been investigated.

1 Introduction

Model-checking is a wide-spread technique in critical systems verification. Several efficient model-checkers, such as SMV [SMV], SPIN [SPI] or DESIGN/CPN [CPN], are available. However, these tools are restricted to finite systems whereas many real systems are infinite, because of parameters or unbounded data structures.

FAST [BFLP03,FAS] is a tool designed to allow automatic verification of systems modeled by automata augmented with (unbounded) integer variables (*extended counter automata*). The main issue addressed by FAST is the computation of the *exact* (*infinite*) set of configurations reachable from a given set of initial configurations. Let us recall that verification of safety properties can be reduced to reachability of a given configuration from a set of initial configurations.

A lot of properties are in general undecidable, but there are two ways to deal with undecidability. The first one is to consider decidable subclasses, thus reducing the expressiveness of the model, while the second one is to accept only a semi-algorithm, which does not terminate in the general case but which is expected to terminate in most practical cases. We follow the second approach. The techniques used in FAST are based on *acceleration* [BF04,FL02,ABS01,WB00]. It comes down to computing the (exact) effect of iterating a control loop of *an arbitrary length* (*cycle*). These cycles are automatically chosen. Both forward and backward reachability are allowed. FAST works on *linear systems*, i.e. *finite sets of linear functions* whose definition domains are defined by a *Presburger formula over non-negative integers*. Most systems with integer variables can be described by such a system. In [FL02], it is proved that for

Case study	variables	transitions	time (s)	memory (MB)
Bounded Petri Nets				
Producer/Consumer	5	3	0.41	2.37
Lamport ME	11	9	2.70	2.88
Dekker ME	22	22	21.72	5.48
RTP	9	12	2.24	2.76
Peterson ME	14	12	4.97	3.78
Reader/Writer	13	9	9.68	23.14
Petri Nets				
CSM	13	13	45.57	6.31
FMS	22	20	157.48	8.02
Multipoll	17	20	22.96	5.13
Kanban	16	16	10.43	6.54
Mesh2x2	32	32	$\geq$ 1800	-
Mesh3x2	52	54	$\geq$ 1800	-
Manufacturing system	7	6	$\geq$ 1800	-
Manufacturing system (check deadlock freedom)	13	6	$\geq$ 1800	-
PNCSA	31	38	$\geq$ 1800	-
extended ReaderWriter	24	22	$\geq$ 1800	-
SWIMMING POOL	9	6	111	29.06
Petri Nets with Transfer Arcs				
Last-in First-served	17	10	1.89	2.74
Esparza-Finkel-Mayr	6	5	0.79	2.55
Broadcast Protocols				
Inc/Dec	32	28	$\geq$ 1800	-
Producer/Consumer with Java threads - 2	18	14	13.27	3.81
Producer/Consumer with Java threads - N	18	14	723.27	12.46
2-Producer/2-Consumer with Java threads	44	38	$\geq$ 1800	-
Central Server system	13	8	20.82	6.83
Consistency Protocol	12	8	275	7.35
M.E.S.I. Cache Coherence Protocol	4	4	0.42	2.44
M.O.E.S.I. Cache Coherence Protocol	4	5	0.56	2.49
Synapse Cache Coherence Protocol	3	3	0.30	2.23
Broadcast Protocols				
Illinois Cache Coherence Protocol	4	6	0.97	2.64
Berkeley Cache Coherence Protocol	4	3	0.49	2.75
Firefly Cache Coherence Protocol	4	8	0.86	2.59
Dragon Cache Coherence Protocol	5	8	1.42	2.72
Futurebus+ Cache Coherence Protocol	9	10	2.19	3.38
Others				
lift controller - N	4	5	4.56	2.90
bakery	8	20	$\geq$ 1800	-
barber m4	8	12	1.92	2.68
ticket 2i	6	6	0.88	2.54
ticket 3i	8	9	3.77	3.08
TTP	10	17	1186.24	73.24
TTP (ad hoc strategy)	10	17	246.67	72.87

Fig. 1. Results using an Intel Pentium 933 Mhz with 512 Mbytes of memory

linear systems whose associated square matrices generate a finite multiplicative monoid – namely *finite linear systems*, acceleration of a loop terminates. It turns out that most integer variables systems appear to be finite linear systems. Even though termination is not guaranteed, in practice, FAST deals with a large amount of examples of our extended counter automata model (see section 2). We believe that Presburger arithmetic is sufficient to model these problems and that most systems are effectively computable.

