Finite Abstractions of Max-Plus-Linear Systems

Theory and Algorithms

Dieky Adzkiya

Cover illustration: The graphical representation of a transition system and a partition is depicted in the foreground and background, respectively.
Cover design: Yazdi Ibrahim Jenie

Finite Abstractions of Max-Plus-Linear Systems

Theory and Algorithms

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op donderdag 9 oktober 2014 om 10:00 uur
door Dieky ADZKIYA,
Master of Science in Mathematics,
Sepuluh Nopember Institute of Technology,
geboren te Lamongan, Indonesië.

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr. A. Abate
Prof.dr.ir. B. De Schutter

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr. A. Abate	University of Oxford, promotor
Prof.dr.ir. B. De Schutter	Technische Universiteit Delft, promotor
Prof.dr.ir. J.H. van Schuppen	Technische Universiteit Delft
Prof.dr. C. Witteveen	Technische Universiteit Delft
Prof.dr. B. Heidergott	Vrije Universiteit Amsterdam
Prof. Dr.-Ing. J. Raisch	Technische Universität Berlin
Prof.dr. S. Sankaranarayanan	University of Colorado Boulder

The research described in this thesis was supported in full by the European Commission Marie Curie grant MANTRAS 249295, and in part by the European Commission STREP project MoVeS 257005 and by the European Commission IAPP project AMBI 324432, and by the NWO VENI grant 016.103.020.

Published and distributed by: Dieky Adzkiya
E-mail: d.adzkiya@tudelft.nl

ISBN 978-94-6203-667-3

Keywords: max-plus-linear systems, switching min-plus-linear systems, network calculus, stochastic max-plus-linear systems, finite abstractions, reachability, linear temporal logic, probabilistic invariance specifications.

Copyright © 2014 by Dieky Adzkiya

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission of the author.

Printed by CPI-Wöhrmann Print Service – Zutphen

Preface

All praise is due to Almighty God, the most gracious and the most merciful. Despite my many shortcomings, He always bestowed His blessings upon me. I hope and pray that His mercy continues upon me and my family unabatedly and that we remain only His servant till the end of our life. Verily, He is the source of all strength.

I owe a lot of thanks to Alessandro and Bart for giving me the opportunity to come to the Netherlands to pursue my PhD. Thanks to Alessandro for teaching me how to do research and for the freedom he gave me to do research. Also I wanted to thank Bart for the valuable comments as well as useful discussions we had during the last four years.

I would like to thank to all people in the Honeywell Prague Laboratory for being kind and friendly to me. Ondřej, thanks for organizing the visit and I missed the post-lunch shopping. Karel, thanks for your patience during our very long and frequent discussions. Ludek, thanks for the training in HVAC system. Jan, thanks for explaining your paper in a great detail. Vít, thanks for mentioning your father's proposition about the comparison between the Czech and the Hawaiian language. Eva, thanks for being a nice officemate. Petr, I believe that hostel is one of your favourite movies. Katerina, thanks for handling the administration. Karel and Zdenek, thanks for attending my presentation. Petr and Jiri, I enjoyed our discussions in the kitchen. Linh and Matthias, both of you are tough people because you can handle many projects simultaneously.

It has been a delight for me to work with Bassilio and Yining during their master project. Coincidentally they have a similar comment: it is hard to relate the theoretical results in the paper and the implementation. Thanks for the constructive feedback: in this thesis, I have tried to clarify the relationship between those two aspects.

During my PhD study, I went to two excellent summer schools. The first summer school was on control of discrete-event systems: automata and Petri nets perspectives. It was organized in the framework of the DISC (Distributed Supervisory Control of Large Plants) european project. I wanted to thank all presenters for the nice presentations and in particular to Serge for introducing me to difference-bound matrices. The second summer school was on specification, design and verification of distributed control systems. It was organized by the European Embedded Control Institute. Richard, Ufuk, and Nok, thanks for giving me a better understanding of temporal logic, model checking and some other interesting stuff.

I have greatly appreciated all my committee members for taking time and for participating in my defense ceremony: Prof. Jan van Schuppen, Prof. Cees Witteveen, Prof. Bernd Heidergott, Prof. Jörg Raisch, and Prof. Sriram Sankaranarayanan. Their constructive comments and suggestions helped me to improve my dissertation in the final stage.

Without the supporting staff at DCSC, life would have been very hard. Thanks to Kitty, Esther, Heleen, Saskia, Marieke for handling all paperworks. Olaf, thanks for handling the financial matters. Will, I still remember your remark that vi is the best editor.

Sadegh, Ilya, Katerina, and Majid: I missed the group meeting. Rudy, thanks for the well commented and highly reliable thesis template. Noortje, thanks for translating the summary and propositions in Dutch. Pawel and Aleksandar, thanks for being nice officemates. Amir, Marco, Mohammad, Renshi, Esmaeil, Patricio, Hans, Yashar, Pieter, and Laurens: I enjoyed playing table football with you. Bart and Jia, I am glad to know that I am not the only one working in max plus. Hildo, Ana, and Sachin: knowledge based control systems are a nice course to assist. Arne, we start our PhD study almost at the same time and our children were born almost at the same time. Jacopo, thanks for your hints in using MEX C. Edwin, it is nice to talk to you about tennis. Mehdi, I know that you are a family man. Hans, we took the english for academic purposes 2 course. Yue, we participated on the GPU programming course. Yihui, thanks for the chinese plug. Max, Kim, and Anna: you are great representatives. Manuel, thanks for offering me a postdoc position. Gabriel, thanks for answering my questions on mathematica. Robert, I enjoyed assisting your course, i.e. knowledge based control systems. Michel, thanks for encouraging me to publish at least three journal papers during my PhD study. Hans, thanks for motivating me to do better in both the scientific and social aspects. Elisabeth, Zhe, Jan-Maarten, Laura, Anqi, Yasin, Juan, Mohammad, Yu, Le, Shuai, Ruxandra, Subramanya, Zhou, Skander, Zhao, Tope, Visa, Shukai, Cornelis, Yiming, Jun, Chengpu, Simone, Xavier, Tamas, Xander, Ton, Jan-Willem for being nice to me and for saying “hi” when we meet each other.

Thanks to the Indonesian community in the Netherlands for creating such a wonderful experience in these years. Jusuf, thanks for the bike. Anisah, Aulia, Orchidea, Lasmini, Wieke, thanks for helping us to take care our children. Yazdi, thanks for designing the cover and the invitation. Asyrof, thanks for the baby bike seat. Agung, Zulkifli, Muhammad, thanks for the ton of useful information about living in the Netherlands. Rini, Deden, Deeyah, Marwan, Ririn, thanks for inviting us to your party. Enik and Tri, thanks for helping us with the first delivery. Hamdi, thanks for teaching me Arabic language. Finally thanks to all members of PPI Delft such as Adhi, Agus, Andri, Budi, Desi, Didin, Dira, Duddy, Dwi, Ida, Isnaeni, Junaedi, Pandu, Pita, Pungky, Rahmi, Reni, Saputra, Senot, Yudha, Yugi for organizing many innovative scientific and social activities.

I would like to thank all colleagues in the Mathematics department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia for encouraging me to take a PhD abroad. I have greatly appreciated the help from Basuki and Erna as the former and current head of the department, respectively. Subchan and Subiono, thanks for the excellent letter of recommendations. Oki, thanks for the amazing accomodation.

Words can never be enough to express my sincerest thank to my family. My parents and my parents in law, your constant prayers have always been the best help that I will ever need. My warmest thank goes to Mia, Thuba and Izah. Mia, thanks for being here with me and accompanying me in every step. Thanks for always being supportive, for sharing every moment of happiness and sadness, and for keeping up with me. Thuba and Izah, thanks for your presence in our life. Finally, I would like to thank all people that helped me and my family directly or indirectly during these years. I hope that all of you have a fruitful life. I wish you all the best.

Dieky Adzkiya,
Delft, August 2014.

Contents

Preface v

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Research Goals and Original Contributions 3
- 1.3 Overview of the Thesis 4
- 1.4 Publications by the Author 5

2 Models **7**
- 2.1 Max-Plus-Linear Systems 7
- 2.2 Related Models 11
  - 2.2.1 Min-Plus-Linear Systems 11
  - 2.2.2 Switching Min-Plus-Linear Systems 12
  - 2.2.3 Stochastic Max-Plus-Linear Systems 13
  - 2.2.4 Piecewise-Affine Systems 15
  - 2.2.5 Piecewise Switched Affine Systems 18
- 2.3 Summary 19

3 Finite Abstractions of Max-Plus-Linear Systems **21**
- 3.1 Related Work 21
- 3.2 Preliminaries 22
  - 3.2.1 Difference-Bound Matrices 22
  - 3.2.2 Transition Systems 29
  - 3.2.3 Linear Temporal Logic 30
  - 3.2.4 Abstractions 31
- 3.3 Autonomous Max-Plus-Linear Systems 34
  - 3.3.1 States: Partitioning Procedure 35
  - 3.3.2 Transitions: One-Step Reachability 39
  - 3.3.3 Bisimulation-Quotienting Procedure 41
- 3.4 Nonautonomous Max-Plus-Linear Systems 44
  - 3.4.1 States: Partitioning Procedure 45
  - 3.4.2 Transitions: One-Step Reachability 46
  - 3.4.3 Bisimulation-Quotienting Procedure 47
- 3.5 Implementation: VeriSiMPL 49
- 3.6 Computational Benchmark 49
- 3.7 Summary 50

4 Reachability Analysis of Max-Plus-Linear Systems 53
4.1 Related Work 53
4.2 Forward Reachability Analysis 54
4.2.1 Sequential Computation of the Reach Tube 55
4.2.2 One-Shot Computation of the Reach Set 58
4.3 Backward Reachability Analysis 60
4.3.1 Sequential Computation of the Backward Reach Tube 60
4.3.2 One-Shot Computation of the Backward Reach Set 63
4.4 Applications 64
4.4.1 Safety Analysis 64
4.4.2 Transient Analysis 64
4.5 Numerical Benchmark 65
4.5.1 Implementation and Setup of the Benchmark 65
4.5.2 Comparison with an Alternative Computation 66
4.6 Summary 68

5 Verification of Properties for Network Calculus Elements via Finite Abstractions 69
5.1 Related Work 69
5.2 Network Calculus 70
5.3 Min-Plus State-Space Formulation 71
5.4 Abstraction of Autonomous Switching Min-Plus-Linear Systems 75
5.4.1 States: Partitioning Procedure 75
5.4.2 Transitions: One-Step Reachability 77
5.5 Formal Verification of Switching Min-Plus-Linear Systems 79
5.6 Summary 79

6 Finite Abstractions of Stochastic Max-Plus-Linear Systems 81
6.1 Related Work 81
6.2 The Probabilistic Invariance Problem 82
6.3 Abstraction by a Finite State Markov Chain 84
6.4 Quantification of the Abstraction Error 85
6.4.1 Lipschitz Continuous Density Functions 86
6.4.2 Piecewise Lipschitz Continuous Density Functions 89
6.5 Summary 92

7 Conclusions and Future Research 95
7.1 Conclusions 95
7.2 Recommendations for Future Research 96

Bibliography 99

Glossary 109

Samenvatting 113

Summary 117

Curriculum Vitae **119**

Chapter 1

Introduction

This thesis discusses the formal verification of Max-Plus-Linear (MPL) systems. In this chapter we introduce MPL systems, verification problems, and an application in communication networks. We further briefly sketch our approach to solve these problems, which will be further elaborated throughout the thesis. The explanation of the organization of the thesis concludes this chapter.

1.1 Motivation

The seminal work in [22, p. ix] characterizes discrete-event dynamic systems as follows: discrete-event dynamic systems encompass man-made systems consisting of a finite number of resources (processors or memories, communication channels, machines) shared by several users (jobs, packets, manufactured objects), which contribute to the achievement of a common goal (a parallel computation, the end-to-end transmission of a set of packets, the assembly of a product in an automated manufacturing line). The dynamics of such systems has to deal with synchronization and with concurrency. Synchronization requires the availability of several resources or users at the same time, whereas concurrency appears when some user must choose among several resources at a particular time instant.

Max-Plus-Linear (MPL) systems are a class of discrete-event dynamic systems [22, 40] with a continuous state space characterizing the timing of the underlying sequential discrete events. MPL systems are predisposed to describe the evolution of timed event graphs in the event domain, under the assumption that timing events are linearly dependent (within the max-plus algebra) on previous event occurrences and (for nonautonomous systems) on exogenous schedules. MPL systems have a wide range of applications: they have been employed in the analysis and scheduling of infrastructure networks, such as communication and railway systems [68], production and manufacturing lines [106, 112], or biological systems [28].

Timed event graphs are a class of timed Petri nets where each place has a single upstream transition and a single downstream transition [22, Sec. 2.5]. These systems describe synchronization without concurrency. The dynamics can be represented either as a dater or as a counter. The dater description uses the max-plus algebra and is called an event-domain description, i.e. the independent variable denotes an event index and the state variable denotes the time of event occurrences. In the max-plus algebra, the “addition” is defined as the maximization and the “multiplication” is defined as the usual addition. On the other hand, the counter description uses the min-plus algebra and is called time-domain description, i.e. the independent variable denotes time and the state variable is a counter of events occurred up to a certain time. As suggested by the name, in min-plus algebra, the “addition” and “multiplication” are defined as the minimization and as the usual addition, respectively. The interested reader is referred to [22, Sec. 5.2] for more details.

Over the past three decades, many fundamental problems for MPL systems have been studied [38, 45, 46, 50, 64, 74, 90, 104]. Classical dynamical analysis of MPL systems is grounded on their algebraic features [22]. It allows investigating global system properties such as its transient or asymptotic behaviors, its periodic regimes, or its ultimate dynamical behavior [45]. Those system properties can be studied by using the spectral theory of system matrices in the max-plus algebra. Recently some results have appeared on the geometric theory for MPL systems introduced in [38], such as the computation of different control on invariant sets [50, 74] and the feedback controller design [90]. The application of model predictive control in MPL systems has been studied in [46] and the subsequent line of work.

In this thesis we develop an alternative approach to analysis of MPL systems based on finite-state abstractions. More precisely we consider the following verification problem. Given an MPL system and a specification, we determine whether the MPL system satisfies the specification. Solution of the verification problem w.r.t. a class of specifications can be determined by reachability computations. This motivates us to study reachability of MPL systems. However specifications can express richer properties and be characterized as formulae in certain temporal logics or as automata.

Reachability analysis is a fundamental problem in the area of formal methods, systems theory, and performance and dependability analysis. It is concerned with assessing whether a certain state of a system is attainable from given initial states of the system. The problem is particularly interesting and compelling over models with continuous components – either in time or in (state) space [16, 18, 25, 32–34, 44, 63, 69, 76, 79, 80, 86, 94–96]. With regards to MPL systems, reachability analysis from a *single* initial condition has been investigated in [38, 58, 61] by leveraging the computation of the reachability matrix, which leads to a parallel with reachability for discrete-time linear dynamical systems. Furthermore, the existing literature does not deal with backward reachability analysis. Under the requirement that the set of initial conditions is expressed as a max-plus polyhedron [60, 120], forward reachability analysis can be performed over the max-plus algebra. Similar results hold for backward reachability analysis of autonomous MPL systems, where in addition the system matrix has to be max-plus invertible.[1] To the best of the author’s knowledge, there are no direct approaches for solving the backward reachability problem of nonautonomous MPL systems in the max-plus algebra. In this thesis we extend the forward and backward reachability computations of MPL systems by considering an arbitrary set of initial and final conditions, respectively. Furthermore in both cases, the system matrices do not have to be max-plus invertible. We leverage the data structure of Difference-Bound Matrices (DBM) [51] that is easy to manipulate computationally. A DBM is the intersection of finitely many half-space representations that are characterized by the difference of two variables.

In order to showcase the effectiveness of the developed theory, we apply our abstrac-

---

[1] A matrix is max-plus invertible iff there is a single finite element (not equal to $-\infty$) in each row and in each column.

tion techniques for MPL systems to verify some properties of communication systems. The communication systems of interest are modeled using network calculus. Network calculus makes use of the min-plus algebra to provide strong performance guarantees for deterministic communication systems [85]. The main quantities of interest are backlog and virtual delay. The backlog is the amount of data that is held inside the system. The virtual delay at time t is the amount of time spent inside the system by the data that has entered at time t, if the data is served after all the data that has entered before time t has been served. The main network calculus results deal with bounds on the backlog [85, Th. 1.4.1] and on the virtual delay [85, Th. 1.4.2]. Both of these quantities are highly relevant in control implementations: the first one is necessary to guarantee the absence of packet drop-outs, whereas the latter can be typically assumed to provide a bound for the delay in the feedback measurements. In this thesis we apply the abstraction techniques to the switching Min-Plus-Linear (MiPL) representation of network calculus. A switching MiPL system is a system that can switch between different modes of operation, where the dynamics in each mode are described by MiPL equations.

Stochastic Max-Plus-Linear (SMPL) systems [67, 100, 105] are MPL systems where the delays between successive events (in the examples above, the processing or transportation times) are now characterized by random quantities. In practical applications SMPL systems are more realistic than simple MPL ones: for instance in a model for a railway network, train running times depend on driver behavior, on weather conditions, and on passenger numbers at stations. As such they are arguably more suitably modeled by random variables than deterministic ones. Only a few approaches have been developed in the literature to study the steady-state behavior of SMPL systems, for example employing Lyapunov exponents and asymptotic growth rates [20–22, 57, 62, 92, 111]. The Lyapunov exponent of SMPL systems under some assumptions has been studied in [111], and later these results have been extended to approximate computations under other technical assumptions in [62, p. 251]. The application of model predictive control and system identification to SMPL systems, under given structural assumptions, has been studied in [54, 55]. In this thesis we investigate the use of finite abstractions to study the finite-horizon probabilistic invariance problem over SMPL systems. The probabilistic invariance problem amounts to determining the probability of satisfying the invariance property for each allowable initial condition. We tailor the techniques in [3, 52] to determine the approximate solution of the problem.

1.2 Research Goals and Original Contributions

The broad aim of this PhD research is to develop a novel and general framework for the formal verification of MPL systems and SMPL systems. In the process, we obtain results in reachability of MPL systems and apply the abstraction techniques to the investigation of existing network calculus elements.

Formal verification of MPL systems via finite abstractions. We propose an analysis method based on finite-state abstractions of autonomous and nonautonomous MPL systems. We seek to synthesize techniques that are computationally agile by employing a novel representation of the quantities into play (regions over state and control spaces, as well as model dynamics). By expressing general dynamical properties as specifications in a modal logic such as Linear Temporal Logic (LTL), the abstraction allows for the formal verification of classes of properties by means of model checking.

Reachability computations of MPL systems. We extend the results in the literature for forward reachability analysis by considering an arbitrary set of initial conditions. Additionally for backward reachability analysis, we are able to handle nonautonomous MPL systems and state matrices that are not max-plus invertible. We illustrate the application of reachability computations over safety and transient analysis of MPL systems.

Implementations. Most abstraction and reachability algorithms have been implemented as a MATLAB software tool, "Verification via biSimulations of MPL models" (VeriSiMPL, as in "very simple"), which is freely available for download at http://www.sourceforge.net/projects/verisimpl.

