USING SHARING TREES IN THE AUTOMATED ANALYSIS OF REAL-TIME SYSTEMS WITH DATA.

David Kendall William Henderson Adrian Robson

Department of Computing and Mathematics, University of Northumbria at Newcastle, Ellison Building, Newcastle upon Tyne, NE1 8ST

Abstract

Reachability analysis and model checking of timed automata are now well-established techniques in the analysis of real-time control systems. The major limiting factor in their use, from a technical point of view, remains the state explosion problem. Symbolic representation of the state space often allows for the analysis of much larger systems than the point-wise representation which is common in enumerative analysis. In particular, the use of rooted, ordered binary decision diagrams (ROBDDs) has been successful, mainly in the analysis of hardware systems where the need for a compact representation of boolean functions is prevalent. However in software systems, it is often desirable to represent data types which are more complicated than booleans. The use of sharing trees [16], which eliminates the requirement to find a boolean encoding of all data types, may offer a more attractive alternative to ROBDDs in these circumstances. This paper considers the use of sharing trees in the context of automata derived from a timed algebra of asynchronous broadcasting systems. It suggests that an encoding of timing constraints may be more easily incorporated into a sharing tree representation of the state space than into one based on ROBDDs.

1 Introduction

This paper outlines an approach to the implementation of reachability analysis of timed automata derived from system models described using a timed process algebra. It suggests that the sharing tree data structure provides a compact encoding of both the discrete and continuous components of a state vector. The rest of the paper is organised as follows: section 2 briefly introduces *CANDLE* [10], a language for modelling broadcasting real-time systems, which includes many of the features which one wishes to handle in automated analysis; section 3 briefly outlines an approach to timed reachability analysis of *CANDLE* models and introduces the main data structures needed to support the analysis; sharing trees are introduced in section 4 and their application in reachability analysis is considered; related work is reviewed in section 5; section 6 concludes and proposes further work.

2 *CANDLE*

CANDLE is an expressive real-time language with a comparatively simple operational semantics. It has been developed for the modelling and analysis of real-time systems which communicate using Controller Area Network (CAN) [9]. At the heart of *CANDLE* is a core language, *bCANDLE* which is a timed process algebra allowing the expression of system models consisting of a set of asynchronous processes, each having its own local data state and communicating by sending and receiving messages on one or more *broadcast* channels. A process can engage in three different kinds of basic action: 1) sending a message on a channel; 2) receiving a message from a channel and 3) performing an operation in a bounded amount of time, possibly transforming its local data state on completion. More complex process behaviour can be defined using sequential composition, guards on data states, non-deterministic choice and interrupt. A *CANDLE* system consists of the asynchronous parallel composition of a fixed number of processes.

The semantics of a *CANDLE* system is given by associating with it a timed transition system. A *timed transition system* $\mathcal{S} = (\Sigma, \sigma^{\mathcal{I}}, \mathcal{L}, \longrightarrow)$ is a tuple in which Σ is the set of states, $\sigma^{\mathcal{I}} \in \Sigma$ is the initial state, $\mathcal{L} = \mathrm{A} \cup \mathbb{R}^{>0}$ is the set of labels consisting of *action* labels A and *time* labels $\mathbb{R}^{>0}$ and $\longrightarrow \subseteq \Sigma \times \mathcal{L} \times \Sigma$ is the transition relation which is required to be deterministic and continuous with respect to the passage of time. The semantics are given in detail in [10]. For the purposes of verifying a *CANDLE* system, we work with a timed safety automaton [8] having a bisimilar transition system. We introduce timed safety automata briefly below.

