VeriSiMPL: Verification via biSimulations of MPL Models*

Dieky Adzkiya[1] and Alessandro Abate[2]

[1] Delft Center for Systems and Control, TU Delft
[2] Department of Computer Science, University of Oxford

Abstract. VeriSiMPL ("very simple") is a software tool to obtain finite abstractions of Max-Plus-Linear (MPL) models. MPL models (Sect. 2), specified in MATLAB, are abstracted to Labeled Transition Systems (LTS). The LTS abstraction is formally put in relationship with the concrete MPL model via a (bi)simulation relation. The abstraction procedure (Sect. 3) runs in MATLAB and leverages sparse representations, fast manipulations based on vector calculus, and optimized data structures such as Difference-Bound Matrices. LTS abstractions can be exported to structures defined in the PROMELA. This enables the verification of MPL models against temporal specifications within the SPIN model checker (Sect. 4). The toolbox is available at

`http://sourceforge.net/projects/verisimpl/`

1 Motivations and Goals

Max-Plus-Linear (MPL) models are discrete-event systems [1] with continuous variables that express the timing of the underlying sequential events. MPL models are employed to describe the timing synchronization between interleaved processes, and as such are widely employed in the analysis and scheduling of infrastructure networks, such as communication and railway systems, production and manufacturing lines [1]. MPL models are classically analyzed by algebraic [1] or geometric techniques [2] over the max-plus algebra, which allows investigating properties such as transient and periodic regimes [1], or ultimate dynamical behavior. They can be simulated via the max-plus toolbox Scilab [3].

The recent work in [4,5] has explored a novel, alternative approach to analysis, which is based on finite-state abstractions of MPL models. The objective of this new approach is to allow a multitude of available tools that has been developed for finite-state models to be employed over MPL systems. We are in particular interested in the Linear Temporal Logic (LTL) model checking of MPL models via LTS abstractions.

This article presents VeriSiMPL, a software toolbox that implements and tests the abstraction technique in [4,5].

* This research is funded by the European Commission under the MoVeS project, FP7-ICT-2009-5 257005, by the European Commission under the NoE FP7-ICT-2009-5 257462, by the European Commission under Marie Curie grant MANTRAS PIRG-GA-2009-249295, and by NWO under VENI grant 016.103.020.

K. Joshi et al. (Eds.): QEST 2013, LNCS 8054, pp. 274–277, 2013.

2 Nuts and Bolts of Max-Plus-Linear Models

Define $\mathbb{R}_\varepsilon$ and ε respectively as $\mathbb{R} \cup \{\varepsilon\}$ and $-\infty$. For a pair $x, y \in \mathbb{R}_\varepsilon$, we define $x \oplus y = \max\{x, y\}$ and $x \otimes y = x + y$. Max-plus algebraic operations are extended to matrices as follows: if $A, B \in \mathbb{R}_\varepsilon^{m\times n}$ and $C \in \mathbb{R}_\varepsilon^{n\times p}$, then $[A \oplus B](i,j) = A(i,j) \oplus B(i,j)$ and $[A \otimes C](i,j) = \bigoplus_{k=1}^{n} A(i,k) \otimes C(k,j)$, for all i, j. An MPL model [1, Corollary 2.82] is defined as:

$$x(k) = A \otimes x(k-1) \oplus B \otimes u(k) \ ,$$

where $A \in \mathbb{R}_\varepsilon^{n\times n}$, $B \in \mathbb{R}_\varepsilon^{n\times m}$, $x(k) \in \mathbb{R}_\varepsilon^{n}$, $u(k) \in \mathbb{R}_\varepsilon^{m}$, for $k \in \mathbb{N}$. In this work, the state and input spaces are taken to be $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively: the independent variable k denotes an increasing discrete-event counter, whereas the n-dimensional state variable x defines the (continuous) timing of the discrete events and the m-dimensional input u characterizes external schedules. If the input matrix B contains at least a finite (not equal to ε) element, the MPL model is called *nonautonomous*, otherwise it is called *autonomous* since it evolves under no external schedule. Nonautonomous models embed nondeterminism in the form of a controller input.