Automatic verification of network properties. We focus on the automatic synthesis of bounds on the virtual delay and on the backlog of a communication network. Although such properties can already be analyzed using network calculus tools, the virtue of our approach lies in its completely automated nature, and in opening the door to the automatic verification of certain communication topologies, e.g. flow aggregates, which network calculus cannot easily cope with. Furthermore, the use of abstraction approaches similar to those proposed for the automatic synthesis of control software, enables the simultaneous verification of control and communication software over more complex properties than those discussed in this thesis.

Finite abstractions of SMPL systems. We investigate the use of finite abstractions to study the finite-horizon probabilistic invariance problem over SMPL systems. The techniques are inspired by [3, 52, 100, 105]. The invariant property characterizes the desired delay of event occurrences w.r.t. a given schedule.

1.3 Overview of the Thesis

This thesis discusses approaches to analysis that are based on finite-state abstractions of MPL systems, switching MiPL systems, and SMPL systems. Additionally for MPL systems, we also discuss an approach based on reachability analysis. This thesis is organized as follows:

- **Chapter 2** introduces the definition of MPL systems and recalls a few of its basic properties. A number of related models that are going to be used throughout the thesis are then briefly discussed: Min-Plus-Linear (MiPL) systems, Switching MiPL systems, Stochastic MPL (SMPL) systems, Piece-wise Affine (PWA) systems, and Piecewise Switched Affine (PWSA) systems.

- In **Chapter 3** the abstraction procedure of autonomous and nonautonomous MPL systems is discussed. First of all, some preliminary concepts are introduced such as Difference-Bound Matrices (DBM), transition systems, Linear Temporal Logic (LTL), and abstractions. The abstraction procedure consists of a partitioning of the state space and of determining possible transitions between pairs of partition sets. A partition-refinement procedure is additionally proposed in order to increase the abstraction precision. The abstraction algorithms are implemented in the VeriSiMPL tool.

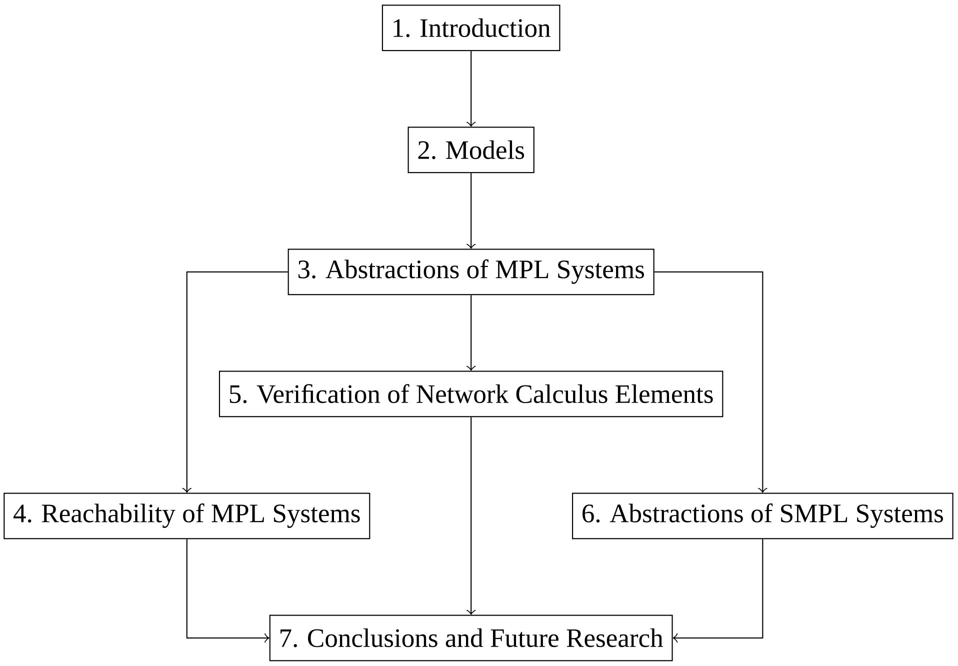
1. Introduction
2. Models
3. Abstractions of MPL Systems
5. Verification of Network Calculus Elements
4. Reachability of MPL Systems
6. Abstractions of SMPL Systems
7. Conclusions and Future Research

Fig. 1.1: Relational structure of this thesis. Arrows indicate relations of inter-dependence.

- Computational techniques for reachability analysis of MPL systems are discussed in **Chapter 4**. This chapter covers autonomous and nonautonomous MPL systems, sequential and "one-shot" computations of reach tubes and sets respectively, as well as forward and backward reachability analysis. The reachability algorithms are implemented as a part of VeriSiMPL.
- In **Chapter 5** we discuss the verification of specific properties of network calculus elements. The approach is again based on finite-state abstractions and is an extension and an application of the techniques elaborated in Chapter 3. The properties of interest for this study are backlog and virtual delay, and extensions are discussed.
- **Chapter 6** studies the finite-horizon probabilistic invariance problem over Stochastic MPL (SMPL) systems. First SMPL systems are formulated as a discrete-time Markov processes. Then the formal abstraction techniques of [3, 52] are tailored to SMPL systems.
- **Chapter 7** summarizes the results of this thesis and outlines directions for future research.

1.4 Publications by the Author

Most of the material presented in Chapters 3-6 of this PhD thesis has appeared in international conference proceedings, both in the area of systems & control and in that of formal verification, or has been published in peer-reviewed journals. In addition to developing the theory, we have implemented most algorithms in this thesis as a MATLAB toolbox VeriSiMPL. The connection between each chapter and the publications is as follows

- Chapter 3 is based on [5–8]
- Chapter 4 is based on [9–11]
- Chapter 5 is based on [43] and [42] where the author is one of the supervisors in the latter reference
- Chapter 6 is based on [12]

Chapter 2

Models

In this chapter we present a brief overview of Max-Plus-Linear (MPL) systems, followed by a concise description of some related models, such as Min-Plus-Linear (MiPL) systems, switching MiPL systems, stochastic MPL systems, Piecewise-Affine (PWA) systems, Piecewise Switched Affine (PWSA) systems.

2.1 Max-Plus-Linear Systems

In this section we introduce the syntax and semantics in the max-plus algebra, followed by a discussion on Max-Plus-Linear (MPL) systems and its properties [22]. Define $\mathbb{R}$, $\mathbb{R}_\varepsilon$, and ε respectively as the set of real numbers, $\mathbb{R}\cup\{\varepsilon\}$, and $-\infty$. For $\alpha,\beta\in\mathbb{R}_\varepsilon$, introduce the two operations

$$\alpha\oplus\beta=\max\{\alpha,\beta\} \qquad \text{and} \qquad \alpha\otimes\beta=\alpha+\beta,$$

where the element ε is considered to be absorbing w.r.t. $\otimes$ [22, Def. 3.4]. Given $\beta\in\mathbb{R}$, the max-algebraic power of $\alpha\in\mathbb{R}$ is denoted by $\alpha^{\otimes\beta}$ and corresponds to $\alpha\beta$ in the conventional algebra. In this thesis the usual multiplication symbol $\times$ is usually omitted, whereas the max-algebraic multiplication symbol $\otimes$ is written. The rules for the order of evaluation of the max-algebraic operators correspond to those of conventional algebra: max-algebraic power has the highest priority, and max-algebraic multiplication has a higher precedence than max-algebraic addition [22, Sec. 3.1]. The basic max-algebraic operations are extended to matrices as follows. If $A,B\in\mathbb{R}_\varepsilon^{m\times n}$; $C\in\mathbb{R}_\varepsilon^{m\times p}$; $D\in\mathbb{R}_\varepsilon^{p\times n}$; and $\alpha\in\mathbb{R}_\varepsilon$,

$$\begin{aligned}
[\alpha\oplus A](i,j)&=\alpha\oplus A(i,j),\\
[A\oplus B](i,j)&=A(i,j)\oplus B(i,j),\\
[C\otimes D](i,j)&=\bigoplus_{k=1}^{p}C(i,k)\otimes D(k,j),
\end{aligned}$$

for all $i=1,\ldots,m$ and $j=1,\ldots,n$. The notation $A(i,j)$ represents the entry of matrix A at i-th row and j-th column. Notice the analogy between $\oplus$, $\otimes$ and $+$, $\times$ for matrix and vector operations in the conventional algebra. Given $m\in\mathbb{N}$, the m-th max-algebraic power of $A\in\mathbb{R}_\varepsilon^{n\times n}$ is denoted by $A^{\otimes m}$ and corresponds to $A\otimes\cdots\otimes A$ (m times). Notice that $A^{\otimes 0}$

is an n-dimensional max-plus identity matrix, i.e. the diagonal and nondiagonal elements are 0 and ε, respectively. In this thesis, the following notation is adopted for reasons of convenience: a vector with each component that equals to 0 (resp. $-\infty$) is also denoted by 0 (resp. ε). Furthermore the state space is taken to be $\mathbb{R}^n$ (rather than $\mathbb{R}^n_\varepsilon$), which also implies that the state matrix A has to be row-finite (cf. Definition 2.2).

An autonomous Max-Plus-Linear (MPL) system [22, Rem. 2.75] is defined as:

$$\mathbf{x}(k) = A \otimes \mathbf{x}(k-1), \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}_\varepsilon$, $\mathbf{x}(k-1) = [x_1(k-1)\dots x_n(k-1)]^T \in \mathbb{R}^n$ for $k \in \mathbb{N}$. We use the bold typeset for vectors and tuples, whereas the entries are denoted by the normal typeset with the same name and index. The independent variable k denotes an increasing event index, whereas the state variable $\mathbf{x}(k)$ defines the (continuous) time of occurrence of the k-th event. Autonomous MPL systems are characterized by deterministic dynamics, namely they are unaffected by exogenous inputs in the form of control signals or of environmental non-determinism.

Many classical concepts of system theory are exportable to MPL systems such as state-space recursive equations, input-output (transfer) functions, feedback loops, eigenvalue, eigenvector etc. In this thesis, we focus on max-plus eigenvalue and eigenvectors. As it will be clear later, the existence of max-plus eigenvalue and eigenvectors depends on irreducibility of the state matrix. The notion of irreducibility can be defined according to the precedence (or communication) graph of the state matrix.

Definition 2.1 (Precedence Graph [22, Def. 2.8]) The precedence graph of $A \in \mathbb{R}^{n\times n}_\varepsilon$, denoted by $\mathcal{G}(A)$, is a weighted directed graph with vertices $1,\dots,n$ and an arc (j,i) with weight $A(i,j)$ for each $A(i,j) \neq \varepsilon$. □

Definition 2.2 (Regular (Row-Finite) Matrix [68, Sec. 1.2]) A matrix $A \in \mathbb{R}^{n\times n}_\varepsilon$ is called regular (or row-finite) if A contains at least one element different from ε in each row. □

Example Consider the following two-dimensional MPL system that models a simple railway network between two cities [68, Sec. 0.1] ($x_i(k)$ is the time of the k-th departure at station i for $i = 1,2$):

$$\begin{aligned} \mathbf{x}(k) &= \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix} \otimes \mathbf{x}(k-1), \quad \text{or equivalently,} \\ \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} &= \begin{bmatrix} \max\{2+x_1(k-1), 5+x_2(k-1)\} \\ \max\{3+x_1(k-1), 3+x_2(k-1)\} \end{bmatrix}. \end{aligned} \tag{2.2}$$

The precedence graph of A is shown in Fig. 2.1 (left) and A is a row-finite matrix. □

The notion of irreducible matrix, to be used shortly, can be given via that of precedence graph.

Definition 2.3 (Irreducible Matrix [22, Th. 2.14]) A matrix $A \in \mathbb{R}^{n\times n}_\varepsilon$ is called irreducible if its precedence graph $\mathcal{G}(A)$ is strongly connected. □

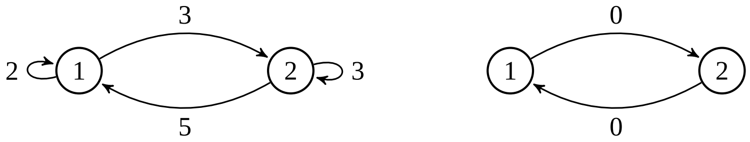

Fig. 2.1: The left and right plots represent precedence and critical graph of matrix A for the autonomous MPL system in (2.2)*, respectively.*

Recall that a directed graph is strongly connected if for any pair of different vertices i, j of the graph, there exists a path from i to j [22, p. 37]. From a max-algebraic perspective, a matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ is irreducible if the nondiagonal elements of $\bigoplus_{k=1}^{n-1} A^{\otimes k}$ are finite (not equal to ε), since this condition means that for two arbitrary vertices i and j of $\mathcal{G}(A)$ with $i \neq j$ there exists at least one path (of length 1, 2, $\ldots$ or $n-1$) from j to i.

Example For the preceding example in (2.2), since $A(1,2) \neq \varepsilon \neq A(2,1)$, the matrix A is irreducible. Equivalently, notice that the precedence graph in Fig. 2.1 (left) is strongly connected. □

In order to investigate the steady-state or ultimate behavior of an autonomous MPL system, we employ the concept of critical graph, which is constructed from the precedence graph.

Definition 2.4 (Critical Graph [22, Def. 3.94]) For a matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$, the following notions are defined:

A circuit of the precedence graph $\mathcal{G}(A)$ is called *critical* if it has maximum average weight. The *critical graph* $\mathcal{G}^c(A)$ consists of those nodes and arcs of $\mathcal{G}(A)$ that belong to a critical circuit of $\mathcal{G}(A)$. The set of nodes in the critical graph is denoted by $\mathcal{V}^c$. The weights are defined as the usual zero [22, p. 143].

The *cyclicity* of a strongly connected graph is the greatest common divisor of the lengths of all its circuits. The cyclicity of a general graph is the least common multiple of the cyclicities of all its strongly connected subgraphs. The cyclicity of $\mathcal{G}^c(A)$ equals to the value c defined in Proposition 2.1. From now on, we will call it the cyclicity of A. □

Example The autonomous MPL system in (2.2) admits the critical circuit $\{1 \rightarrow 2 \rightarrow 1\}$, which coincides with the critical graph (cf. right plot of Fig. 2.1). Since the critical graph is strongly connected, the max-plus eigenvector is unique [22, Sec. 3.7.2] up to the max-plus multiplication by a finite scalar. Furthermore the cyclicity of A is 2, as also results from Proposition 2.1. □

If A is irreducible, there exists a unique max-plus eigenvalue $\lambda \in \mathbb{R}$ [22, Th. 3.23] and a corresponding eigenspace $E(A) = \{\mathbf{x} \in \mathbb{R}^n : A \otimes \mathbf{x} = \lambda \otimes \mathbf{x}\}$ [22, Sec. 3.7.2]. From a graph-theoretical point of view, the max-plus eigenvalue is defined as the maximum cycle mean of the associated precedence graph [22, Th. 3.23]. Algorithms have been developed to compute this quantity, e.g. [36, Sec. 4] and [41]. The eigenspace $E(A)$ is the max-plus linear combination of the i-th column of A_λ^+, for $i \in \mathcal{V}^c$ [22, Sec. 3.7.1], where $A_\lambda^+ = \bigoplus_{k=1}^{\infty}((-\lambda) \otimes A)^{\otimes k}$. Thus the eigenspace is a max-plus cone [60, Def. 2.1], which was introduced in [120]. Proposition 2.1 implies $A_\lambda^+ = \bigoplus_{k=1}^{k_0(A)+c-1}(A-\lambda)^{\otimes k}$, which justifies that A_λ^+ can be computed in finite time.

Proposition 2.1 (Length of the Transient Part [22, Sec. 3.7]) Let $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ be an irreducible matrix with max-plus eigenvalue $\lambda \in \mathbb{R}$. There exist $k_0, c \in \mathbb{N}$ such that $A^{\otimes(k+c)} = \lambda^{\otimes c} \otimes A^{\otimes k}$, for all $k \geq k_0$. The smallest k_0 and c verifying the property are defined as the length of the transient part[1] and the cyclicity, respectively. $\square$

Proposition 2.1 allows to establish the existence of a periodic behavior. Given an initial condition $\mathbf{x}(0) \in \mathbb{R}^n$, there exists a finite $k_0(\mathbf{x}(0))$, such that $\mathbf{x}(k+c) = \lambda^{\otimes c} \otimes \mathbf{x}(k)$, for all $k \geq k_0(\mathbf{x}(0))$. Notice that we can seek a specific length of the transient part $k_0(\mathbf{x}(0))$, in general less conservative than the global $k_0 = k_0(A)$, as in Proposition 2.1. Upper bounds for the length of the transient part k_0 and for its computation have been discussed in [65, Ths. 10 and 13] and more recently in [31].

The complete set of periodic behaviors are encompassed by the eigenspace of $A^{\otimes c}$, where c is the cyclicity of A. It is formulated as $E(A^{\otimes c}) = \{\mathbf{x} \in \mathbb{R}^n : A^{\otimes c} \otimes \mathbf{x} = \lambda^{\otimes c} \otimes \mathbf{x}\}$ and contains the eigenspace of A, i.e. $E(A) \subseteq E(A^{\otimes c})$.

Example In the numerical example (2.2), from Proposition 2.1 we obtain a max-plus eigenvalue $\lambda = 4$, cyclicity $c = 2$, and a (global) length of the transient part $k_0 = 2$. The specific length of the transient part for $\mathbf{x}(0) = [0, 0]^T$ can be computed observing the trajectory

$$\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}5\\3\end{bmatrix}, \begin{bmatrix}8\\8\end{bmatrix}, \begin{bmatrix}13\\11\end{bmatrix}, \begin{bmatrix}16\\16\end{bmatrix}, \begin{bmatrix}21\\19\end{bmatrix}, \begin{bmatrix}24\\24\end{bmatrix}, \begin{bmatrix}29\\27\end{bmatrix}, \begin{bmatrix}32\\32\end{bmatrix}, \begin{bmatrix}37\\35\end{bmatrix}, \begin{bmatrix}40\\40\end{bmatrix}, \begin{bmatrix}45\\43\end{bmatrix}, \begin{bmatrix}48\\48\end{bmatrix}, \ldots$$

Notice that the periodic behavior occurs immediately, i.e. $k_0([0,0]^T) = 0$, and shows a period equal to 2, namely $\mathbf{x}(2) = 4^{\otimes 2} \otimes \mathbf{x}(0) = 8 + \mathbf{x}(0)$. Furthermore notice that $\mathbf{x}(k+2) = 8 \otimes \mathbf{x}(k)$, for $k \in \mathbb{N} \cup \{0\}$.

By using [22, Ths. 3.100 and 3.101], the eigenspace of A is $E(A) = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 = 1\}$ and the complete periodic behaviors are $E(A^{\otimes 2}) = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 2\}$. $\square$

For the backward reachability analysis we introduce the quantity $k_{\emptyset}(\mathbf{x})$, for any given $\mathbf{x} \in \mathbb{R}^n \setminus E(A^{\otimes c})$, as the smallest k such that the system of max-plus linear equations $A^{\otimes k} \otimes \mathbf{x}' = \mathbf{x}$ does not have a solution. (Practically, there is no point $\mathbf{x}' \in \mathbb{R}^n$ that can reach $\mathbf{x}$ in $k_{\emptyset}$ steps or more.) The solution can be computed by using the method in [22, Sec. 3.2.3.2]. Otherwise if $\mathbf{x} \in E(A^{\otimes c})$, $k_{\emptyset}(\mathbf{x})$ is set to 0. This (arguably counter-intuitive) definition will be useful for the ensuing work. It is easy to see that the quantity can be bounded as $k_{\emptyset}(\mathbf{x}) \leq k_0(A) - k_0(\mathbf{x}) + 1$, for each $\mathbf{x} \in \mathbb{R}^n$.