Let H be a set $\{h_0, h_1, \cdots, h_n\}$ of real-valued variables, called *clocks*, where each $h_i \in \mathrm{H}$ ranges over the non-negative reals $\mathbb{R}^+$. A *clock valuation* is a function $\mathbf{v} : \mathrm{H} \to \mathbb{R}^+$ which assigns a value in $\mathbb{R}^+$ to each clock in H. We assume that h_0 is given the value 0 by every clock valuation. We denote by $\mathbf{v}[\mathrm{H}' := 0]$, the clock valuation $\mathbf{v}'$ such that $\mathbf{v}'(h) = 0$ for all $h \in \mathrm{H}'$ and $\mathbf{v}'(h) = \mathbf{v}(h)$ otherwise. For $t \in \mathbb{R}^+$, we denote by $\mathbf{v} + t$ the clock valuation $\mathbf{v}'$ such that

$\mathbf{v}'(h) = \mathbf{v}(h) + t$ for all clocks in H. $\mathbf{0}$ is the clock valuation which assigns 0 to every clock. A *bound* over H is a constraint of the form $h_i - h_j \# c$ where $i, j \in \{0, \ldots, n\}$, $\# \in \{<, \leq\}$ and $c \in \mathbb{Z} \cup \infty$. A *clock constraint* is a conjunction of bounds. We denote the set of clock constraints over the clocks H by $\Phi(\mathrm{H})$.

A *timed safety automaton* (TSA) is a tuple $\mathcal{A} = (\mathrm{Q}, q^{\mathcal{I}}, \mathrm{A}, \mathrm{E}, \mathrm{H}, \mathrm{I})$ where: Q is a finite set of *control locations*, $q^{\mathcal{I}}$ is the initial control location, A is a finite set of event names, E is a finite set of edges where an edge is of the form (q, ϕ, a, X, q') where $q, q' \in \mathrm{Q}$ are control locations; $\phi \in \Phi(\mathrm{H})$ is a clock constraint, $a \in \mathrm{A}$ is an event name and $X \subseteq \mathrm{H}$ is a set of clocks to be reset; H is a finite set of clocks, and $\mathrm{I} : \mathrm{Q} \to \Phi(\mathrm{H})$ is a function which associates a *time progress* condition (or *invariant*) with each control location.

The semantics of the TSA $\mathcal{A} = (\mathrm{Q}, q^{\mathcal{I}}, \mathrm{A}, \mathrm{E}, \mathrm{H}, \mathrm{I})$ is given by the timed transition system $\mathcal{T}[\![A]\!] = (\Sigma, \sigma^{\mathcal{I}}, \mathcal{L}, \longrightarrow)$ where Σ is the set of pairs $(q, \mathbf{v})$ such that $q \in \mathrm{Q}$ is a location of A and $\mathbf{v}$ is a clock valuation for H which satisfies the invariant $\mathrm{I}(q)$; $\sigma^{\mathcal{I}} = (q^{\mathcal{I}}, \mathbf{0})$ is the initial state; $\mathcal{L} = \mathrm{A} \cup \mathbb{R}^{>0}$ is the set of labels; $\longrightarrow \subseteq \Sigma \times \mathcal{L} \times \Sigma$ is the transition relation containing transitions as follows:

- *Time transitions*: A state can change due to the elapse of time. There is a transition $(q, \mathbf{v}) \xrightarrow{t} (q, \mathbf{v} + t)$ if for all $t' \leq t$, $\mathbf{v} + t'$ satisfies the invariant $\mathrm{I}(q)$.
- *Event transitions:* A state can change by moving location. For each state $(q, \mathbf{v}) \in \Sigma$, if there is an edge $(q, \phi, a, X, q') \in \mathrm{E}$ such that $\mathbf{v}$ satisfies ϕ, then there is a transition $(q, \mathbf{v}) \xrightarrow{a} (q', \mathbf{v}[X := 0])$.