Implementation: VeriSiMPL accepts MPL models written in MATLAB. For practical reasons, the state matrix A is assumed to be row-finite, namely characterized in each row with at least one element different from ε.

Example: Consider the following autonomous MPL model from [1, p. 4], representing the scheduling of train departures from two connected stations $i = 1, 2$ (event k denotes the time of the k-th departure at time $x_i(k)$ for station i)

$$x(k) = \begin{bmatrix} 3 & 7 \\ 2 & 4 \end{bmatrix} \otimes x(k-1), \text{ i.e. } \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \begin{bmatrix} \max\{3 + x_1(k-1), 7 + x_2(k-1)\} \\ \max\{2 + x_1(k-1), 4 + x_2(k-1)\} \end{bmatrix} \ .$$

3 From MPL Models to Labeled Transition Systems

We seek to construct a finite-state Labeled Transition System (LTS) as an abstraction of an (autonomous or nonautonomous) MPL model. An LTS comprises a set of finitely many states (Sect. 3.1), of a set of transitions relating pairs of states (Sect. 3.2), and is further decorated with labels on either states or transitions (Sect. 3.3).

3.1 LTS States: Partitioning of MPL Space

LTS states are obtained by partitioning the state space $\mathbb{R}^n$ based on the underlying dynamics, that is based on the state matrix A [4, Algorithms 1,2]. The partition can be further refined (in order to seek a bisimulation of the concrete model) or otherwise coarsened by merging adjacent regions (in order to reduce the cardinality of the set of abstract states).

Implementation: VeriSiMPL implements two alternative approaches [4, Algorithms 1,2]. In order to improve the performance of the procedure, standard pruning tricks are applied. Each generated region is shown to be a Difference-Bound Matrix (DBM) [6, Sect. 4.1]: this allows a computationally efficient representation based on the expression $x_i - x_j \bowtie \alpha_{i,j}$, $\bowtie \in \{<, \leq\}$. VeriSiMPL represents a DBM as a row cell with two elements: the first element is a real-valued matrix representing the upper bound $\alpha_{i,j}$, whereas the second is a Boolean matrix representing the value of $\bowtie$. A collection of DBM is also represented as a row with two elements, where the corresponding matrices are stacked along the third dimension. Quite importantly, DBM are closed under MPL operations.

Example: The partitioning regions generated for the MPL model in Sect. 2 are $R_1 = \{x \in \mathbb{R}^2 : x_1 - x_2 > 4\}$, $R_2 = \{x \in \mathbb{R}^2 : 2 < x_1 - x_2 \leq 4\}$, and $R_3 = \{x \in \mathbb{R}^2 : x_1 - x_2 \leq 2\}$.

3.2 LTS Transitions: Forward-Reachability Analysis

An LTS transition between any two abstract states R and R' is generated based on the relation between the two corresponding partitioning regions. At any given event counter k, there is a transition from R to R' if there exists an $x(k-1) \in R$ and possibly a $u(k) \in U \subseteq \mathbb{R}^m$ such that $x(k) \in R'$. Such a transition can be determined by a forward-reachability computation, i.e. checking the non-emptiness of $R' \cap \{x(k) : x(k-1) \in R,\ u(k) \in U\}$. We assume that the set of allowed inputs $U \subseteq \mathbb{R}^m$ is characterized via a DBM.

Implementation: VeriSiMPL performs forward reachability by mapping and manipulating DBM. It represents a transition in MATLAB as a sparse Boolean matrix. As in a precedence graph [1, Definition 2.8], the (i, j)-th element equals to 1 if there is a transition from j to i, else it is equal to 0.