Definition 2.5 (Length of Transient Part of a Set) Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a nonempty set, $k_0(\mathcal{X}) = \max_{\mathbf{x} \in \mathcal{X}} k_0(\mathbf{x})$ and $k_{\emptyset}(\mathcal{X}) = \max_{\mathbf{x} \in \mathcal{X}} k_{\emptyset}(\mathbf{x})$. $\square$

A nonautonomous MPL system [22, Cor. 2.82] is defined by embedding an external input u in the dynamics of (2.1) as:

$$\mathbf{x}(k) = A \otimes \mathbf{x}(k-1) \oplus B \otimes \mathbf{u}(k), \tag{2.3}$$

where $A \in \mathbb{R}_{\varepsilon}^{n\times n}$, $B \in \mathbb{R}_{\varepsilon}^{n\times m}$, $\mathbf{x}(k-1) \in \mathbb{R}^n$, $\mathbf{u}(k) \in \mathbb{R}^m$, for $k \in \mathbb{N}$. As suggested in [22, Sec. 2.5.4], the nonautonomous MPL system (2.3) can be transformed into an augmented MPL system

$$\mathbf{x}(k) = \bar{A} \otimes \bar{\mathbf{x}}(k-1), \tag{2.4}$$

where $\bar{A} = [A, B]$, $\bar{\mathbf{x}}(k-1) = [\mathbf{x}(k-1)^T, \mathbf{u}(k)^T]^T$.

---

[1] Length of transient part is also called coupling time [37, 68].

Example A timetable can be incorporated in (2.2) as the input [68, p. 137]. We obtain a nonautonomous MPL system

$$\mathbf{x}(k) = \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix} \otimes \mathbf{x}(k-1) \oplus \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 0 \end{bmatrix} \otimes \mathbf{u}(k). \tag{2.5}$$

The augmented MPL system is simply

$$\mathbf{x}(k) = \begin{bmatrix} 2 & 5 & 0 & \varepsilon \\ 3 & 3 & \varepsilon & 0 \end{bmatrix} \otimes \bar{\mathbf{x}}(k-1), \tag{2.6}$$

where $\mathbf{x}(k) \in \mathbb{R}^2$ and $\bar{\mathbf{x}}(k-1) \in \mathbb{R}^4$, for $k \in \mathbb{N}$. □

2.2 Related Models

This section introduces models that are related to MPL systems, and that are going to be used throughout the thesis. Min-Plus-Linear (MiPL) systems and Switching Min-Plus-Linear (MiPL) systems are used to model network calculus elements, whereas Piecewise Switched Affine (PWSA) systems are used to construct an abstraction of switching MiPL systems (cf. Chapter 5). Finally finite abstractions of Stochastic Max-Plus-Linear (SMPL) systems is discussed in Chapter 6.

2.2.1 Min-Plus-Linear Systems

Min-Plus-Linear (MiPL) systems are the dual of MPL systems. MiPL systems are the time-domain description of timed event graph and are based on min-plus algebra, whereas MPL systems are the event-domain description of timed event graph and are based on max-plus algebra.

Define $\mathbb{R}_\top$, $\mathbb{N}_\top$ and $\top$ respectively as $\mathbb{R} \cup \{\top\}$, $\mathbb{N} \cup \{\top\}$ and $+\infty$. For $\alpha, \beta \in \mathbb{R}_\top$, introduce the two operations

$$\alpha \oplus' \beta = \min\{\alpha, \beta\}^2 \qquad \text{and} \qquad \alpha \otimes \beta = \alpha + \beta,$$

where the element $\top$ is considered to be absorbing w.r.t. $\otimes$, namely $\alpha \otimes \top = \top$ for all $\alpha \in \mathbb{R}_\top$. Given $\beta \in \mathbb{R}$, the min-algebraic power of $\alpha \in \mathbb{R}$ is denoted by $\alpha^{\otimes \beta}$ and corresponds to $\alpha\beta$ in the conventional algebra. The definition of min-algebraic and max-algebraic power is the same. The rules for the order of evaluation of the min-algebraic operators correspond to those of conventional algebra: min-algebraic power has the highest priority, and min-algebraic multiplication has a higher precedence than min-algebraic addition. The basic min-algebraic operations are extended to matrices as follows. If $A, B \in \mathbb{R}_\top^{m \times n}$; $C \in \mathbb{R}_\top^{m \times p}$; $D \in \mathbb{R}_\top^{p \times n}$; and $\alpha \in \mathbb{R}_\top$,

$$\begin{aligned} [\alpha \oplus' A](i,j) &= \alpha \oplus' A(i,j), \\ [A \oplus' B](i,j) &= A(i,j) \oplus' B(i,j), \\ [C \otimes' D](i,j) &= \bigoplus_{k=1}^{p}{}' C(i,k) \otimes D(k,j), \end{aligned}$$

---

[2] For the minimization operator, the author follows the notation used in [108, p. 380].

for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Notice the analogy between $\oplus'$, $\otimes'$ and $+$, $\times$ for matrix and vector operations in the conventional algebra. Given $m \in \mathbb{N}$, the m-th min-algebraic power of $A \in \mathbb{R}_{\top}^{n\times n}$ is denoted by $A^{\otimes' m}$ and corresponds to $A \otimes' \cdots \otimes' A$ (m times). Notice that $A^{\otimes' 0}$ is an n-dimensional min-plus identity matrix, i.e. the diagonal and nondiagonal elements are 0 and $\top$, respectively. In this thesis, the following notation is adopted for reasons of convenience: a vector with each component that equals to 0 (resp. $+\infty$) is also denoted by 0 (resp. $\top$). Furthermore, the state space is taken to be $\mathbb{R}^n$ (rather than $\mathbb{R}_{\top}^n$).

Remark In matrix operations, the notation of multiplication operator in max-plus algebra and min-plus algebra is different, since their definitions are also different. In max-plus algebra, the addition is defined as maximum, whereas in min-plus algebra, the addition is defined as minimum. On the other hand in scalar operations, the symbol of max-algebraic and min-algebraic multiplications are the same, since both symbols are interpreted as the usual addition. □

A Min-Plus-Linear (MiPL) system is defined as:

$$\mathbf{x}(k) = A \otimes' \mathbf{x}(k-1) \oplus' B \otimes' \mathbf{u}(k), \tag{2.7}$$

where $A \in (\mathbb{N}_{\top} \cup \{0\})^{n\times n}$, $B \in (\mathbb{N}_{\top} \cup \{0\})^{n\times m}$, $\mathbf{x}(k-1) \in (\mathbb{N} \cup \{0\})^n$, $\mathbf{u}(k) \in (\mathbb{N} \cup \{0\})^m$, for $k \in \mathbb{N}$. If the input matrix B contains at least a finite (not equal to $\top$) element, the MiPL system is called nonautonomous, otherwise it is called autonomous. MiPL systems are used to describe the evolution of timed event graphs in the time domain [22, Sec. 5.2]. Here the independent variable k denotes time. The state $\mathbf{x}(k)$ is a counter that represents the number of "events" observed up to and including time k. Each event is assumed to occur instantaneously [22, p. 215]. Thus $\mathbf{x}(\cdot)$ takes values in the set of nonnegative integers. As related models, MPL systems are used to describe the evolution of timed event graphs in the event domain.

Example Consider the following MiPL system representing a simple railway network between two connected stations [68, Sec. 0.5]. The state variables $x_i(k)$ for $i = 1,2$ denote the number of trains that have left station i up to and including time k:

$$\begin{aligned} x_1(k) &= \min\{1 + x_1(k-2), 1 + x_2(k-5)\}, \\ x_2(k) &= \min\{1 + x_1(k-3), 1 + x_2(k-3)\}. \end{aligned}$$

With the introduction of auxiliary variables the MiPL system can be written as a set of first-order recurrence relations as in (2.7). □

2.2.2 Switching Min-Plus-Linear Systems

A switching MiPL system is a discrete-event system that can switch between different modes of operation, where the dynamics in each mode are described by MiPL equations. In Petri-net theory, a system with this property is called free choice Petri nets [48]. Let the switching MiPL system be in mode $\ell(k) \in \{1, \ldots, n_m\}$ at step k, the dynamics are described by the following MiPL equation

$$\mathbf{x}(k) = A^{(\ell(k))} \otimes' \mathbf{x}(k-1) \oplus' B^{(\ell(k))} \otimes' \mathbf{u}(k), \tag{2.8}$$

where $A^{(\ell(k))} \in (\mathbb{N}_\top \cup \{0\})^{n\times n}$, $B^{(\ell(k))} \in (\mathbb{N}_\top \cup \{0\})^{n\times m}$, $\mathbf{x}(k-1) \in (\mathbb{N} \cup \{0\})^n$, $\mathbf{u}(k) \in (\mathbb{N} \cup \{0\})^m$, for $k \in \mathbb{N}$. If there exists a mode ℓ such that the input matrix $B^{(\ell)}$ contains at least a finite (not equal to $\top$) element, the switching MiPL system is called nonautonomous, otherwise it is called autonomous. The mode that is active at each step can be either assumed as a control variable [109] or assumed to be chosen in a purely nondeterministic fashion, i.e. the outcome is not known a priori. In the latter case, we cannot control the mode at each step.

2.2.3 Stochastic Max-Plus-Linear Systems

Stochastic Max-Plus-Linear (SMPL) systems [67, 100, 105] are MPL systems where the time duration (i.e. the processing or transportation times) are now characterized by random quantities. An autonomous SMPL system is defined as:

$$\mathbf{x}(k) = A(k) \otimes \mathbf{x}(k-1), \tag{2.9}$$

where $\mathbf{x}(k-1) \in \mathbb{R}^n$; each entry of the state matrix $A(k)$ is independent and identically distributed w.r.t. $k \in \mathbb{N}$; and $A_{ij}(\cdot)$ are independent for all $i, j \in \{1,\ldots,n\}$[3]. The notation $A_{ij}(\cdot)$ represents the entry of matrix $A(\cdot)$ at the i-th row and the j-th column[4]. We assume each random variable has fixed support [67, Def. 1.4.1], i.e. the probability of ε is either 0 or 1. The random sequence $\{A_{ij}(\cdot)\}$ is then characterized by a given density function $t_{ij}(\cdot)$ and corresponding distribution function $T_{ij}(\cdot)$ (cf. Theorem 2.1).

The independent variable k denotes an increasing event index, whereas the state variable $\mathbf{x}(k)$ defines the (continuous) time of occurrence of the k-th event. Since this thesis is based exclusively on autonomous (that is, not non-deterministic) SMPL systems, the adjective will be dropped for simplicity.

Example Consider the following SMPL system representing a simple railway network between two connected stations. The state variables $x_i(k)$ for $i = 1, 2$ denote the time of the k-th departure at station i:

$$\begin{gathered}\mathbf{x}(k) = A(k) \otimes \mathbf{x}(k-1),\;\; A(k) = \begin{bmatrix} 2+e_{11}(k) & 5+e_{12}(k) \\ 3+e_{21}(k) & 3+e_{22}(k)\end{bmatrix} \quad \text{or equivalently,} \\ \begin{bmatrix} x_1(k) \\ x_2(k)\end{bmatrix} = \begin{bmatrix} \max\{2+e_{11}(k)+x_1(k-1), 5+e_{12}(k)+x_2(k-1)\} \\ \max\{3+e_{21}(k)+x_1(k-1), 3+e_{22}(k)+x_2(k-1)\}\end{bmatrix},\end{gathered} \tag{2.10}$$

where we have assumed that $e_{11}(\cdot) \sim Exp(1)$, $e_{12}(\cdot) \sim Exp(5/2)$, $e_{21}(\cdot) \sim Exp(3/2)$, and $e_{22}(\cdot) \sim Exp(3/2)$, and $Exp(\mu)$ represents the exponential distribution with mean μ. Notice that $A_{ij}(\cdot)$ denotes the traveling time from station j to station i and amounts to a deterministic constant plus a delay modeled by the random variable $e_{ij}(\cdot)$. A few sample trajectories of the SMPL system, initialized at $\mathbf{x}(0) = [1,0]^T$, are displayed in Fig. 2.2. Note that when all random delays are assumed to be equal to zero, the above deterministic system admits the unique solution $\mathbf{x}(k) = \mathbf{x}(0) + 4k = [1+4k, 4k]^T$, where 4 is the max-plus eigenvalue of matrix A, and $[1,0]^T$ is the corresponding eigenvector of the deterministic MPL system

---

[3] Notice that, for deterministic MPL systems, the matrix A is instead given and time-invariant (cf. Section 2.1).

[4] Recall that, for time-invariant matrix A, the notation for the entry at i-th row and j-th column is $A(i, j)$ (cf. page 7).

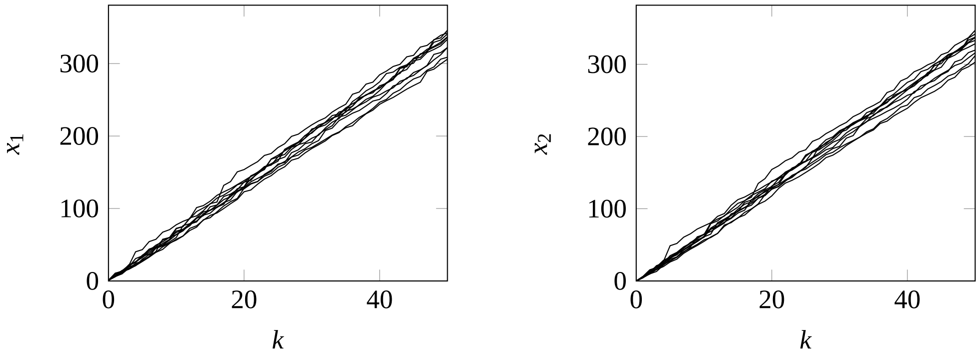

Fig. 2.2: Sample trajectories of the SMPL system in (2.10) *for 50 discrete steps (horizontal axis) and both coordinates (vertical axis).*

[22]. Such a max-plus eigenvalue can be used as a lower bound for the period of a regular schedule for the train departures. □

The next theorem shows that the SMPL system can be described as a discrete-time homogeneous Markov process. The translation of SMPL system into a Markov process has been discussed in the literature [22, 47, 98, 100, 105]. In the literature [22, Th. 8.44], [100, p. 300], [105, Prop. 3.1], [47, Th. 3.2], entries of the state vector are normalized w.r.t. the first entry, whereas in our work entries of the state vector are normalized w.r.t. a regular schedule. For didactical purposes, we present the following theorem together with its complete proof.

Theorem 2.1 The SMPL system in (2.9) is fully characterized by the following conditional density function

$$t^x(\overline{\mathbf{x}}|\mathbf{x}) = \prod_{i=1}^{n} t_i^x(\overline{x_i}|\mathbf{x}), \text{ where}$$

$$t_i^x(\overline{x_i}|\mathbf{x}) = \sum_{j=1}^{n} \left[t_{ij}(\overline{x_i} - x_j) \prod_{k=1, k\neq j}^{n} T_{ik}(\overline{x_i} - x_k) \right], \text{ for all } i \in \{1,\dots,n\},$$

for $\overline{\mathbf{x}}, \mathbf{x} \in \mathbb{R}^n$. The notation $t^x(\overline{\mathbf{x}}|\mathbf{x})$ represents the conditional density function of the next state $\overline{\mathbf{x}}$ w.r.t. the current state $\mathbf{x}$. The notation $t_i^x(\overline{x_i}|\mathbf{x})$ represents the conditional density function of the i-th component of the next state $\overline{x_i}$ w.r.t. the current state $\mathbf{x}$ for all $i \in \{1,\dots,n\}$. The notation $T_{ij}(\cdot)$ represents the distribution function associated with the density function $t_{ij}(\cdot)$ for all $i, j \in \{1,\dots,n\}$. □

Proof The independence property of $A_{ij}(\cdot)$, for all $i, j \in \{1,\dots,n\}$, leads to the multiplicative expression of $t^x(\overline{\mathbf{x}}|\mathbf{x})$. In order to show the expression of the components $t_i^x(\overline{x_i}|\mathbf{x})$, first we compute the i-th conditional distribution function $T_i^x(\overline{x_i}|\mathbf{x})$, then we compute the i-th conditional density function $t_i^x(\overline{x_i}|\mathbf{x})$ by taking the derivative of $T_i^x(\overline{x_i}|\mathbf{x})$ w.r.t. $\overline{x_i}$:

$$\begin{aligned} T_i^x(\overline{x_i}|\mathbf{x}) &= \Pr\{\max\{A_{i1} + x_1, \dots, A_{in} + x_n\} \leq \overline{x_i}|\mathbf{x}\}, \\ &= \Pr\{A_{i1} + x_1 \leq \overline{x_i}, \dots, A_{in} + x_n \leq \overline{x_i}|\mathbf{x}\}, \end{aligned}$$

$$= \prod_{j=1}^{n} \Pr\{A_{ij} \le \bar{x}_i - x_j | \mathbf{x}\} = \prod_{j=1}^{n} T_{ij}(\bar{x}_i - x_j|\mathbf{x}).$$

The notation $T_i^x(\bar{x}_i|\mathbf{x})$ represents the conditional distribution function of the i-th component of the next state $\bar{x}_i$ w.r.t. the current state $\mathbf{x}$ for all $i \in \{1,\dots,n\}$. Finally one can show by virtue of simple algebraic manipulations that the derivative of $T_i^x(\bar{x}_i|\mathbf{x})$ w.r.t. $\bar{x}_i$ coincides with the expression of $t_i^x(\bar{x}_i|\mathbf{x})$. □

2.2.4 Piecewise-Affine Systems

Piece-wise Affine (PWA) systems are characterized by a cover of the state space and by affine (linear, plus a constant) dynamics within each set of the cover [87, 107]. PWA systems are well-posed if the next state and the next output are uniquely solvable once the current state and the current input are specified. PWA systems are sufficiently expressive to model a large number of physical processes, such as systems with static nonlinearities (for instance, actuator saturation), and they can approximate nonlinear dynamics with arbitrary accuracy via multiple linearizations at different operating points [27, p. 1864]. PWA systems have been studied by several authors [27, 73, 75, 87, 107, 113].

This section discusses PWA systems generated by an autonomous and by a nonautonomous MPL system [66]. The obtained PWA systems are well-posed because the autonomous and nonautonomous MPL systems are also well-posed. The construction of PWA systems has a combinatorial complexity. In order to improve the performance, we propose to use a backtracking approach. The PWA system will play a fundamental role in the abstraction procedure and reachability analysis of MPL systems.

Every MPL system characterized by a generic row-finite matrix $A \in \mathbb{R}_\varepsilon^{n\times p}$ can be expressed as a PWA system in the event domain [66, Sec. 3]. The affine dynamics, along with the corresponding region on the state space, can be constructed from the coefficients $\mathbf{g} = (g_1,\dots,g_n) \in \{1,\dots,p\}^n$. For each i, the coefficient g_i characterizes the maximum term in the i-th state equation $x_i(k) = \max\{A(i,1)+x_1,\dots,A(i,p)+x_p\}$, that is $A(i,j)+x_j \le A(i,g_i)+x_{g_i}$, for all $j = 1,\dots,p$[5]. It follows that the set of states corresponding to $\mathbf{g}$, denoted by $R_\mathbf{g}$, is

$$R_\mathbf{g} = \bigcap_{i=1}^{n}\bigcap_{j=1}^{p}\{\mathbf{x}\in\mathbb{R}^n : A(i,j)+x_j \le A(i,g_i)+x_{g_i}\}. \tag{2.11}$$

Alternatively, a point $\mathbf{x}\in\mathbb{R}^n$ is in $R_\mathbf{g}$ if $\max_{j=1,\dots,p} A(i,j)+x_j = A(i,g_i)+x_{g_i}$, for all $i = 1,\dots,n$.