For a TSA derived from a *CANDLE* system model, a location comprises information about: 1) the currently active process terms, given as a set of integers $\{p_1, p_2, \ldots, p_c\}$ for some variable number c where each p_i defines a marked place in a net-like representation of the process term for the model; 2) the values of all data variables, given as a sequence of values $\langle v_1, v_2, \ldots, v_d \rangle$ for some fixed number d of variables; and 3) the state of the network channels, given as a sequence of pairs (s, m) where s gives the status of the channel, *free, transmitting*, etc. and m is a variable-length, priority ordered sequence of messages awaiting transmission on the channel. A *CANDLE* state vector, then, consists of a location as described above and a (set of) clock valuation(s), see figure 1.

3 Reachability Analysis

Of the several approaches to automated analysis of real-time systems, reachability analysis (RA) is one of the more easily implemented and informative. The basic reachability problem is to determine the set of system states which are encountered on any execution starting from some given state. RA allows the checking of *safety* properties of a system by answering the question: is it possible to reach an incorrect or unsafe state from an initial state. Other verification problems can be solved by building upon the solution of the basic RA problem [1]. Algorithm 1 outlines a basic algorithm for computing the set of states reachable from a given initial state.

Algorithm 1 (Reachable States)

VISITED $:= \{(q^{\mathcal{I}}, \mathbf{0})\}$
WAITING $:= \{(q^{\mathcal{I}}, \mathbf{0})\}$
while *WAITING* $\neq \emptyset$ **do**
  remove some (q, D) from *WAITING*
  $succ := \{(q_s, \mathrm{D}_s) : (q, \mathrm{D}) \xrightarrow{S} (q_s, \mathrm{D}_s) \wedge \mathrm{D}_s \neq \emptyset\}$
  foreach $(q_s, \mathrm{D}_s) \in succ$ **do**
    if $\forall (q_s, \mathrm{D}') \in$ *VISITED* $\bullet \mathrm{D}_s \not\subseteq \mathrm{D}'$
      add (q_s, D_s) to *VISITED*
      add (q_s, D_s) to *WAITING*
    fi
  od
od

The termination of the algorithm depends upon the construction of a finite quotient of the infinite transition system given by the TSA semantics. Important aspects of such a construction are 1) the use of *symbolic states* (q, D) where q is a location as usual and D is a clock constraint system which represents the set of clock valuations satisfying it, and 2) the definition of a transition relation $\xrightarrow{S}$ between symbolic states. Lack of space prevents us from giving further details here; the reader is referred to [15] for an excellent discussion of these issues.

The problem is to arrange for the compact storage of the set *VISITED* of visited state vectors, where a major difficulty has been to combine a good encoding of the locations (the discrete part of the system) with a good encoding for the clock valuations (the continuous part). This problem is addressed in the following section.

4 Sharing trees

The sharing tree data structure [16] has been designed for the compact storage of large sets of tuples. Their efficacy in the verification of untimed systems has been demonstrated; of particular interest to us is the work reported in [6, 7] which shows impressive space reductions for sets of state vectors

$\{p_1, p_2, \cdots, p_c\}$	$\langle v_1, v_2, \cdots, v_d \rangle$	$\langle (s_1, m_1), (s_2, m_2), \cdots, (s_n, m_n) \rangle$	$\langle \phi_1, \phi_2, \cdots, \phi_r \rangle$
Control	Data	Network	Clock Valuation
LOCATION – q			CLOCK VALUATION(S) – D

Figure 1: Structure of a *bCANDLE* state vector

in which each vector is very similar to the location component of a *CANDLE* state vector.