Example: The transitions for the model in Sect. 2 are represented in Fig. 1. In a nonautonomous version of the model, the finite-state structure in Fig. 1 will simply present additional transitions.

3.3 LTS Labels: Fast Manipulation of DBM

LTS labels are quantities associated with states or transitions and characterize

1) the difference between the timing of a single event (k) for any two variables of the original MPL model, i.e. $x_i(k) - x_j(k)$, where $1 \leq i < j \leq n$; or

2) the time difference between consecutive events of the MPL model, i.e. $x_i(k) - x_i(k-1)$, for $1 \leq i \leq n$.

The first class of labels is determined by the representation of a partitioning region, whereas the second is derived from an outgoing partitioning region and its affine dynamics.

Implementation: Practically, in both cases VeriSiMPL stores the labels as (unions of) vectors of real-valued intervals in MATLAB. In the second case the labels are computed by fast DBM manipulations.

Example: The obtained LTS can be expressed as a simple text file and parsed by Graphviz for plotting, as displayed in Fig. 1.

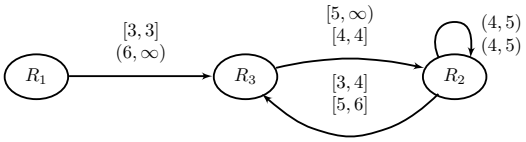

Fig. 1. LTS abstraction of the MPL model in Sect. 2, inclusive of abstract states, transitions, and labels

4 Computational Benchmark and Case Study

We have computed the runtime required to abstract an autonomous MPL system as a finite-state LTS, for increasing dimensions n of the MPL model, and kept track of the number of states and of transitions of the obtained LTS (memory requirement). Compared to partition-based abstraction procedures in the literature for other classes of dynamical systems [7], the present procedure comfortably manages MPL models with significant size (number of continuous variables).

Implementation: For any n, we have generated row-finite matrices A with 2 finite elements (random integers taking values between 1 and 100) placed randomly in each row. The algorithms have been implemented in MATLAB 7.13 (R2011b) and the experiments have been run on a 12-core Intel Xeon 3.47 GHz PC with 24 GB of memory. For $n = 15$, VeriSiMPL generates an LTS with about 10^4 states and 10^6 transitions, with a runtime limited within a few hours.

Example: The obtained LTS can be exported to PROMELA (a PROcess MEta LAnguage), to be later used by the SPIN model checker [8]. Consider the specification $\Psi : \forall k \in \mathbb{N}, \psi(k)$, where $\psi(k) = \{x_2(k+1) - x_2(k) \leq 6\}$. Notice that Ψ can be expressed as $\Box\psi$. We obtain the satisfiability set $\mathrm{Sat}(\Psi) = \{R_2, R_3\}$.

References

1. Baccelli, F., Cohen, G., Olsder, G., Quadrat, J.P.: Synchronization and Linearity, An Algebra for Discrete Event Systems. John Wiley and Sons (1992)
2. Katz, R.: Max-plus (A,B)-invariant spaces and control of timed discrete-event systems. IEEE Trans. Autom. Control 52(2), 229–241 (2007)
3. Plus, M.: Max-plus toolbox of Scilab (Online) (1998), `http://www.cmap.polytechnique.fr/~gaubert/MaxplusToolbox.html`
4. Adzkiya, D., De Schutter, B., Abate, A.: Abstraction and verification of autonomous max-plus-linear systems. In: Proc. 31st Amer. Control Conf., pp. 721–726 (2012)
5. Adzkiya, D., De Schutter, B., Abate, A.: Finite abstractions of nonautonomous max-plus-linear systems. In: Proc. 32nd Amer. Control Conf. (June 2013)
6. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 197–212. Springer, Heidelberg (1990)
7. Yordanov, B., Belta, C.: Formal analysis of discrete-time piecewise affine systems. IEEE Trans. Autom. Control 55(12), 2834–2840 (2010)
8. Holzmann, G.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley (2003)