The affine dynamics that are active in $R_\mathbf{g}$ follow directly from the definition of $\mathbf{g}$ (see previous paragraph) as

$$x_i(k) = x_{g_i}(k-1) + A(i,g_i), \qquad i = 1,\dots,n. \tag{2.12}$$

Given a row-finite state matrix A, Algorithm 2.1 describes a general procedure to construct a PWA system corresponding to an autonomous MPL system. Similarly, if we run

---

[5] The way $\mathbf{g}$ is defined is closely related to the idea of a policy [36] in Howard's algorithm, i.e. both definitions choose a single finite element in each row of matrix A. Howard's algorithm, also known as the policy iteration algorithm, is an iterative algorithm for computing a generalized eigenmode. This algorithm consists of two parts: value determination and policy improvement. In value determination, the aim is to determine a generalized eigenmode from a given matrix A and a given policy.

the algorithm with the augmented matrix $\bar{A}$, we obtain a PWA system related to the nonautonomous MPL system. Correspondingly, the parameter p above equals n or $n+m$. On the side, notice that the affine dynamics associated with a dynamical system generated by Algorithm 2.1 are a special case of the general PWA dynamics as defined in [107, Sec. 1].

Algorithm 2.1 Generation of a PWA system from a row-finite MPL matrix

```
Input: $A \in \mathbb{R}_{\varepsilon}^{n\times p}$, a row-finite max-plus matrix
Output: $\mathbf{R},\mathbf{A},\mathbf{B}$, a PWA system over $\mathbb{R}^p$,
  where $\mathbf{R}$ is a set of regions and $\mathbf{A},\mathbf{B}$ represent a set of affine dynamics

initialize $\mathbf{R},\mathbf{A},\mathbf{B}$ with the empty set
for all $\mathbf{g} \in \{1,\dots,p\}^n$ do
  generate region $R_{\mathbf{g}}$ according to (2.11)
  if $R_{\mathbf{g}}$ is not empty then
    generate matrices $A_{\mathbf{g}}, B_{\mathbf{g}}$ s.t. $\mathbf{x}(k) = A_{\mathbf{g}}\mathbf{x}(k-1) + B_{\mathbf{g}}$ corresponding to (2.12)
    save the results, i.e. $\mathbf{R} := \mathbf{R} \cup \{R_{\mathbf{g}}\}$, $\mathbf{A} := \mathbf{A} \cup \{A_{\mathbf{g}}\}$, $\mathbf{B} := \mathbf{B} \cup \{B_{\mathbf{g}}\}$
  end if
end for
```

The crucial observation that allows for an improvement of the complexity is that it is not necessary to iterate over all possible coefficients as in Algorithm 2.1. Instead, we can apply a backtracking technique. In the backtracking approach, the partial coefficients are $(g_1,\dots,g_k)$ for $k=1,\dots,n$ and the corresponding region is

$$R_{(g_1,\dots,g_k)} = \bigcap_{i=1}^{k}\bigcap_{j=1}^{n}\{\mathbf{x} \in \mathbb{R}^n : A(i,g_i) + x_{g_i} \geq A(i,j) + x_j\}.$$

Notice that if the region associated with some partial coefficient $(g_1,\dots,g_k)$ is empty, then the regions of the corresponding coefficients $(g_1,\dots,g_n)$ are also empty, for all $g_{k+1},\dots,g_n$. The set of all coefficients can be represented as a potential search tree. For a 2-dimensional MPL system, the potential search tree is given in Fig. 2.3 (left). The backtracking algorithm traverses the tree recursively, starting from the root, in a depth-first order. At each node, the algorithm checks whether the corresponding region is empty. If the region is empty, the whole sub-tree rooted at the node is skipped (pruned).

Example With reference to the autonomous MPL example in (2.2), the obtained PWA system is

$$\mathbf{x}(k) = \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \mathbf{x}(k-1) + \begin{bmatrix} 2 \\ 3 \end{bmatrix}, & \text{if } \mathbf{x}(k-1) \in R_{(1,1)}, \\[2ex] \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{x}(k-1) + \begin{bmatrix} 5 \\ 3 \end{bmatrix}, & \text{if } \mathbf{x}(k-1) \in R_{(2,1)}, \\[2ex] \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{x}(k-1) + \begin{bmatrix} 5 \\ 3 \end{bmatrix}, & \text{if } \mathbf{x}(k-1) \in R_{(2,2)}, \end{cases}$$

where $R_{(1,1)} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 3\}$, $R_{(2,1)} = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 3\}$, and $R_{(2,2)} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \leq 0\}$, as depicted in Fig. 2.3 (right). Region $R_{(1,2)}$ does not appear since

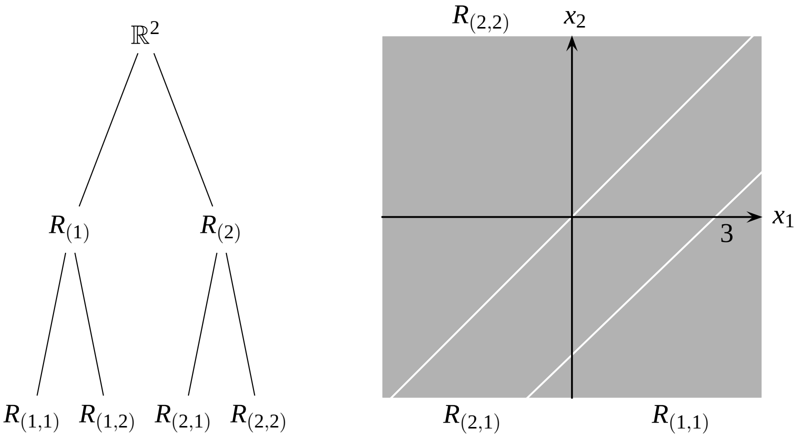

Fig. 2.3: (Left plot) Potential search tree for a 2-dimensional MPL system. (Right plot) Regions associated with the PWA system generated by the autonomous MPL system in (2.2)*.*

it corresponds to an empty set. As explained above, the affine dynamics corresponding to a region are characterized by set **g**: for example the affine dynamics of $R_{(2,1)}$ are given by $x_1(k) = x_2(k-1)+5$, $x_2(k) = x_1(k-1)+3$. Similarly, for the nonautonomous MPL system (2.5), the nonempty regions of the corresponding PWA system are: $\bar{R}_{(1,1)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \geq 3,\ x_1 - u_1 \geq -2,\ x_1 - u_2 \geq -3\}$; $\bar{R}_{(1,4)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \geq 3,\ x_1 - u_1 \geq -2,\ x_1 - u_2 \leq -3,\ x_2 - u_2 \leq -3\}$; $\bar{R}_{(2,1)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : 0 \leq x_1 - x_2 \leq 3,\ x_1 - u_2 \geq -3,\ x_2 - u_1 \geq -5\}$; $\bar{R}_{(2,2)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \leq 0,\ x_2 - u_1 \geq -5,\ x_2 - u_2 \geq -3\}$; $\bar{R}_{(2,4)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \leq 3,\ x_1 - u_2 \leq -3,\ x_2 - u_1 \geq -5,\ x_2 - u_2 \leq -3\}$; $\bar{R}_{(3,1)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \geq 0,\ x_1 - u_1 \leq -2,\ x_1 - u_2 \geq -3,\ x_2 - u_1 \leq -5\}$; $\bar{R}_{(3,2)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \leq 0,\ x_1 - u_1 \leq -2,\ x_2 - u_1 \leq -5,\ x_2 - u_2 \geq -3\}$; $\bar{R}_{(3,4)} = \{\bar{\mathbf{x}} \in \mathbb{R}^4 : x_1 - u_1 \leq -2,\ x_1 - u_2 \leq -3,\ x_2 - u_1 \leq -5,\ x_2 - u_2 \leq -3\}$. □

Remark Every MiPL system characterized by a generic row-finite matrix $A \in \mathbb{R}_{\top}^{n \times p}$ can also be transformed to a PWA system in the time domain. A matrix $A \in \mathbb{R}_{\top}^{n \times p}$ is row-finite if A contains at least one element different from $\top$ in each row (cf. Definition 2.2). The affine dynamics and the corresponding region on the state space, are constructed from the coefficients $\mathbf{g} = (g_1, \ldots, g_n) \in \{1, \ldots, p\}^n$. For each i, the coefficient g_i characterizes the minimum term in the i-th state equation $x_i(k) = \min\{A(i,1)+x_1, \ldots, A(i,p)+x_p\}$, that is $A(i,j)+x_j \geq A(i,g_i)+x_{g_i}$, for all $j = 1, \ldots, p$. It follows that the set of states corresponding to **g**, denoted by $R_{\mathbf{g}}$, is

$$R_{\mathbf{g}} = \bigcap_{i=1}^{n} \bigcap_{j=1}^{p} \{\mathbf{x} \in \mathbb{R}^n : A(i,j) + x_j \geq A(i,g_i) + x_{g_i}\}.$$

The affine dynamics that are active in $R_{\mathbf{g}}$ follow directly from the definition of **g** (see previous paragraph) as

$$x_i(k) = x_{g_i}(k-1) + A(i,g_i), \qquad i = 1, \ldots, n.$$

Algorithm 2.1 can be tailored to generate a PWA system from a row-finite MiPL matrix. □

Implementation In VeriSiMPL version 1.4, the procedure to construct a PWA system from an autonomous MPL and MiPL system has been implemented in the function `mpl2pwa`. In the case of autonomous MPL system, the function `mpl2pwa` requires a row-finite state matrix (`Ampl`) and generates a PWA system characterized by a collection of regions (`D`) and a set of affine dynamics (`A`,`B`). The affine dynamics that are active in the j-th region are characterized by the j-th column of both `A` and `B`. Each column of `A` and the corresponding column of `B` contain the coefficients $[g_1,\ldots,g_n]^T$ and the constants $[A(1,g_1),\ldots,A(n,g_n)]^T$, respectively. The data structure of `D` will be discussed in Section 3.2.1.

Considering the autonomous MPL example in (2.2), the following MATLAB script generates the PWA system:

```
>> Ampl = [2 5;3 3], [A,B,D] = mpl2pwa(Ampl)
```

It will become clear in Section 3.2.1 that the nonempty regions of the PWA system produced by the script are: $R_{(1,1)} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 3\}$, $R_{(2,1)} = \{\mathbf{x} \in \mathbb{R}^2 : e \leq x_1 - x_2 \leq 3\}$, and $R_{(2,2)} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \leq e\}$. The affine dynamics corresponding to a region $R_{\mathbf{g}}$ are characterized by $\mathbf{g}$, e.g. those for region $R_{(2,1)}$ are given by $x_1(k) = x_2(k-1)+5$, $x_2(k) = x_1(k-1)+3$.

The function `mpl2pwa` can be also used to determine the PWA system generated by an augmented MPL system. In this case, the input of `mpl2pwa` is the augmented matrix and the output is a PWA system in the augmented space $\mathbb{R}^{m+n}$.

In order to determine a PWA system from an autonomous MiPL system, the function `mpl2pwa` is called with two arguments. The first argument is the row-finite state matrix (`Amipl`) and the second argument is the Boolean constant `false`. The function `mpl2pwa` can be also used to determine the PWA system generated by an augmented MiPL system. In this case, the function `mpl2pwa` is also called with two arguments, i.e. the augmented matrix and the Boolean constant `false`. □

2.2.5 Piecewise Switched Affine Systems

This section discusses Piece-wise Switched Affine (PWSA) systems generated by switching MiPL systems. PWSA systems are an extension of PWA systems. Recall that PWA systems are described by a set of affine dynamics defined over a corresponding region in the state space. In PWSA system, the dynamics that are active in each region are switched affine. Switched affine dynamics have different modes of operation, where in each mode the dynamics are affine. The PWSA system will play a key role in the abstraction procedure of switching MiPL systems.

Every switching MiPL system characterized by a collection of $n \times p$ generic row-finite matrices $A^{(1)},\ldots,A^{(n_m)}$ can be expressed as a PWSA system in the time domain. Let the PWA system generated by $A^{(\ell)}$ be characterized by $\mathbf{g}^{(\ell)}$. The switched affine dynamics, along with the corresponding region on the state space, can be constructed from coefficients $(\mathbf{g}^{(1)},\ldots,\mathbf{g}^{(n_m)}) \in \{1,\ldots,p\}^n \times \cdots \times \{1,\ldots,p\}^n$. The regions of the PWSA system is the refinement of PWA regions generated by the MiPL dynamics associated with each mode:

$$R_{(\mathbf{g}^{(1)},\ldots,\mathbf{g}^{(n_m)})} = \bigcap_{\ell=1}^{n_m} R_{\mathbf{g}^{(\ell)}} .$$

The collection of affine dynamics that is active in $R_{(\mathbf{g}^{(1)},\ldots,\mathbf{g}^{(n_m)})}$ follows directly from preceding equation and is given by

$$x_i(k) = x_{g_i^{(\ell)}}(k-1) + A(i, g_i^{(\ell)}), \qquad i = 1, \ldots, n, \qquad \text{for each mode } \ \ell = 1, \ldots, n_m.$$

Algorithm 2.1 can be used to construct a PWSA system corresponding to a switching MiPL system. The input matrix is defined as the collection of row-finite matrices stacked vertically, i.e. $[(A^{(1)})^T, \ldots, (A^{(n_m)})^T]^T$.

2.3 Summary

In this chapter we have discussed Max-Plus-Linear (MPL) systems and some of its basic properties. We have then briefly discussed some related models such as Min-Plus-Linear (MiPL) systems, switching MiPL systems, stochastic MPL systems, Piece-wise Affine (PWA) systems, and Piecewise Switched Affine (PWSA) systems. We have shown a procedure to generate PWA systems from MPL systems and from MiPL systems. Similarly we have also shown a procedure to construct PWSA systems from switching MiPL systems.

Chapter 3

Finite Abstractions of Max-Plus-Linear Systems

In this chapter we develop a framework for formal verification of MPL systems. Specifically, we check whether an MPL system with a predefined set of initial states X_0 satisfies an LTL formula over a fixed set of atomic propositions *AP*. We propose the following approach. First a transition system is generated from the given concrete MPL system. Then we generate an abstract transition system that simulates the transition system. By using model checking techniques, we next determine whether the abstract transition system satisfies the given LTL formula. If the LTL formula is satisfied, the concrete transition system also satisfies the LTL formula. Otherwise if the LTL formula is not satisfied, it does not imply that the concrete transition system does not satisfy the LTL formula. In this case, a partition refinement technique can be used to obtain a more precise abstraction.

The computational aspects related to the abstraction procedure have been under particular scrutiny, and have brought to 1) the selection of DBM as a framework for the representation and manipulation of regions over the state and control spaces; and 2) the use of PWA representations of the MPL dynamics [66], which nicely couples with quantities expressed as DBM. The computational costs of the abstraction procedure are discussed in detail and its overall performance is benchmarked over a case study in Section 3.6.

3.1 Related Work

To the best of the author's knowledge, this contribution represents the first work on finite-state abstractions of MPL systems. The approach to attain abstractions developed in this work is inspired by those developed for other models in [17, 101, 117], and can be interpreted in the context of literature focused on the construction of finite-state (quotient) models of given systems. The construction of quotient systems has been treated in depth in [115, Sec. 0.7] and in [116] for time-invariant linear systems. However this technique cannot be used in our problem because there is no guarantee that the properties of interest are preserved in the quotient system. Notice that we leverage a PWA representation of the given MPL dynamics [66] – a particular case of the PWA system used in [117] – to build the finite-state abstraction. However, techniques for abstractions of PWA systems developed in the literature [117] do not appear to be directly usable in the context of the models derived from MPL systems, since spatial boundaries can non-trivially affect the semantics of the trajectories [117, Rem. 1]. Likewise, related verification approaches developed for timed Petri nets (such as that for safety analysis based on existential zones [4]) do not appear to being exportable to MPL systems.

3.2 Preliminaries

This section introduces Difference-Bound Matrices (DBM), transition systems modeling framework, Linear Temporal Logic (LTL) formula, and finally the notion of abstraction.

3.2.1 Difference-Bound Matrices

This section introduces the definition of a Difference-Bound Matrix (DBM) [51, Sec. 4.1], its canonical-form representation, and the connection with max-plus polyhedra. DBM will be used extensively in the abstraction procedure and reachability of MPL systems.

Definition 3.1 (Difference-Bound Matrix) A DBM in $\mathbb{R}^n$ is the intersection of finitely many sets defined by $x_j - x_i \bowtie_{i,j} \alpha_{i,j}$, where $\bowtie_{i,j} \in \{<, \leq\}$ denotes the strictness of the sign, the specified number $\alpha_{i,j} \in \mathbb{R}_\top$ represents the upper bound, for $i, j \in \{0, \ldots, n\}$ and the value of the special variable x_0 always equal to 0. The sets are characterized by the values of variables $x_1, \ldots, x_n$, which imply that the sets are a subset of $\mathbb{R}^n$. □

The special variable x_0 is used to represent bounds over a single variable: $x_i \leq \alpha$ can be written as $x_i - x_0 \leq \alpha$. In the following, a "stripe" is defined as a DBM that does not contain x_0. Definition 3.1 can be likewise given over the input and augmented spaces.

Implementation VeriSiMPL represents a DBM in $\mathbb{R}^n$ as a 1×2 cell: the first element is an $(n+1)$-by-$(n+1)$ matrix with entries in the real numbers representing the upper bound α, and the second element is an $(n+1)$-by-$(n+1)$ matrix with entries in the Boolean domain representing the value of $\bowtie$. More precisely, the $(i+1, j+1)$-th element represents the upper bound and the strictness of the sign of $x_j - x_i$, for $i = 0, \ldots, n$ and $j = 0, \ldots, n$ (cf. Definition 3.1).[1] The non-strict sign $\leq$ corresponds to `true` and the strict sign $<$ corresponds to `false`. Furthermore, a collection of DBM is also represented as a 1×2 cell, where the corresponding matrices are stacked along the third dimension. □

Each DBM admits an equivalent and unique canonical-form representation, which is a DBM with the tightest possible bounds [51, Sec. 4.1]. Since computing the canonical-form representation of a DBM is equivalent to the all-pairs shortest path problem over the corresponding potential graph [51, Sec. 4.1], the Floyd-Warshall algorithm [56] can be used over the graph with a complexity that is cubic w.r.t. its dimension.