Definition 1 *A* sharing tree *is a rooted acyclic graph* $(N, V, val, r, succ)$ *where* $N = N_0 + N_1 \cdots + N_k$, *with* $k \geq 0$, *is a finite set of nodes which are organised into* $k+1$ *layers,* N_i *being the set of nodes of layer* $i, 0 \leq i \leq k$; V *is a set of values,* $\top, \bot \notin V$, *with valuation function* $val : N \to V \cup \{\top, \bot\}$; r *is the root node,* $N_0 = \{r\}$ *and* $\forall\, n \in N \bullet val(n) = \top \iff n \in N_0$; $succ : N \to 2^N$ *is the successor function which for a given node,* $n \in N_i$, *identifies the set of all nodes,* $succ(n) \subseteq N_{i+1}$, *which are directly descended from* n; *and the following properties hold: 1)* $\forall\, i \mid 0 \leq i < k, \forall\, n \in N \bullet succ(n) \subseteq N_{i+1}$: *each nodes has all of its successors in the next layer; 2)* $\forall\, n \in N, \forall\, s_1, s_2 \in succ(N) \bullet s_1 \neq s_2 \implies val(s_1) \neq val(s_2)$: *a node does not have distinct successors with equal values; 3)* $\forall\, i \mid 0 \leq i \leq k, \forall\, n_1, n_2 \in N_i \mid n_1 \neq n_2 \bullet val(n_1) = val(n_2) \implies succ(n_1) \neq succ(n_2)$: *if 2 or more nodes in the same layer have the same values then they have different sets of successors; 4)* $\forall\, n \in N \bullet val(n) = \bot \implies succ(n) = \emptyset$ *5)* $\forall\, n \in N \bullet succ(n) = \emptyset \implies (val(n) = \bot \vee val(n) = \top)$

The elements of a sharing tree are just those tuples of values which occur by following a path from the root node to a node whose value is $\bot$. For example, figure 2 shows a sharing tree representing the set of tuples: $\{(a, b, d), (a, c, d), (a, b, d, e, g), (a, b, d, f, g), (a, c, d, e, g), (a, c, d, f, g)\}$.

The data compression achieved by a sharing tree arises from the guaranteed sharing of all identical prefixes and a 'best-effort' sharing of identical suffixes. It is clear that a set of state vectors will usually contain many states that differ in few components and so allow for considerable sharing. The use of a sharing tree for the state store in a reachability analysis requires the partitioning of each state vector (figure 1) into a tuple of values, where each tuple component is allocated to a distinct layer in the tree. The mapping of components of the state vector into tuple components and the ordering of components within tuples can have a significant impact upon the space reductions achieved [7]. An important property of a sharing tree is that it can contain tuples of differing lengths. This allows considerable discretion in the mapping of variable length components of the state vector, for example the message

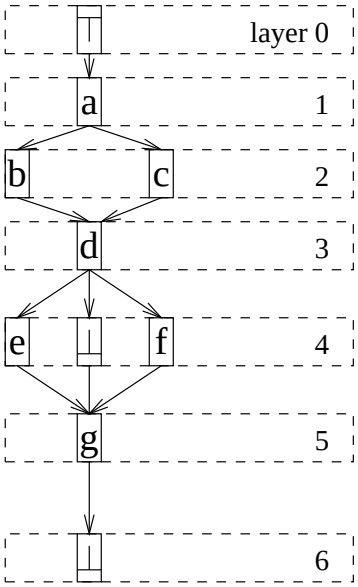

Figure 2: Sharing Tree Example

queues. The key observation concerning the suitability of sharing trees in the analysis of timed systems is that the clock constraint component D of a symbolic state (q, D) can be represented simply by one or more components of the complete state tuple and so can be easily incorporated into the state store. This contrasts significantly with approaches based upon a BDD representation of the state store where this integration is much more problematic.

5 Related work

Representation of timing constraints by DBMs was proposed by Dill [5] and has been preferred in the most efficient verification tools for timed systems, such as KRONOS [8] and UPPAAL [12].

Wong-Toi and Dill [14] and Balarin [3] have each shown techniques for encoding DBMs using BDDs and incorporating them into BDD encodings of the transition relation, approximating unions of zones using convex hulls. Bozga et. al. [4] offer a canonical representation of discretized sets of clock configurations using NDDs[1] [2], which are a BDD-based encoding amenable to combination with a symbolic representation of the discrete part of the system.