2 Results

FAST has been applied to a large number of examples (about 40), ranging from Petri nets to abstract multi-threaded JAVA programs, mainly taken from [Del]. The table in figure 1 presents the number of variables and transitions of the models, the computation time, the memory consumption. Despite the number of variables and transitions of some examples, they are rather non-trivial (see for instance the Swimming Pool protocol studied in [FO97]). The tool computes efficiently the reachability set of about 80% of these examples. It proves that choices made during FAST design, like having only a semi-algorithm or restricting FAST to non-negative integers, are sound for practical infinite systems verification. Moreover, most of these examples require only a basic predefined strategy (a forward search), thus only little input from the user. For the particular case of the TTP protocol (a complex group membership protocol), a more elaborate strategy was also tested leading to considerable decrease in computation time [BFL04].

Tool FAST and many examples are freely downloadable on the web page [FAS].

References

[ABS01] A. Annichini, A. Bouajjani, and M. Sighireanu. TREX: a Tool for Reachability analysis of Complex Systems. In *Proc. 13th Int. Conf. Computer Aided Verification (CAV'2001), Paris, France, July 2001*, volume 2102 of *LNCS*, pages 368–372. Springer, 2001.

[BF04] S. Bardin and A. Finkel. Composition of accelerations to verify infinite heterogeneous systems. In *Proc. 2nd Int. Symp. Automated Technology for Verification and Analysis (ATVA 2004), Taipei, Taiwan, Nov. 2004*, volume 3299 of *LNCS*, pages 248–262. Springer, 2004.

[BFL04] S. Bardin, A. Finkel, and J. Leroux. Faster acceleration of counter automata. In *Proc. 10th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004) Barcelona, Spain, Mar. 2004*, volume 2988 of *LNCS*, pages 576–590. Springer, 2004.

[BFLP03] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. FAST: Fast Acceleration of Symbolic Transition systems. In *Proc. 15th Int. Conf. Computer Aided Verification (CAV'2003), Boulder, CO, USA, July 2003*, volume 2725 of *LNCS*, pages 118–121. Springer, 2003.

[CPN] DESIGN/CPN *online*. `http://www.daimi.au.dk/designCPN`.

[Del] G. Delzanno. *Home Page – Giorgio Delzanno*. `http://www.disi.unige.it/person/DelzannoG/`.

[FAS] FAST *homepage*. `http://www.lsv.ens-cachan.fr/fast/`.

[FL02] A. Finkel and J. Leroux. How to compose Presburger-accelerations: Applications to broadcast protocols. In *Proc. 22nd Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'2002), Kanpur, India, Dec. 2002*, volume 2556 of *LNCS*, pages 145–156. Springer, 2002.

[FO97] L. Fribourg and H. Olsén. Proving safety properties of infinite state systems by compilation into Presburger arithmetic. In *Proc. 8th Int. Conf. Concurrency Theory (CONCUR'97), Warsaw, Poland, Jul. 1997*, volume 1243 of *LNCS*, pages 213–227. Springer, 1997.

[SMV] SMV *homepage*. `http://www-cad.eecs.berkeley.edu/~kenmcmil/`.

[SPI] SPIN *homepage*. `http://spinroot.com/spin/`.

[WB00] P. Wolper and B. Boigelot. On the construction of automata from linear arithmetic constraints. In *Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000), Berlin, Germany, Mar.-Apr. 2000*, volume 1785 of *LNCS*, pages 1–19. Springer, 2000.