Example Consider the PWA system generated by the nonautonomous MPL system (2.5). A few regions are not in the canonical-form representation, and can then be expressed as

---

[1] The author was inspired by the definition of precedence graph w.r.t. the state matrix when choosing this representation.

follows: $\mathrm{cf}(\overline{R}_{(1,4)}) = \{\overline{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \geq 3,\ x_1 - u_1 \geq -2,\ x_1 - u_2 \leq -3,\ x_2 - u_2 \leq -6,\ u_1 - u_2 \leq -1\}, \mathrm{cf}(\overline{R}_{(2,1)}) = \{\overline{\mathbf{x}} \in \mathbb{R}^4 : 0 \leq x_1 - x_2 \leq 3,\ x_1 - u_1 \geq -5,\ x_1 - u_2 \geq -3,\ x_2 - u_1 \geq -5,\ x_2 - u_2 \geq -6\}, \mathrm{cf}(\overline{R}_{(2,4)}) = \{\overline{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \leq 3,\ x_1 - u_2 \leq -3,\ x_2 - u_1 \geq -5,\ x_2 - u_2 \leq -3,\ u_1 - u_2 \leq 2\}, \mathrm{cf}(\overline{R}_{(3,1)}) = \{\overline{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \geq 0,\ x_1 - u_1 \leq -2,\ x_1 - u_2 \geq -3,\ x_2 - u_1 \leq -5,\ u_1 - u_2 \geq -1\}, \mathrm{cf}(\overline{R}_{(3,2)}) = \{\overline{\mathbf{x}} \in \mathbb{R}^4 : x_1 - x_2 \leq 0,\ x_1 - u_1 \leq -5,\ x_2 - u_1 \leq -5,\ x_2 - u_2 \geq -3,\ u_1 - u_2 \geq 2\}$, where cf is a generic operator yielding the canonical form [51, Sec. 4.1]. Other regions appear already in canonical form, for instance $\overline{R}_{(1,1)} = \mathrm{cf}(\overline{R}_{(1,1)})$. ☐

One advantage of the canonical-form representation is that it is straightforward to compute orthogonal projections w.r.t. a subset of its variables. This is simply performed by deleting rows and columns corresponding to the complementary variables [51, Sec. 4.1]. The orthogonal projection of a DBM in canonical form is again in canonical form [51, Obs. 1].

Definition 3.2 (Orthogonal Projection) The orthogonal projection w.r.t. the state space X (the input space U) of a region in the augmented space is defined as $proj_X : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ ($proj_U : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^m$), where $proj_X : [\mathbf{x}^T, \mathbf{u}^T]^T \mapsto \mathbf{x}$ ($proj_U : [\mathbf{x}^T, \mathbf{u}^T]^T \mapsto \mathbf{u}$). ☐

Remark The two terms "the orthogonal projection w.r.t. the state space" and "the orthogonal projection w.r.t. the state variables" are used as synonyms. A similar argument holds for "the orthogonal projection w.r.t. the input space" and "the orthogonal projection w.r.t. the input variables". ☐

Another advantage of the canonical-form representation is that its emptiness can be checked very efficiently. By using the potential graph representation, the unfeasible sets of constraints are only those which form a circuit with a strictly negative weight in the graph. As a consequence, in order to test whether a DBM is empty or not, we simply have to check for the existence of such a circuit: this can be achieved by the Bellman-Ford algorithm [26, Sec. 5], which is cubic w.r.t. its dimension. Whenever a DBM is in canonical form, testing for strictly negative cycles can be reduced to checking whether there is an i such that $\bowtie_{i,i}$ is $<$ or $\alpha_{i,i} < 0$. Thus, the complexity of emptiness checking is linear w.r.t. dimension of the DBM.

Implementation In VeriSiMPL version 1.4, the Floyd-Warshall algorithm has been implemented in the function `floyd_warshall`. Given a collection of finitely many DBM, this function generates its canonical-form representation. The following MATLAB script computes the canonical-form representation of $\{\mathbf{x} \in \mathbb{R}^5 : x_1 - x_2 \geq 3, x_1 - x_3 \geq -2, x_1 - x_4 \leq -3, x_2 - x_4 \leq -3\}$:

```
>> D = cell(1,2)
>> D{1} = [0 Inf Inf Inf Inf;Inf 0 -3 2 Inf;
Inf Inf 0 Inf Inf;Inf Inf Inf 0 Inf;Inf -3 -3 Inf 0]
>> D{2} = [true false false false false;false true true true false;
false false true false false;false false false true false;
false true true false true]
>> Dcf = floyd_warshall(D)
```

The canonical-form representation (`Dcf`) is $\{\mathbf{x} \in \mathbb{R}^5 : x_1 - x_2 \geq 3, x_1 - x_3 \geq -2, x_1 - x_4 \leq -3, x_2 - x_4 \leq -6, x_3 - x_4 \leq -1\}$. Notice that the bounds of $x_2 - x_4$ and $x_3 - x_4$ are tighter.

The procedure to determine the emptiness of a collection of finitely many DBM has been implemented in the function `dbm_isempty` that is included in VeriSiMPL version 1.4. This function returns `true` if the DBM is empty and `false` if the DBM is not empty. The following MATLAB script checks whether the DBM `D` defined above is empty:

```
>> dbm_isempty(D)
```

The result is `false` which means that `D` is not empty. □

Each region and the corresponding affine dynamics of the PWA system generated by Algorithm 2.1 (for both autonomous and nonautonomous MPL systems) can be characterized by a DBM. From (2.11), each region of the PWA system generated by a row-finite max-plus matrix is a DBM in $\mathbb{R}^p$. Each affine dynamics (2.12) can generate a DBM in $\mathbb{R}^p \times \mathbb{R}^n$, which comprises points $(\mathbf{x}(k-1), \mathbf{x}(k)) \in \mathbb{R}^p \times \mathbb{R}^n$ such that $\mathbf{x}(k)$ is the image of $\mathbf{x}(k-1)$, i.e. $\mathbf{x}(k) = A \otimes \mathbf{x}(k-1)$. More precisely, the DBM is obtained by rewriting the expression of the affine dynamics as $\bigcap_{i=1}^{n} \{(\mathbf{x}(k-1), \mathbf{x}(k)) : x_i(k) - x_{g_i}(k-1) \leq A(i, g_i)\} \cap \bigcap_{i=1}^{n} \{(\mathbf{x}(k-1), \mathbf{x}(k)) : x_i(k) - x_{g_i}(k-1) \geq A(i, g_i)\}$.

Looking back at the backtracking approach to generate the PWA system (cf. Section 2.2.4), its worst-case complexity can be formulated as $\mathcal{O}(p^n(np + p^3))$ [8, p. 3043]. This happens if the matrix does not have infinite elements and all regions are nonempty. However, in practice this worst-case is not incurred since many regions can happen to be empty.

Proposition 3.1 ([8, Th. 1]) The image and the inverse image of a DBM with respect to affine dynamics (in particular the PWA expressions in (2.11)-(2.12) generated by an MPL system) is a DBM. □

The general procedure to compute the image of a DBM in $\mathbb{R}^p$ w.r.t. affine dynamics $\mathbb{R}^p \to \mathbb{R}^n$ involves: 1) computing the cross product of the DBM and $\mathbb{R}^n$; then 2) intersecting the cross product with the DBM generated by the expression of the affine dynamics; 3) calculating the canonical form of the obtained intersection; and finally 4) projecting the canonical-form representation over $\{x_1(k), \ldots, x_n(k)\}$. The complexity of computing the image depends critically on the third step and is $\mathcal{O}((n+p)^3)$. The illustration of the procedure to compute the image for $p = 1 = n$ is depicted in Fig. 3.1 (left).

Example Let us compute the image of $\{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, x_1 - x_2 \leq 0\}$ w.r.t. $x'_1 = x_2 + 5$, $x'_2 = x_2 + 3$ by using the above procedure. The cross product of the DBM and $\mathbb{R}^2$ is $\{(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^4 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, x_1 - x_2 \leq 0\}$. The intersection of the cross product and the DBM generated by the expression of the affine dynamics is $\{(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^4 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, x_1 - x_2 \leq 0, x'_1 - x_2 = 5, x'_2 - x_2 = 3\}$. The canonical form of the obtained intersection is $\{(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^4 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, 5 \leq x'_1 \leq 6, 3 \leq x'_2 \leq 4, 0 \leq x_2 - x_1 \leq 1, 5 \leq x'_1 - x_1 \leq 6, 3 \leq x'_2 - x_1 \leq 4, x'_1 - x_2 = 5, x'_2 - x_2 = 3, x'_2 - x'_1 = -2\}$. The projection w.r.t. $\{x'_1, x'_2\}$ is computed by removing all inequalities containing x_1 or x_2, which yields $\{\mathbf{x}' \in \mathbb{R}^2 : 5 \leq x'_1 \leq 6, 3 \leq x'_2 \leq 4, x'_2 - x'_1 = -2\}$.

In VeriSiMPL version 1.4, the procedure to compute the image of a DBM w.r.t. an affine dynamic has been implemented in `dbm_image` as a function. This function requires the

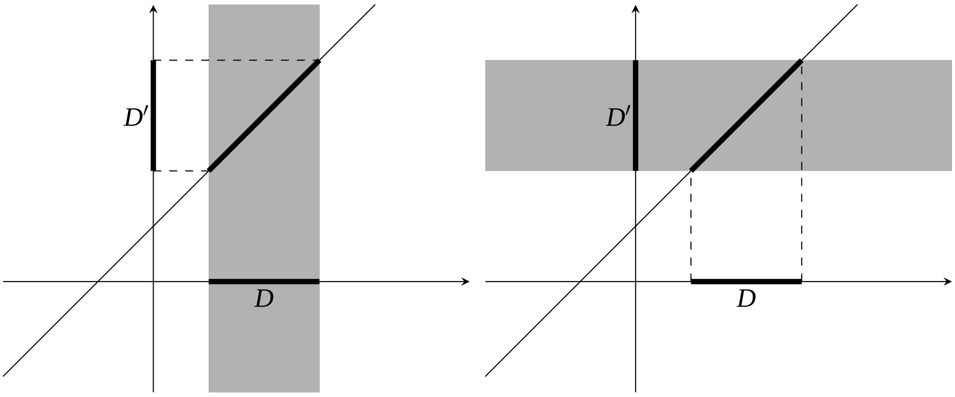

Fig. 3.1: The left and right plots illustrate the algorithms to determine the image and inverse image of a DBM w.r.t. an affine dynamic, respectively.

affine dynamics (`A`,`B`) and the DBM (`D`). The following MATLAB script re-calculates the numerical example in the preceding paragraph:

```
>> A = [2;2], B = [5;3]
>> D = cell(1,2)
>> D{1} = [0 1 1;0 0 Inf;0 0 0]
>> D{2} = [true true true;true true false;true true true]
>> Dim = dbm_image(A,B,D)
```
□

Similarly, the general procedure to compute the inverse image of a DBM in $\mathbb{R}^n$ w.r.t. affine dynamics $\mathbb{R}^p \to \mathbb{R}^n$ involves: 1) computing the cross product of $\mathbb{R}^p$ and the DBM; then 2) intersecting the cross product with the DBM generated by the expression of the affine dynamics; 3) calculating the canonical form of the obtained intersection; finally 4) projecting the canonical-form representation over $\{x_1(k-1),\ldots,x_p(k-1)\}$. The complexity of computing the inverse image is again $\mathcal{O}\left((n+p)^3\right)$. The illustration of the procedure to compute the inverse image for $p = 1 = n$ is shown in Fig. 3.1 (right).

Example Let us determine the inverse image of $\{\mathbf{x}' \in \mathbb{R}^2 : 0 \le x'_1 \le 1, 0 \le x'_2 \le 1\}$ w.r.t. $x'_1 = x_1 + 2$, $x'_2 = x_1 + 3$ by using the discussed procedure. The cross product of $\mathbb{R}^2$ and the DBM is $\{(\mathbf{x},\mathbf{x}') \in \mathbb{R}^4 : 0 \le x'_1 \le 1, 0 \le x'_2 \le 1\}$. The intersection of the cross product and the DBM generated by the expression of the affine dynamics is $\{(\mathbf{x},\mathbf{x}') \in \mathbb{R}^4 : 0 \le x'_1 \le 1, 0 \le x'_2 \le 1, x'_1 - x_1 = 2, x'_2 - x_1 = 3\}$. The canonical form of the obtained intersection is $\{(\mathbf{x},\mathbf{x}') \in \mathbb{R}^4 : x_1 = -2, x'_1 = 0, x'_2 = 1, x'_1 - x_1 = 2, x'_2 - x_1 = 3, x'_2 - x'_1 = 1\}$. The projection w.r.t. $\{x_1, x_2\}$ is computed by removing all inequalities containing x'_1 or x'_2, which yields $\{\mathbf{x} \in \mathbb{R}^2 : x_1 = -2\}$.

In VeriSiMPL version 1.4, the procedure to determine the inverse image of a DBM w.r.t. an affine dynamic has been implemented in the function `dbm_invimage`. This function requires the affine dynamics (`A`,`B`), the DBM (`D`), and dimension of domain of the affine dynamics. The following MATLAB script re-calculates the numerical example in the preceding paragraph:

```
>> A = [1;1], B = [2;3]
>> D = cell(1,2)
>> D{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D{2} = [true true true;true true false;true false true]
>> Dinv = dbm_invimage(A,B,D,2)
```
□

Computing the image and the inverse image of a DBM (in $\mathbb{R}^p$ and in $\mathbb{R}^n$) w.r.t. switched affine dynamics $\mathbb{R}^p \to \mathbb{R}^n$ can be performed by computing the image and inverse image w.r.t. each affine dynamics. The complexity of both cases is $\mathcal{O}(n_m(n+p)^3)$, where n_m is the number of affine dynamics.

The following procedure computes the image of a DBM in $\mathbb{R}^p$ w.r.t. MPL dynamics characterized by $A \in \mathbb{R}_\varepsilon^{n\times p}$ or w.r.t. MiPL dynamics characterized by $A \in \mathbb{R}_\top^{n\times p}$, and uses the corresponding PWA system: 1) intersecting the DBM with each region of the PWA system; then 2) computing the image of nonempty intersections according to the corresponding affine dynamics (cf. Proposition 3.1). The worst-case complexity depends on the last step and amounts to $\mathcal{O}(q_A(n+p)^3)$, where q_A is the number of regions in the PWA system generated by matrix A.

Example Let us compute the image of $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le 1, 0 \le x_2 \le 1\}$ w.r.t. the MPL system (2.2). The intersection of $\mathcal{X}_0$ and the regions is $\mathcal{X}_0 \cap R_{(1,1)} = \emptyset$, $\mathcal{X}_0 \cap R_{(2,1)} = \{\mathbf{x} : 0 \le x_1 \le 1, 0 \le x_2 \le 1, 0 \le x_1 - x_2 \le 3\}$, and $\mathcal{X}_0 \cap R_{(2,2)} = \{\mathbf{x} : 0 \le x_1 \le 1, 0 \le x_2 \le 1, x_1 - x_2 \le 0\}$. Skipping the details, the image of $\mathcal{X}_0 \cap R_{(2,1)}$ and $\mathcal{X}_0 \cap R_{(2,2)}$ is $\{\mathbf{x} : 5 \le x_1 \le 6, 3 \le x_2 \le 4, x_1 - x_2 = 2\}$ and $\{\mathbf{x} : 5 \le x_1 \le 6, 3 \le x_2 \le 4, 1 \le x_1 - x_2 \le 2\}$, respectively. Thus the image of $\mathcal{X}_0$ is $\mathcal{X}_1 = \{\mathbf{x} \in \mathbb{R}^2 : 5 \le x_1 \le 6, 3 \le x_2 \le 4, 1 \le x_1 - x_2 \le 2\}$ as depicted in Fig. 3.2. □

Similarly, the inverse image of a DBM in $\mathbb{R}^n$ w.r.t. the MPL system characterized by $A \in \mathbb{R}_\varepsilon^{n\times p}$ or w.r.t. the MiPL system characterized by $A \in \mathbb{R}_\top^{n\times p}$ can be computed via its PWA representation: 1) computing the inverse image of the DBM w.r.t. each affine dynamics of the PWA system (cf. Proposition 3.1); then 2) intersecting the inverse image with the corresponding region, which is a DBM; finally 3) collecting the nonempty intersections. The worst-case complexity is quantified again as $\mathcal{O}(q_A(n+p)^3)$.

Example Let us compute the inverse image of $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le 1, 0 \le x_2 \le 1\}$ w.r.t. the MPL system (2.2). Without going into the details, the inverse image of $\mathcal{X}_0$ w.r.t. the affine dynamics in $R_{(1,1)}$, $R_{(2,1)}$, and $R_{(2,2)}$ is $\{\mathbf{x} : x_1 = -2\}$, $\{\mathbf{x} : -3 \le x_1 \le -2, -5 \le x_2 \le -4, 1 \le x_1 - x_2 \le 3\}$, and $\emptyset$, respectively. The intersection of the obtained inverse images with the corresponding region is $\{\mathbf{x} : x_1 = -2, x_2 \le -5\}$, $\{\mathbf{x} : -3 \le x_1 \le -2, -5 \le x_2 \le -4, 1 \le x_1 - x_2 \le 3\}$, and $\emptyset$. The inverse image of $\mathcal{X}_0$ is $\mathcal{X}_{-1} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 = -2, x_2 \le -5\} \cup \{\mathbf{x} \in \mathbb{R}^2 : -3 \le x_1 \le -2, -5 \le x_2 \le -4, 1 \le x_1 - x_2 \le 3\}$ as shown in Fig. 3.2. □

The image and the inverse image of a DBM w.r.t. switching MiPL dynamics can be obtained by computing the image and inverse image w.r.t. each MiPL dynamics.

Proposition 3.1 can be extended as follows.

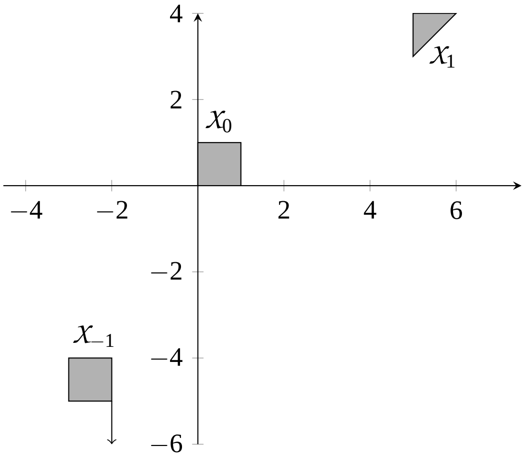

Fig. 3.2: The image and inverse image of $\mathcal{X}_0$ w.r.t. the MPL system in (2.2)*. The down-pointing arrow in $\mathcal{X}_{-1}$ indicates a half-line: that set can be expressed as a union of two DBM.*

Corollary 3.1 The image and the inverse image of a union of finitely many DBM w.r.t. an MPL system or an MiPL system or switching MiPL system are also a union of finitely many DBM. □

Computing the image and the inverse image of a union of q DBM w.r.t. an MPL system or an MiPL system, characterized by matrix A can be done by computing the image and the inverse image of each DBM w.r.t. the matrix. Thus the complexity of both cases is $\mathcal{O}(q(n+p)^3 q_A)$. A similar approach can be used to determine the image and inverse image of a union of finitely many DBM w.r.t. a switching MiPL system.

Remark 3.1 Some of the above results can be generalized to DBM in $\mathbb{R}_\varepsilon^p$ and to matrices that are not row-finite by using similar proof techniques. One of them is the following: the image of a DBM in $\mathbb{R}_\varepsilon^p$ w.r.t. a matrix in $\mathbb{R}_\varepsilon^{n\times p}$ is a union of finitely many DBM in $\mathbb{R}_\varepsilon^n$. □

We have mentioned an alternative approach to reachability analysis of MPL systems based on operations over max-plus polyhedra, and emphasized the limitations of such an approach. A max-plus polyhedron is defined as the max-plus Minkowski sum of a max-plus cone and a finitely-generated max-plus convex set [15, Sec. 2.2]. Max-plus cones in $\mathbb{R}_\varepsilon^n$, a special case of max-plus polyhedra, are a max-plus linear combination of finitely many vectors in $\mathbb{R}_\varepsilon^n$. Equivalently, a max-plus cone can be represented as the image of $\mathbb{R}_\varepsilon^p$ w.r.t. a matrix in $\mathbb{R}_\varepsilon^{n\times p}$. Based on Remark 3.1 and by using the homogeneous coordinates representation [15, Sec. 2.2], one can show the following proposition.