Larsen et. al. [11] propose a compact encoding

[1] Numerical Decision Diagrams

for DBMs which provides a minimal and canonical representation of clock constraints and allows for efficient inclusion checking between constraint systems. They do not consider how this representation may be combined with a symbolic representation of the rest of the system. Their approach is orthogonal to ours and it would be interesting to consider their combination experimentally.

Larsen et. al. [13] have recently proposed clock difference diagrams as a data structure for the compact representation of unions of zones. CDDs show some similarities both with BDDs and sharing trees. As far as we know, they remain to be used in practice.

6 Conclusions and further work

In this paper, we have proposed the use of sharing trees for the representation of the state space in the reachability analysis of timed systems. We believe that there is good reason to suppose that such a representation will offer significant space reduction, particularly in the analysis of asynchronous, data-bearing systems. The use of this representation comes with a time cost by comparison with the use of a traditional hash table but there is no reason to suppose that this penalty will be any greater in the analysis of timed systems than it is in the analysis of untimed systems where it has been shown to be acceptable in many circumstances [6, 7]. These conclusions remain to be confirmed by experiment.

References

[1] L. Aceto, A. Burgueño, and K. Larsen. Model checking via reachability testing for timed automata. Technical report, BRICS,Aarhus University, 1997.

[2] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. Data structures for the verification of timed automata. In O. Maler, editor, *Proc. 1st Int. Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*, pages 346–360. Springer Verlag, March 1997.

[3] F. Balarin. Approximate reachability analysis of timed automata. In *Proc. of 17th IEEE Real Time Systems Symposium*, pages 52–61. IEEE Computer Society Press, 1996.

[4] M. Bozga, O. Maler, A. Pnueli, and S. Yovine. Some progress in the symbolic verification of timed automata. volume 1254 of *Lecture Notes in Computer Science*, pages 179–190, Haifa, Israel, June 1997. Springer Verlag.

[5] D. Dill. Timing assumptions and verification of finite state concurrent systems. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer Verlag, 1989.

[6] F. Gagnon, J.-C. Grégoire, and D. Zampunieris. Sharing tree for "on-the-fly" verification. In *Proc. Int. Conf. on Formal Description Techniques VIII (FORTE'95)*. IEEE Computer Society Press, 1995.

[7] J.-Ch. Grégoire. State space compression in SPIN with ge-sets. In *Proc. 2nd SPIN Workshop, Rutgers University, New Jersey, USA*, August 1996.

[8] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[9] ISO/DIS 11898: Road Vehicles – interchange of digital information – Controller Area Network (CAN) for high speed communication, 1992.

[10] D. Kendall, S. Bradley, W.D. Henderson, and A.P. Robson. *bCANDLE*: Formal modelling and analysis of CAN control systems. In *Proceedings of 4th IEEE Real Time Technology and Applications Symposium (RTAS'98)*, pages 171–177. IEEE Computer Society Press, June 1998.

[11] K. Larsen, F. Larsson, P. Pettersson, and Wang Yi. Efficient verification of real-time systems: Compact data structure and state-space reduction. In *Proc. of 18th IEEE Real Time Systems Symposium*, December 1997.

[12] K. Larsen, P. Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Springer International Journal on Software Tools for Technology Transfer*, October 1997.

[13] K. Larsen, C. Weise, Wang Yi, and J. Pearson. Clock difference diagrams. Technical Report Nr 98/99, DoCs, Uppsala University, August 1998. ISSN 0283-0574.

[14] H. Wong-Toi and D. Dill. Approximations for verifying timing properties. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-time System Development*. World Scientific Publishing, 1994.

[15] S. Yovine. Model checking timed automata. In G. Rozenberg and F. Vaandrager, editors, *Embedded Systems, Papers from the European Educational Forum School on Embedded Systems, Veldhoven, The Netherlands*, Lecture Notes in Computer Science. Springer Verlag, 1997.

[16] D. Zampuniéris. *The Sharing Tree Data Structure*. PhD thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, May 1997.