Proposition 3.2 Every max-plus cone and max-plus polyhedron can be expressed as a union of finitely many DBM. □

Proof Recall that max-plus cones are a max-plus linear combination of finitely many vectors. A max-plus cone can be represented as the image of max-plus linear map governed by a max-plus matrix made up of those vectors. Remark 3.1 implies each max-plus cone can be expressed as a union of finitely many DBM. This result can be generalized to max-plus polyhedra since each max-plus polyhedron can be expressed as a max-plus cone by using homogeneous coordinates representation [15, Sec. 2.2]. □

As mentioned in [91, p. 1785], each max-plus linear combination of finitely many two-dimensional vectors is a stripe, which is a particular kind of DBM. For didactical purposes, we present the following proposition together with its complete proof.

Proposition 3.3 Every two-dimensional max-plus cone can be expressed as a DBM. □

Proof The proof consists of two steps. In the first step, we show that each max-plus cone can be formulated as a max-plus linear combination of two vectors. Then we show that any max-plus linear combination of two vectors can be expressed as a DBM.

We assume the max-plus cone is given by $\alpha_1 \otimes \mathbf{x}^1 \oplus \cdots \oplus \alpha_q \otimes \mathbf{x}^q$. We show that the preceding max-plus cone can be expressed as the following max-plus linear combination $\alpha_{max} \otimes \mathbf{x}^{max} \oplus \alpha_{min} \otimes \mathbf{x}^{min}$, where $\mathbf{x}^{max}, \mathbf{x}^{min} \in \{\mathbf{x}^1, \ldots, \mathbf{x}^q\}$ such that $x_1^{max} - x_2^{max} \geq x_1^k - x_2^k$ and $x_1^{min} - x_2^{min} \leq x_1^k - x_2^k$ for all $k = 1, \ldots, q$. This can be done by showing each $\mathbf{x}$ that satisfies $x_1^{min} - x_2^{min} \leq x_1 - x_2 \leq x_1^{max} - x_2^{max}$ can be written as a max-plus linear combination of $\mathbf{x}^{max}$ and $\mathbf{x}^{min}$, i.e. $\mathbf{x} = \alpha_{max} \otimes \mathbf{x}^{max} \oplus \alpha_{min} \otimes \mathbf{x}^{min}$ (cf. Fig. 3.3). By virtue of simple algebraic manipulations, one can show that $\alpha_{max} = x_1 - x_1^{max}$ and $\alpha_{min} = x_2 - x_2^{min}$ are the solution. This implies the generators $\mathbf{x}^1, \ldots, \mathbf{x}^q$ can be expressed as a max-plus linear combination of $\mathbf{x}^{max}$ and $\mathbf{x}^{min}$.

In the second step, we show that the following max-plus cone $\alpha_{max} \otimes \mathbf{x}^{max} \oplus \alpha_{min} \otimes \mathbf{x}^{min}$ can be expressed as the following DBM $\{\mathbf{x} : x_1^{min} - x_2^{min} \leq x_1 - x_2 \leq x_1^{max} - x_2^{max}\}$ where we assume $x_1^{max} - x_2^{max} \geq x_1^{min} - x_2^{min}$. In the first part, we show that the max-plus cone is a subset of the DBM, then in the second part, we show that the DBM is a subset of the max-plus cone.

Let us prove that the max-plus cone is a subset of the DBM. If $\mathbf{x}$ is a vector in the max-plus cone, then there exist $\alpha_{max}, \alpha_{min}$ such that $\mathbf{x} = \alpha_{max} \otimes \mathbf{x}^{max} \oplus \alpha_{min} \otimes \mathbf{x}^{min}$. We will show that $\mathbf{x}$ is in the DBM by proving that $x_1^{min} - x_2^{min} \leq x_1 - x_2 \leq x_1^{max} - x_2^{max}$. We consider four possible cases. In each case, we compute the lower and upper bounds of $x_1 - x_2$ by using the corresponding assumptions.

- We assume $\alpha_{max} + x_1^{max} \geq \alpha_{min} + x_1^{min}$ and $\alpha_{max} + x_2^{max} \geq \alpha_{min} + x_2^{min}$. In this case $x_1 - x_2 = x_1^{max} - x_2^{max}$.
- We assume $\alpha_{max} + x_1^{max} \geq \alpha_{min} + x_1^{min}$ and $\alpha_{max} + x_2^{max} \leq \alpha_{min} + x_2^{min}$. Applying the inequalities results in $x_1^{min} - x_2^{min} \leq x_1 - x_2 \leq x_1^{max} - x_2^{max}$.
- We assume $\alpha_{max} + x_1^{max} \leq \alpha_{min} + x_1^{min}$ and $\alpha_{max} + x_2^{max} \geq \alpha_{min} + x_2^{min}$. Combining both inequalities and the previous assumption, i.e. $x_1^{max} - x_2^{max} \geq x_1^{min} - x_2^{min}$, yields $x_1^{max} - x_2^{max} = x_1^{min} - x_2^{min}$. Applying the inequalities results in $x_1^{max} - x_2^{max} \leq x_1 - x_2 \leq x_1^{min} - x_2^{min}$.
- We assume $\alpha_{max} + x_1^{max} \leq \alpha_{min} + x_1^{min}$ and $\alpha_{max} + x_2^{max} \leq \alpha_{min} + x_2^{min}$. In this case $x_1 - x_2 = x_1^{min} - x_2^{min}$.

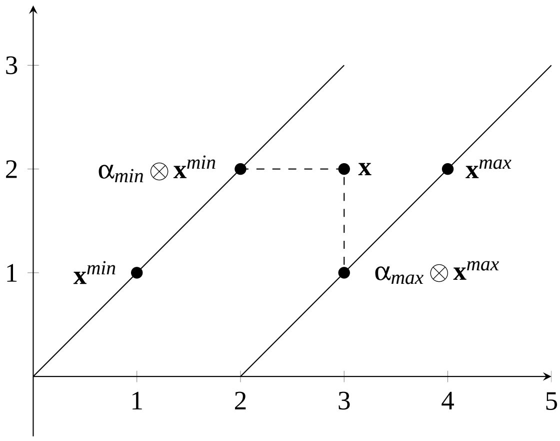

Fig. 3.3: Graphical illustration of computing the constants α_{max} *and* α_{min} *corresponding to a vector* $\mathbf{x}$.

Finally the arguments in the first step show that the DBM is a subset of the max-plus cone. □

As a result, the reachability analysis based on DBM is more general than the one based on max-plus polyhedra.

3.2.2 Transition Systems

This section introduces transition systems, a (by now) standard class of models to represent hardware and software systems [23, Sec. 2.1].

Definition 3.3 (Transition System [23, Def. 2.1]) A transition system TS is characterized by a sextuple $(S, Act, \longrightarrow, I, AP, L)$ where

- S is a set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- AP[2] is a set of atomic propositions, and
- $L : S \rightarrow 2^{AP}$ is a labeling function.

TS is called finite if S, Act, and AP are finite. □

---

[2] The notation AP does not represent the multiplication of matrix A and matrix P, unless stated explicitly.

For convenience, we write $s \xrightarrow{\gamma} s'$ instead of $(s,\gamma,s') \in \longrightarrow$. The behavior of a transition system can be described as follows. The transition system starts in some initial state $s_0 \in I$ and evolves according to the transition relation $\longrightarrow$. If a state has more than one outgoing transition, the "next" transition is chosen in a purely nondeterministic fashion. Recall that 2^{AP} denotes the power set of AP. The labeling function relates each state to set of atomic propositions that are satisfied by the state.

Definition 3.4 (Direct Predecessors and Direct Successors [23, Def. 2.3]) Let $TS = (S, Act, \longrightarrow, I, AP, L)$ be a transition system. For $s \in S$ and $\gamma \in Act$, the set of direct γ-successors of s is defined as

$$Post(s,\gamma) = \left\{ s' \in S : s \xrightarrow{\gamma} s' \right\}, \qquad Post(s) = \bigcup_{\gamma \in Act} Post(s,\gamma).$$

The set of direct γ-predecessors of s is defined by

$$Pre(s,\gamma) = \left\{ s' \in S : s' \xrightarrow{\gamma} s \right\}, \qquad Pre(s) = \bigcup_{\gamma \in Act} Pre(s,\gamma). \qquad \square$$

The notations for the sets of direct successors and predecessors are expanded to subsets of S in the obvious way (i.e. pointwise extension): for $C \subseteq S$, let

$$Post(C,\gamma) = \bigcup_{s \in C} Post(s,\gamma), \qquad Post(C) = \bigcup_{s \in C} Post(s).$$

The notations $Pre(C,\gamma)$ and $Pre(C)$ are defined in an analogous way:

$$Pre(C,\gamma) = \bigcup_{s \in C} Pre(s,\gamma), \qquad Pre(C) = \bigcup_{s \in C} Pre(s).$$

A transition system $TS = (S, Act, \longrightarrow, I, AP, L)$ is called deterministic if $|I| \leq 1$ and $|Post(s,\gamma)| \leq 1$ for all states s and actions γ [23, Def. 2.5]. A path of transition system TS is a sequence of states starting from some initial state; evolves according to the transition relation; and cannot be prolonged, i.e. either it is infinite or it is finite but ends in a terminal state [23, Defs. 2.4 and 3.6]. The set of all paths in transition system TS is denoted by $Paths(TS)$. A trace of a path is defined as the finite or infinite word over the alphabet 2^{AP} obtained by applying the labeling function to the path. The set of traces of transition system TS is defined as the trace of all paths in TS, i.e. $Traces(TS) = trace(Paths(TS))$ [23, p. 98].

3.2.3 Linear Temporal Logic

This section introduces (propositional) Linear Temporal Logic (LTL), a logical formalism that is suited for specifying properties [23, Ch. 5]. The syntax and semantics of LTL will be discussed.

LTL formulae are recursively defined over a set of atomic propositions, by Boolean operators, and temporal operators. More formally, the syntax of LTL formulae is defined as follows:

Definition 3.5 (Syntax of Linear Temporal Logic [23, Def. 5.1]) LTL formulae over the set AP of atomic proposition are formed according to the following grammar:

$$\varphi ::= true \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

where $a \in AP$. □

Boolean operators are $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction), whereas temporal operators are $\bigcirc$ (next), U (until), $\Box$ (always), and $\Diamond$ (eventually). The until operator allows to derive the temporal modalities $\Diamond$ and $\Box$ [23, p. 232]. The $\bigcirc$-modality is a unary prefix operator and requires a single LTL formula as argument. Formula $\bigcirc\varphi$ holds at the current moment, if φ holds in the next "step". The U-modality is a binary infix operator and requires two LTL formulae as argument. Formula $\varphi_1 \mathsf{U} \varphi_2$ holds at the current moment, if there is some future moment for which φ_2 holds and φ_1 holds at all moments until that future moment. The $\Diamond$-modality and $\Box$-modality is a unary prefix operator and requires a single LTL formula as argument. The formula $\Diamond\varphi$ is satisfied if φ will be true eventually in the future, whereas the formula $\Box\varphi$ is satisfied if φ holds from now on forever. The intuitive meaning of temporal modalities for a simple case is described in Fig. 3.4.

Safety properties are a class of LTL formulae and often characterized as "nothing bad should happen" [23, p. 107]. As an example consider a specification of a traffic light with the usual three phases "red", "green", and "yellow". The requirement that each red phase should be immediately preceded by a yellow phase is a safety property. Invariant property is a particular kind of safety properties that is given by a condition for the states. As an example consider the following specification of a traffic light. The requirement that each red and green phases should not occur simultaneously is an invariant property.

LTL formulae stand for properties of paths (or in fact their trace). This means that a path can either fulfill an LTL-formula or not. An infinite path satisfies an LTL formula φ if the trace of the path satisfies φ [23, p. 236]. Recall that the trace of an infinite path is an infinite word over the alphabet 2^{AP}. A transition system satisfies an LTL formula if all paths of the transition system satisfy the LTL formula [23, p. 237].

3.2.4 Abstractions

Abstraction is a fundamental concept that permits the analysis of large [23, Ex. 7.53] or even infinite [23, Ex. 7.54] transition systems. An abstraction is identified by a set of abstract states $\hat{S}$; an abstraction function f, that associates to each (concrete) state s of the transition system TS the abstract state $f(s)$ which represents it; and a set AP of atomic propositions that label the concrete and abstract states. Abstractions differ in the choice of the set $\hat{S}$ of abstract states, the abstraction function f, and the relevant propositions AP.

Typically an abstract transition system simulates the concrete transition system. Simulation relations are used as a basis for abstraction techniques where the rough idea is to replace the model to be verified by a smaller abstract model and to verify the latter instead of the original one. Simulation relations are preorders on the state space requiring that whenever s' simulates s, state s' can mimic all stepwise behavior of s, but the reverse is not guaranteed. The formal definition of the simulation order is given below.

Definition 3.6 (Simulation Order [23, Def. 7.47]) Let $TS_i = (S_i, Act_i, \longrightarrow_i, I_i, AP, L_i)$, $i =$

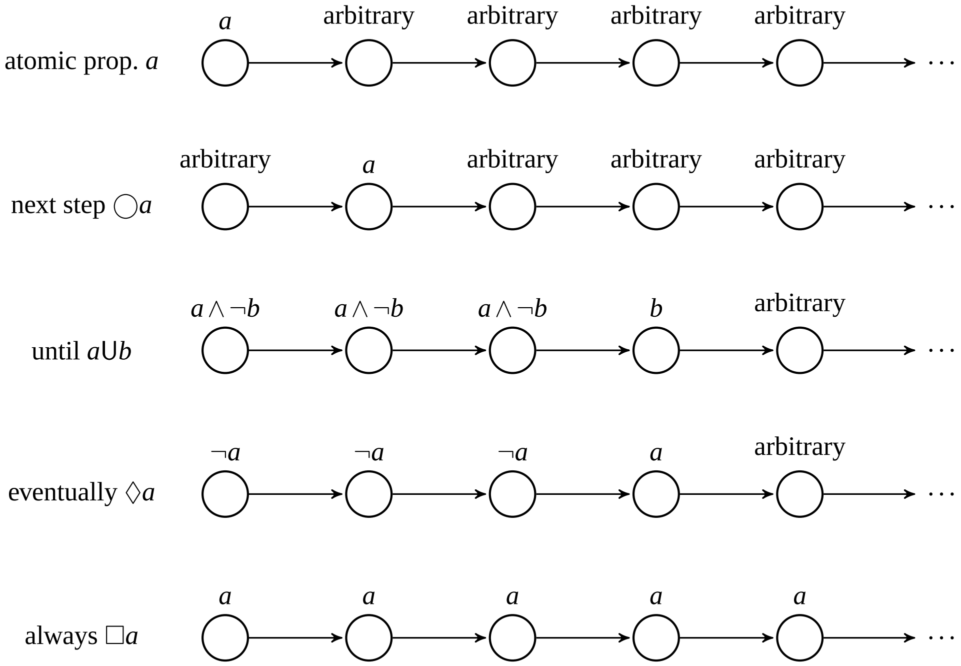

Fig. 3.4: Intuitive semantics of temporal modalities.

$1,2$, be transition systems over AP. A simulation for (TS_1, TS_2) is a binary relation $\mathcal{R} \subseteq S_1 \times S_2$ such that

1. for each $s_1 \in I_1$ there exists $s_2 \in I_2$ such that $(s_1, s_2) \in \mathcal{R}$
2. for all $(s_1, s_2) \in \mathcal{R}$ it holds:
   (a) $L_1(s_1) = L_2(s_2)$
   (b) if $s_1' \in Post(s_1)$ then there exists $s_2' \in Post(s_2)$ with $(s_1', s_2') \in \mathcal{R}$.

Transition system TS_1 is simulated by TS_2 (or, equivalently, TS_2 simulates TS_1) if there exists a simulation $\mathcal{R}$ for (TS_1, TS_2). □

Intuitively speaking, TS_1 is simulated by TS_2 means for every path in TS_1 there exists a path in TS_2 such that their traces coincide. Recall that a path is a sequence of states and trace is a sequence of subsets of AP.

We briefly outline the essential ideas of abstractions that are obtained by aggregating disjoint sets of concrete states into single abstract states. Abstraction functions map concrete states onto abstract ones, such that abstract states are associated with equally labeled concrete states only [23, Def. 7.50].

The abstract transition system TS_f originates from TS by identifying all states that are represented by the same abstract state under abstraction function f. An abstract state is initial whenever it represents an initial concrete state. Similarly, there is a transition from abstract state $f(s)$ to state $f(s')$ if there is a transition from s to s' [23, Def. 7.51].

Proposition 3.4 ([23, Lem. 7.52]) Let $TS = (S, Act, \longrightarrow, I, AP, L)$ be a (concrete) transition system, $\hat{S}$ a set of (abstract) states, and $f : S \to \hat{S}$ an abstraction function. Then TS_f simulates TS. □

Proposition 3.5 ([23, Cor. 7.68 and Th. 7.70]) Let TS_2 simulates TS_1, assume TS_1 does not have terminal states, let φ be a linear-time property. If TS_2 satisfies φ, then TS_1 also satisfies φ. □

Informally speaking, linear-time property specifies the admissible (or desired) behavior of the system under consideration [23, p. 100]. The result also applies to LTL formulae, since each LTL formula is a linear-time property [23, Defs. 3.10 and 5.6]. In general the reverse of the preceding proposition is not true. If TS_2 does not satisfy φ, we cannot deduce that TS_1 does not satisfy φ since the trace of paths that violate φ might be behaviors that TS_1 cannot perform at all.

There is a close connection between abstraction functions and partitions. For the abstraction function $f : S \to \hat{S}$, notice that $\bigcup_{\hat{s} \in \hat{S}} \{s : f(s) = \hat{s}\}$ is a partition of S. Recall that there is a connection between equivalence relations and partitions [23, Rem. 7.30]. Let us construct an abstraction function f and a set of (abstract) states $\hat{S}$ from a given equivalence relation. The set of (abstract) states $\hat{S}$ is defined as the collection of equivalence classes. The abstraction function f maps each (concrete) state to the unique equivalence class containing the (concrete) state.

The bisimulation-quotienting algorithms [23, Sec. 7.3] can be used to obtain a bisimulation quotient transition system if the concrete transition system is finite. In this case, the initial partition is defined as the partition induced by the abstraction function (see previous paragraph), which is finer than the *AP* partition [23, Def. 7.31]. However if the concrete transition system is infinite, the termination of the algorithms is not guaranteed [23, p. 477].

Definition 3.7 (Bisimulation Equivalence [23, Def. 7.1]) For $i = 1, 2$ let TS_i be transition systems over *AP*, i.e. $TS_i = (S_i, Act_i, \longrightarrow_i, I_i, AP, L_i)$. A bisimulation for (TS_1, TS_2) is a binary relation $\mathcal{R} \subseteq S_1 \times S_2$ such that

1. for each $s_1 \in I_1$ there exists $s_2 \in I_2$ such that $(s_1, s_2) \in \mathcal{R}$ and for each $s_2 \in I_2$ there exists $s_1 \in I_1$ such that $(s_1, s_2) \in \mathcal{R}$
2. for all $(s_1, s_2) \in \mathcal{R}$ it holds that
    (a) $L_1(s_1) = L_2(s_2)$
    (b) if $s_1' \in Post(s_1)$ then there exists $s_2' \in Post(s_2)$ with $(s_1', s_2') \in \mathcal{R}$
    (c) if $s_2' \in Post(s_2)$ then there exists $s_1' \in Post(s_1)$ with $(s_1', s_2') \in \mathcal{R}$.

Transition systems TS_1 and TS_2 are bisimulation-equivalent (bisimilar, for short) if there exists a bisimulation $\mathcal{R}$ for (TS_1, TS_2). □

Bisimulation equivalence denotes the possibility of mutual, stepwise simulation. Bisimulation equivalence preserves all formulae that can be formulated in CTL*, which is strictly more expressive than LTL [23, p. 469]. This result allows performing model checking on the bisimulation quotient transition system while preserving both affirmative and negative outcomes of the model checking.

As a side note, here the notion of simulation and bisimulation is defined over the state labels and does not consider the action labels. These notions can be also defined on action labels rather than state labels. This connection is discussed in [23, Sec. 7.1.2].

3.3 Autonomous Max-Plus-Linear Systems

Recall that the idea of abstraction is to replace a model to be verified by a smaller abstract model and to verify the latter instead of the original one, where both models are expressed as transition systems. Let us introduce a transition system related to an autonomous MPL system.

Definition 3.8 (Transition Systems Associated with Autonomous MPL Systems) Consider an autonomous MPL system (2.1) with X_0 as the set of initial conditions and a set of atomic propositions AP together with the corresponding labeling function L. The associated transition system TS is a tuple $(S, Act, \longrightarrow, I, AP, L)$ where

- set of states S is $\mathbb{R}^n$,
- set of actions Act is $\{\tau\}$,
- there exists a transition relation $\mathbf{x} \xrightarrow{\tau} \mathbf{x}'$ if $\mathbf{x}' = A \otimes \mathbf{x}$, and
- set of initial states I is X_0.

In cases where action names are irrelevant, we use a special symbol τ. □

In this work, we assume the set of states satisfying each atomic proposition is a DBM, i.e. for each $a \in AP$, the set of states $\{\mathbf{x} : a \in L(\mathbf{x})\}$ is a DBM. A transition system can be restricted to a set of states, as defined next.

Definition 3.9 (Restriction of Transition Systems) Consider a transition system $TS = (S, Act, \longrightarrow, I, AP, L)$. The restriction of TS to a nonempty set of states $S' \subseteq S$ is defined as $TS' = (S', Act', \longrightarrow', I', AP', L')$ with

- set of actions $Act' = Act$,
- transition relation $\longrightarrow' = \longrightarrow \cap (S' \times Act' \times S')$,
- set of initial states $I' = I \cap S'$,
- set of atomic propositions $AP' = AP$, and
- labeling function $L' = L|_{S'}$.

The notation $L|_{S'} : S' \rightarrow AP'$ describes a restriction of function L to set S' defined by $L|_{S'}(s') = L(s')$ for every $s' \in S'$. □

3.3.1 States: Partitioning Procedure

We construct a partition of S and then the abstraction function f maps each state in the same block to a unique abstract state. A partition of a set is a division of the set as a union of non-overlapping and non-empty subsets, called “blocks” [23, Def. 7.29]. More precisely we develop an approach to construct a partition Π_0 of the set of states S, where Π_0 is an *AP* partition [23, Def. 7.31], each block is a DBM, and the dynamics in each block are affine. The approach is as follows. We first determine an *AP* partition of S, denoted by Π_{AP}, where each block is a DBM. Then we determine a partition Π_{AD} of S where each block is a DBM and the dynamics in each block are affine.[3] Finally the partition Π_0 is defined as the refinement of Π_{AP} and Π_{AD}, i.e. $\mathcal{R}_{\Pi_0} = \mathcal{R}_{\Pi_{AP}} \cap \mathcal{R}_{\Pi_{AD}}$ [23, Rem. 7.30].

The *AP* Partition

We discuss a procedure to generate an *AP* partition of S where each block is a DBM. Algorithm 29 in [23] cannot be used because the algorithm requires that the cardinality of S is finite. We propose the following approach. First we compute the coarsest *AP* partition, i.e. for each $\mathbf{a} \in 2^{AP}$ we define a block as the inverse image of $\mathbf{a}$ w.r.t. the labeling function L, i.e. $L^{-1}(\mathbf{a}) = \{\mathbf{x} : L(\mathbf{x}) = \mathbf{a}\} = \cap_{a\in\mathbf{a}}\{\mathbf{x} : a \in L(\mathbf{x})\} \setminus \cup_{a\in 2^{AP}\setminus\mathbf{a}}\{\mathbf{x} : a \in L(\mathbf{x})\}$. Notice that in general each block is a union of finitely many DBM, since the set difference between two DBM is a union of finitely many DBM. Finally the coarsest *AP* partition is refined such that each block is a DBM.

Example Suppose that $AP = \{a\}$ and the set of states satisfying a is the following stripe $\{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 < 3\}$. The coarsest *AP* partition contains two blocks, i.e. $\{\mathbf{x} : 0 \leq x_1 - x_2 < 3\}$ and $\{\mathbf{x} : x_1 - x_2 < 0\} \cup \{\mathbf{x} : x_1 - x_2 \geq 3\}$. Since the latter block is a union of two DBM, it is refined into two blocks, i.e. $\{\mathbf{x} : x_1 - x_2 < 0\}$ and $\{\mathbf{x} : x_1 - x_2 \geq 3\}$. The resulting *AP* partition contains three blocks, i.e. $B_1 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 < 0\}$, $B_2 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 < 3\}$, and $B_3 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 3\}$ as shown in Fig. 3.5. The procedure to construct an *AP* partition has not been implemented in VeriSiMPL version 1.4. □

The *AD* Partition

We discuss two different approaches to construct a partition of S where each block is a DBM and the dynamics in each block are affine. The first approach generates the partition directly from the state matrix, whereas the second approach constructs the partition from the regions of the PWA system generated by the state matrix. By using a tailored refinement procedure, the second approach leads to a partition that is coarser than the one generated by the first approach (cf. Proposition 3.7).

The First Approach We determine a partition of the state space based on the value of $A(i, j) + x_j$, similar to Section 2.2.4. Given an autonomous MPL system characterized by a row-finite max-plus matrix $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ and a generic $\mathbf{x} \in \mathbb{R}^n$, for notational purposes we define $W_{\mathbf{x}}(i, j) = A(i, j) + x_j - [A \otimes \mathbf{x}]_i$. Notice that each element of $W_{\mathbf{x}}$ is nonpositive,

---

[3] *AD* stands for “affine dynamics”; thus, *AD* does not represent the multiplication of matrix A and matrix D, unless stated explicitly.

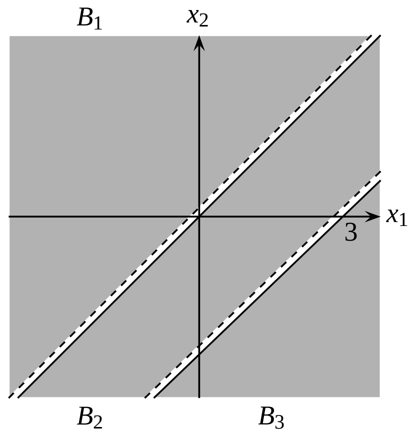

Fig. 3.5: AP partition of $\mathbb{R}^2$ *where all blocks are DBM.*

depends (given a matrix A) only on $\mathbf{x}$, and that there exists a nonempty set of usual zero (0) elements in each of its rows. Each region generated by this approach is characterized by a parameter set $\mathbf{f} = (f_1, \ldots, f_n) \in (2^{\{1,\ldots,n\}} \setminus \{\emptyset\})^n$, where $f_i = \{j : W_{\mathbf{x}}(i,j) = 0\} = \{j : [A \otimes \mathbf{x}]_i = A(i,j) + x_j\}$ for $i = 1, \ldots, n$. More precisely the region characterized by $\mathbf{f}$, denoted by $R_{\mathbf{f}}$, is defined as the set of points $\mathbf{x} \in \mathbb{R}^n$ verifying the condition for matrix $W_{\mathbf{x}}$, i.e. $R_{\mathbf{f}} = \{\mathbf{x} \in \mathbb{R}^n : W_{\mathbf{x}}(i,j) = 0 \text{ iff } j \in f_i \text{ for each } i = 1, \ldots, n\}$.

In order to design a procedure for the proposed approach, we need to characterize each point $\mathbf{x} \in R_{\mathbf{f}}$ based on the value of $A(i,j) + x_j$. For each $i = 1, \ldots, n$; $j \in f_i$; and $j' = 1, \ldots, n$; the following property holds: if $j' \in f_i$, then $A(i,j) + x_j = A(i,j') + x_{j'}$; if $j' \notin f_i$, then $A(i,j) + x_j > A(i,j') + x_{j'}$. Thus a constructive definition of $R_{\mathbf{f}} \subseteq \mathbb{R}^n$ is as follows:

$$R_{\mathbf{f}} = \bigcap_{i=1}^{n} \bigcap_{j \in f_i} \bigcap_{j'=1}^{n} \begin{cases} \{\mathbf{x} \in \mathbb{R}^n : A(i,j) + x_j = A(i,j') + x_{j'}\}, & \text{if } j' \in f_i, \\ \{\mathbf{x} \in \mathbb{R}^n : A(i,j) + x_j > A(i,j') + x_{j'}\}, & \text{if } j' \notin f_i. \end{cases} \tag{3.1}$$

Algorithm 3.1 Generation of a partition from a row-finite state matrix
Input: $A \in \mathbb{R}_{\varepsilon}^{n \times n}$, a row-finite max-plus matrix
Output: Π_{AD}, a partition of S

```
initialize $\Pi_{AD}$ with the empty set
for all $\mathbf{f} \in (2^{\{1,\ldots,n\}} \setminus \{\emptyset\})^n$ do
  generate region $R_{\mathbf{f}}$ according to (3.1)
  if $R_{\mathbf{f}}$ is not empty then
    save the region, i.e. $\Pi_{AD} := \Pi_{AD} \cup \{R_{\mathbf{f}}\}$
  end if
end for
```

The worst-case complexity of Algorithm 3.1 is $O(n^3(2^n - 1)^n)$ [8, p. 3044]. The crucial observation that allows for an improvement of the complexity is that it is not necessary to iterate over all possible characterizations of $\mathbf{f}$ as in Algorithm 3.1. Instead we can apply the backtracking technique, similar to the one used for Algorithm 2.1. In the backtracking

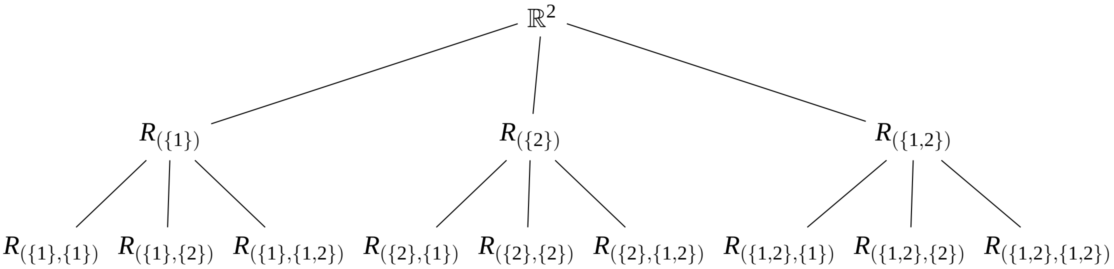

Fig. 3.6: Potential search tree for a 2-dimensional MPL system.

approach, the partial characterizations are $(f_1,\ldots,f_k)$ for $k = 1,\ldots,n$ and the corresponding region is

$$R_{(f_1,\ldots,f_k)} = \bigcap_{i=1}^{k} \bigcap_{j \in f_i} \bigcap_{j'=1}^{n} \begin{cases} \{\mathbf{x} \in \mathbb{R}^n : A(i,j) + x_j = A(i,j') + x_{j'}\}, & \text{if } j' \in f_i, \\ \{\mathbf{x} \in \mathbb{R}^n : A(i,j) + x_j > A(i,j') + x_{j'}\}, & \text{if } j' \notin f_i. \end{cases}$$

Notice that if the region associated with a certain partial characterization $(f_1,\ldots,f_k)$ is empty, then the regions of the corresponding characterizations $(f_1,\ldots,f_n)$ are also empty, for all $f_{k+1},\ldots,f_n$. The set of all characterizations can be represented as a potential search tree. For a 2-dimensional MPL system, the potential search tree is given in Fig. 3.6.

Example Consider the autonomous MPL system in (2.2). The regions generated by the scheme in Algorithm 3.1 are $R_{(\{1\},\{1\})} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 > 3\}$, $R_{(\{1,2\},\{1\})} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 = 3\}$, $R_{(\{2\},\{1\})} = \{\mathbf{x} \in \mathbb{R}^2 : 0 < x_1 - x_2 < 3\}$, $R_{(\{2\},\{1,2\})} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 = 0\}$, and $R_{(\{2\},\{2\})} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 < 0\}$. The regions are shown in Fig. 3.7 (left).

In VeriSiMPL version 1.4, the procedure to generate an *AD* partition by using the first approach (cf. Algorithm 3.1) has been implemented in `mpl2pwa_part` as a function. This function requires the state matrix (`Ampl`). This function generates a collection of finitely many DBM (`D`,`sysD`) and the corresponding affine dynamics (`A`,`B`). Variable `sysD` relates each DBM to the affine dynamics that are active in the DBM. The following MATLAB script re-calculates the numerical example in the preceding paragraph:

```
>> Ampl = [2 5;3 3]
>> [A,B,D,sysD] = mpl2pwa_part(Ampl)
```

$\square$

The Second Approach A partition of S can be also obtained from the regions of the PWA system generated by the state matrix. The procedure to obtain a partition is not unique: with focus on memory usage, we propose one that leads to a partition that is coarser than the one generated by Algorithm 3.1 (cf. Proposition 3.7). Let us start with the following concept.

Definition 3.10 (Adjacent Regions) Let $R_{\mathbf{g}}$ and $R_{\mathbf{g}'}$ be regions generated by an n-dimensional state space matrix. We say that they are adjacent ($R_{\mathbf{g}} > R_{\mathbf{g}'}$) if there exists a single $i \in \{1,\ldots,n\}$ such that $g_i > g'_i$ and $g_j = g'_j$ for each $j \neq i$. $\square$

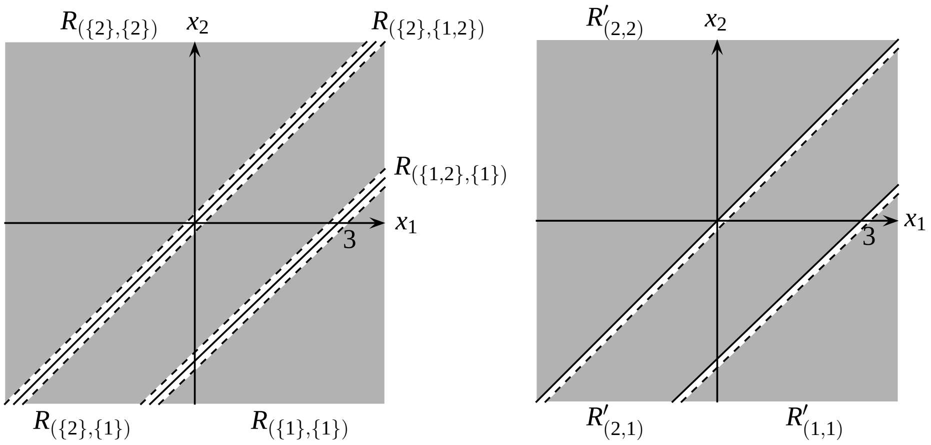

Fig. 3.7: The left and right plots are AD partition of the autonomous MPL system in (2.2) *obtained using the first and second approaches, respectively.*

Given a collection of regions generated by the state space matrix using Algorithm 2.1, the procedure (cf. Algorithm 3.2) works as follows. For each pair of adjacent regions, their intersection is combined to the region with higher index.

Algorithm 3.2 Generation of a partition from regions of PWA system

```
Input: $A \in \mathbb{R}_{\varepsilon}^{n \times n}$, a row-finite max-plus matrix
Output: $\Pi_{AD}$, a partition of $S$
initialize $\Pi_{AD}$ with the regions of the PWA system (cf. Algorithm 2.1)
for all $R_{\mathbf{g}}, R_{\mathbf{g}'} \in \Pi_{AD}$ do
  if $R_{\mathbf{g}} > R_{\mathbf{g}'}$ then
    the intersection is removed from the region with lower index,
    i.e. $R_{\mathbf{g}'} := R_{\mathbf{g}'} \setminus R_{\mathbf{g}}$
  end if
end for
```

It has been shown that this procedure generates a partition of S [8, p. 3045]. Proving that the procedure does not increase the number of regions equates to showing that the set difference of two adjacent regions is a DBM (cf. Proposition 3.6). The worst-case complexity of Algorithm 3.2 is $\mathcal{O}(n^{2n+1})$ [8, p. 3045].

Proposition 3.6 ([8, Prop. 3]) If $R_{\mathbf{g}} > R_{\mathbf{g}'}$, then $R_{\mathbf{g}'} \setminus R_{\mathbf{g}} = R_{\mathbf{g}'} \cap \{\mathbf{x} \in \mathbb{R}^n : A(i, g_i') + x_{g_i'} > A(i, g_i) + x_{g_i}\}$, which is a DBM. □

Example Consider the autonomous MPL system in (2.2). The regions generated by the scheme in Algorithm 3.2 are $R'_{(1,1)} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 > 3\}$, $R'_{(2,1)} = \{\mathbf{x} \in \mathbb{R}^2 : 0 < x_1 - x_2 \leq 3\}$, and $R'_{(2,2)} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \leq 0\}$. Notice that $R'_{(2,2)} = R_{(2,2)}$. The regions are shown in Fig. 3.7 (right).

In VeriSiMPL version 1.4, the second approach to generate an *AD* partition has been implemented in two functions `mpl2pwa` and `mpl2pwa_refine`. Recall that the function `mpl2pwa` generates a PWA system from an autonomous MPL system (cf. Section 2.2.4). The function `mpl2pwa_refine` refines the PWA regions to obtain a partition. This function requires a state matrix (`Ampl`) and the PWA system generated by the state matrix (`A`,`B`,`D`). This function returns a PWA system (`A`,`B`,`D`) where the PWA regions (`D`) are a partition of $\mathbb{R}^n$. The following MATLAB script determines the *AD* partition of the autonomous MPL system in (2.2):

```
>> Ampl = [2 5;3 3]
>> [A,B,D] = mpl2pwa(Ampl)
>> [A,B,D] = mpl2pwa_refine(Ampl,A,B,D)
```

Let us determine the partition Π_0 of the MPL system (2.2). If we use the Π_{AD} generated by the first approach, Π_0 coincides with the Π_{AD} since Π_{AD} is finer than Π_{AP}. If we use the Π_{AD} generated by the second approach, one can show that Π_0 coincides with the Π_{AD} generated by the first approach.

Finally we define the set of abstract states $\hat{S}$, the abstraction function f, and the labeling function of the abstract transition system $L_f : \hat{S} \rightarrow 2^{AP}$. Since Π_0 contains 5 blocks, $\hat{S} = \{\hat{s}_1, \hat{s}_2, \hat{s}_3, \hat{s}_4, \hat{s}_5\}$. The abstraction function f and the labeling function L_f are defined as follows

$$f(\mathbf{x}) = \begin{cases} \hat{s}_1, & \text{if } x_1 - x_2 < 0, \\ \hat{s}_2, & \text{if } x_1 - x_2 = 0, \\ \hat{s}_3, & \text{if } x_1 - x_2 > 3, \\ \hat{s}_4, & \text{if } x_1 - x_2 = 3, \\ \hat{s}_5, & \text{if } 0 < x_1 - x_2 < 3, \end{cases} \qquad L_f(\hat{s}_i) = \begin{cases} \{a\}, & \text{if } i = 2,5, \\ \emptyset, & \text{if } i = 1,3,4. \end{cases} \qquad \square$$

The following proposition justifies that the second approach is computationally advantageous, since it generates a coarser partition.

Proposition 3.7 The partition generated by the first approach is finer than the one generated by the second approach. □

Proof We will prove that each block generated by the second approach is a union of blocks generated by the first approach [23, p. 476]. Notice that each block generated by the second approach is the set difference between a PWA region and a union of PWA regions (cf. Algorithm 3.2). Recall that each PWA region is a union of blocks generated by the first approach, i.e. $R_{\mathbf{g}} = \bigcup_{\mathbf{f} \in F(\mathbf{g})} R_{\mathbf{f}}$ where $F(\mathbf{g}) = \{\mathbf{f} : g_i \in f_i \text{ for each } i = 1, \ldots, n\}$ [8, Prop. 2]. It follows that the set difference between a PWA region and a union of PWA regions is also a union of blocks generated by the first approach, i.e. $R_{\mathbf{g}} \setminus \bigcup_{i=1}^{q} R_{\mathbf{g}'_i} = \bigcup_{\mathbf{f} \in F(\mathbf{g}) \setminus \cup_{i=1}^{q} F(\mathbf{g}'_i)} R_{\mathbf{f}}$. □

3.3.2 Transitions: One-Step Reachability

We investigate a technique to determine the transition relations of the abstract transition system, that is between two blocks of the partition induced by the abstraction function. The (concrete) states associated with an abstract state $\hat{s}$ equals to the inverse image of $\hat{s}$ w.r.t. the abstraction function f, i.e. $f^{-1}(\hat{s}) = \{s : f(s) = \hat{s}\}$. Recall that $f^{-1}(\hat{s})$ is a block (or in fact a DBM) and the dynamics in each block are affine.

If there exists a transition from an outgoing state s to an incoming state s' in the concrete transition system, i.e. $s \xrightarrow{\gamma} s'$, then there is a transition from $f(s)$ to $f(s')$ in the abstract transition system, i.e. $f(s) \xrightarrow{\gamma}_f f(s')$ [23, Def. 7.51]. Such a transition can be determined by a forward- or backward-reachability approach. According to the former, we calculate $f^{-1}(\hat{s}') \cap Post(f^{-1}(\hat{s}))$, whereas if we use the backward approach we compute $f^{-1}(\hat{s}) \cap Pre(f^{-1}(\hat{s}'))$. The nonemptiness of the resulting set characterizes the presence of a transition from $\hat{s}$ to $\hat{s}'$.

Remark The equivalent terms “image” and “direct successors” are used when the dynamical system is represented as a function and a transition relation, respectively. A similar argument holds for “inverse image” and “direct predecessors”. □

In this work we focus on the forward-reachability approach, since it is computationally more attractive than the backward one. More precisely (cf. Proposition 3.1), since both approaches leverage the affine dynamics associated with the outgoing abstract state,[4] the number of direct-successors computations in the forward-reachability approach is linear w.r.t. the number of abstract states, whereas the number of direct-predecessors computations in the backward-reachability approach is quadratic w.r.t. the number of abstract states.

With focus on the forward-reachability approach, given an abstract state $\hat{s}$ we employ the affine dynamics that are active in $f^{-1}(\hat{s})$ to compute the direct successors as

$$Post(f^{-1}(\hat{s})) = \{A \otimes \mathbf{x} : \mathbf{x} \in f^{-1}(\hat{s})\}.$$

Since $f^{-1}(\hat{s})$ is a DBM, $Post(f^{-1}(\hat{s}))$ is a union of finitely many DBM (cf. page 26). The complete approach to determine the transitions of the abstract transition system is shown in Algorithm 3.3, which incurs a worst-case complexity of $\mathcal{O}(n^3|\hat{S}|^2)$ [8, p. 3046].

Example Let us consider a set of initial conditions of the autonomous MPL system in (2.2) that coincides with the eigenspace, i.e. $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 = 1\}$. Thus the set of initial abstract states is $I_f = \{\hat{s}_5\}$ [23, Def. 7.51]. The abstract transition system is shown in Fig. 3.8.

In VeriSiMPL version 1.4, the procedure to determine the transitions of the abstract transition system has been implemented in the function `ts_trans`. The inputs are a PWA system (`A`,`B`,`D`,`sysD`) where the regions `D` are a partition of the state space. The output is an adjacency matrix that is represented by a sparse Boolean matrix in MATLAB. The entry at i-th row and j-th column is `true` if there is a transition from j to i, else it is equal to `false`. The following MATLAB script determines the transitions of the abstract transition system for the autonomous MPL system in (2.2):

```
>> Ampl = [2 5;3 3]
>> [A,B,D,sysD] = mpl2pwa_part(Ampl)
>> adj = ts_trans(A,B,D,sysD)
>> adj([1 3 4 5 2],[1 3 4 5 2])
```

---

[4] The affine dynamics associated with an abstract state $\hat{s}$ are the ones that are active in $f^{-1}(\hat{s})$.

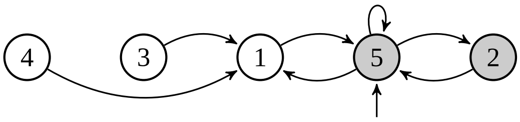

Fig. 3.8: Abstract transition system where states 1,…,5 represent respectively $\hat{s}_1, \ldots, \hat{s}_5$. The initial state is $\hat{s}_5$ indicated by an incoming arrow. The grayed states $\hat{s}_2, \hat{s}_5$ satisfy the atomic proposition a.

Recall that in this example, the initial partition Π_0 coincides with the *AD* partition generated by the first approach. Thus we use the function `mpl2pwa_part` to generate the initial partition. However the ordering of regions generated by the function `mpl2pwa_part` and the one used in the example is different. The last statement is used to re-arrange the ordering of regions generated by the function such that the ordering coincides with the one used in the example.

Suppose that we want to verify an invariant condition $\Box a$ over the abstract transition system. Recall that invariant condition $\Box a$ requires that a holds for all reachable states. According to [23, p. 107], an invariant condition is satisfied by a transition system if and only if the condition is satisfied by the reachable states. Thus the invariant condition is not satisfied by the abstract transition system. Recall that this does not imply that the invariant condition is not satisfied by the concrete transition system. □

Algorithm 3.3 Computations of the transitions of the abstract transition system via forward-reachability analysis

```
Input: $\hat{S}$, a set of abstract states
  $f : S \to \hat{S}$, an abstraction function
Output: $\longrightarrow_f \subseteq \hat{S} \times Act \times \hat{S}$, a transition relation where $Act = \{\tau\}$ (cf. Definition 3.8 and [23, Def. 7.51])

initialize $\longrightarrow_f$ with the empty set
for all $\hat{s} \in \hat{S}$ do
  compute the direct successors of $\hat{s}$, i.e. $Post(f^{-1}(\hat{s}))$
  for all $\hat{s}' \in \hat{S}$ do
    if $f^{-1}(\hat{s}') \cap Post(f^{-1}(\hat{s}))$ is not empty then
      define a transition from $\hat{s}$ to $\hat{s}'$, i.e. $\hat{s} \xrightarrow{\tau}_f \hat{s}'$
    end if
  end for
end for
```

3.3.3 Bisimulation-Quotienting Procedure

Having obtained an abstract transition system that simulates the concrete transition system, it makes sense to attempt deriving an abstract transition system that bisimulates the concrete transition system. Theorem 3.2 implies the abstract transition system bisimulates the concrete transition system if and only if there is one outgoing transition from each abstract state.

Theorem 3.2 Let TS be the concrete transition system generated by an autonomous MPL system and TS_f be the abstract transition system induced by an abstraction function $f : S \to \hat{S}$. Binary relation $\mathcal{R} = \{(f(s),s) : s \in S\}$ is a simulation for (TS_f, TS) if and only if $|Post(\hat{s})| = 1$ for all $\hat{s} \in \hat{S}$. □

Proof Suppose that $\mathcal{R} = \{(f(s),s) : s \in S\}$ is a simulation for (TS_f, TS). Condition 2(b) in Definition 3.6 means for all $(\hat{s}, s) \in \mathcal{R}$ it holds that: if $\hat{s}' \in Post(\hat{s})$, there exists $s' \in Post(s)$ with $(\hat{s}', s') \in \mathcal{R}$. If there exists an abstract state $\hat{s}$ such that $|Post(\hat{s})| > 1$, then $|Post(s)| > 1$ for all s that satisfy $f(s) = \hat{s}$, which is a contradiction. The contradiction comes from the fact that $|Post(s)| = 1$ for all s, because the value of $A \otimes \mathbf{x}$ is uniquely defined for all $\mathbf{x} \in \mathbb{R}^n$ (cf. Definition 3.8). It follows that $|Post(\hat{s})| = 1$ for all abstract states $\hat{s} \in \hat{S}$.

We assume $|Post(\hat{s})| = 1$ for all abstract states $\hat{s} \in \hat{S}$. Conditions 1 and 2(a) in Definition 3.6 are satisfied because $\mathcal{R}$ is a simulation for (TS, TS_f). Next we prove that condition 2(b) is also satisfied. Let $s \in S$, $\hat{s} = f(s)$, $Post(\hat{s}) = \{\hat{s}'\}$, and $Post(s) = \{s'\}$. Since $\mathcal{R}$ is a simulation for (TS, TS_f), then condition 2(b) and the preceding assumption imply $(\hat{s}', s') \in \mathcal{R}$. □

The procedure to generate an abstract transition system that bisimulates the concrete transition system works as follows. For each abstract state $\hat{s}$ with more than one outgoing transition, the corresponding set of states $f^{-1}(\hat{s})$ is refined according to the direct successors. Then the incoming and outgoing transitions are updated. The preceding steps are repeated until all abstract states have one outgoing transition.

Let us focus on the refinement step of the procedure. Suppose that an abstract state $\hat{s}$ has more than one outgoing transition, i.e. $|Post(\hat{s})| > 1$. The refinement step generates a partition of $f^{-1}(\hat{s})$ according to the direct successors. More precisely for each $\hat{s}' \in Post(\hat{s})$, we define a block consisting of the set of states such that the direct successor is in $f^{-1}(\hat{s}')$, i.e. $\{s \in f^{-1}(\hat{s}) : f(Post(s)) = \hat{s}'\} = f^{-1}(\hat{s}) \cap Pre(f^{-1}(\hat{s}'))$. Computationally we determine the inverse image of $f^{-1}(\hat{s}')$ w.r.t. the affine dynamics that are active in $f^{-1}(\hat{s})$, then we intersect the obtained inverse image with $f^{-1}(\hat{s})$. Notice that each block is a DBM since $f^{-1}(\hat{s}')$ is a DBM, the inverse image of a DBM w.r.t. affine dynamics is a DBM, and the intersection of two DBM is a DBM (cf. Section 3.2.1).

Example Let us apply the procedure to the abstract transition system in Fig. 3.8. The set of states $f^{-1}(\hat{s}_5)$ is partitioned into the following three blocks $\{\mathbf{x} : 0 < x_1 - x_2 < 2\}$, $\{\mathbf{x} : x_1 - x_2 = 2\}$, and $\{\mathbf{x} : 2 < x_1 - x_2 < 3\}$. After the refinement step, the partition Π_1 is a set of 7 blocks.

Next we characterize the abstract transition system associated with the refined partition Π_1. Since Π_1 contains 7 blocks, $\hat{S}' = \{\hat{s}'_1, \ldots, \hat{s}'_7\}$. The abstraction function f' and the labeling function $L_{f'}$ are defined as follows:

$$f'(\mathbf{x}) = \begin{cases} \hat{s}'_1, & \text{if } x_1 - x_2 < 0, \\ \hat{s}'_2, & \text{if } x_1 - x_2 = 0, \\ \hat{s}'_3, & \text{if } x_1 - x_2 > 3, \\ \hat{s}'_4, & \text{if } x_1 - x_2 = 3, \\ \hat{s}'_5, & \text{if } 2 < x_1 - x_2 < 3, \\ \hat{s}'_6, & \text{if } 0 < x_1 - x_2 < 2, \\ \hat{s}'_7, & \text{if } x_1 - x_2 = 2, \end{cases} \qquad L_{f'}(\hat{s}'_i) = \begin{cases} \{a\}, & \text{if } i = 2,5,6,7, \\ \emptyset, & \text{if } i = 1,3,4. \end{cases}$$

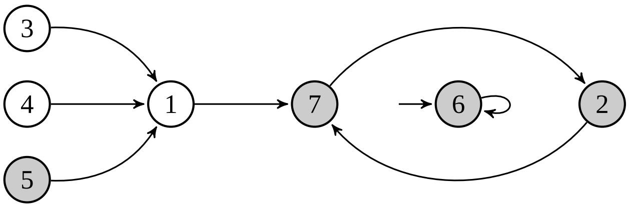

Fig. 3.9: Abstract transition system where states 1,…,7 represent respectively $\hat{s}'_1,\dots,\hat{s}'_7$. The initial state is $\hat{s}'_6$ indicated by an incoming arrow. The grayed states $\hat{s}'_2,\hat{s}'_5,\hat{s}'_6,\hat{s}'_7$ satisfy the atomic proposition a.

Recall that the set of initial states is $\{\mathbf{x} : x_1 - x_2 = 1\}$. Thus the set of initial abstract states is $I_{f'} = \{\hat{s}'_6\}$. The abstract transition system is depicted in Fig. 3.9 and it bisimulates the concrete transition system since all abstract states have one outgoing transition (cf. Theorem 3.2).

In VeriSiMPL version 1.4, the partition-refinement procedure has been implemented in the function `ts_refine`. This function requires the PWA system (`A`,`B`,`D`,`sysD`) where `D` is a partition of $\mathbb{R}^n$, the adjacency matrix (`adj`), and finally the upper bound on the number of blocks of the refined partition. This function generates a collection of regions (`D`,`sysD`) and the corresponding adjacency matrix (`adj`). Let us re-calculate the numerical example in the preceding paragraph:

```
>> Ampl = [2 5;3 3]
>> [A,B,D,sysD] = mpl2pwa_part(Ampl)
>> adj = ts_trans(A,B,D,sysD)
>> [D,sysD,adj] = ts_refine(A,B,D,sysD,adj,1000)
```

One can see that the ordering of regions generated by `ts_refine` and the one used in the example are the same.

Recall that the invariant condition $\square a$ is not satisfied by the abstract transition system before the refinement (cf. Fig. 3.8). One can check that the invariant condition is satisfied by the abstract transition system after refinement (cf. Fig. 3.9). □

Unfortunately, such a procedure in general does not necessarily terminate, especially in the presence of a cycle in the abstract transition system containing an abstract state with more than one outgoing transition. An upper bound on the number of abstract states can be used as a stopping criterion. In the remainder of this subsection, sufficient conditions for the existence of an abstract transition system that bisimulates the concrete transition system will be discussed.

Proposition 3.8 ([8, Th. 5]) Given an irreducible MPL system characterized by matrix A with cyclicity c. There exists an abstract transition system that bisimulates the concrete transition system if there exists an abstract transition system that bisimulates the concrete transition system restricted to $E(A^{\otimes c})$. □

Proposition 3.9 If the set of states satisfying each atomic proposition is a stripe, there exists an abstract transition system that bisimulates the concrete transition system generated by a two-dimensional irreducible MPL system. □

Proof Let A denote the system matrix and let the cyclicity be c. We define the concrete transition system as the transition system associated with the MPL system restricted to $E(A^{\otimes c})$. We first construct an abstract transition system that bisimulates the concrete transition system. Then the claim follows from Proposition 3.8.

Recall that the eigenspace of the irreducible MPL system is a max-plus cone [22, Ths. 3.100 and 3.101]. Proposition 3.3 implies the eigenspace of a two-dimensional MPL system is a DBM or in fact a stripe. First we determine the initial partition Π_0 of the eigenspace (cf. Section 3.3.1). The initial partition is defined as the partition (of the eigenspace) generated by the atomic propositions, i.e. $\Pi_0 = \Pi_{AP}$. In this case, each block is a stripe since the states satisfying each atomic proposition are a stripe. There is no need to consider the partition generated by the affine dynamics since this is a proof and is not going to be implemented. The periodic behavior of the states in the eigenspace and the fact that each block is a stripe imply each abstract state has a self loop. From Theorem 3.2, the abstract transition system bisimulates the concrete transition system. □

For a higher dimensional irreducible MPL system, the existence of an abstract transition system that bisimulates the corresponding concrete transition system depends on the cyclicity of A, as stated in the following result.

Proposition 3.10 Given an irreducible MPL system with state matrix A, if the cyclicity of A is equal to 1 and the states in the eigenspace $E(A)$ satisfy the same set of atomic propositions, then there exists an abstract transition system that bisimulates the corresponding concrete transition system. □

Proof We define the concrete transition system as the transition system associated with the MPL system restricted to the eigenspace. Then we abstract the concrete transition system. From Theorem 3.2, the abstract transition system bisimulates the concrete transition system. The conclusion follows from the application of Proposition 3.8.

Notice that if the cyclicity of A is 1 then the eigenspace equals the complete periodic behaviors, i.e. $E(A) = E(A^{\otimes c})$. Furthermore via [22, Ths. 3.100-3.101] we conclude that $E(A)$ is a max-plus cone. From Proposition 3.2, $E(A)$ can be expressed as a union of finitely many stripes that are not necessarily pairwise disjoint: in order to obtain a partition of $E(A)$ we can employ a generic refinement procedure. In this case, each block of the partition is a stripe because $E(A)$ is a union of finitely many stripes. Since each state in the eigenspace satisfies the same set of atomic propositions (see the assumption above), the obtained partition is proposition preserving. There is no need to refine the partition based on the affine dynamics since this is a proof and is not a procedure that is going to be implemented. From the periodic behavior of the states in the eigenspace and the fact that each block is a stripe, each abstract state has a self loop. □

3.4 Nonautonomous Max-Plus-Linear Systems

We introduce a transition system related to a nonautonomous MPL system. Notice that the transitions are action abstract in the sense that the transition system does not care which particular action $\mathbf{u}$ is responsible for the transition of the MPL system.

Definition 3.11 (Transition Systems Associated with Nonautonomous MPL Systems) Consider a nonautonomous MPL system (2.3) with $\mathcal{X}_0$ as the set of initial conditions, $\mathcal{U}$ as the set of possible inputs, and a set of atomic propositions AP together with the corresponding labeling function L. The associated transition system TS is a tuple $(S, Act, \longrightarrow, I, AP, L)$ with

- set of states $S = \mathbb{R}^n$,
- set of actions $Act = \{\tau\}$,
- there exists a transition relation $\mathbf{x} \xrightarrow{\tau} \mathbf{x}'$ if there exists an input $\mathbf{u} \in \mathcal{U}$ such that $\mathbf{x}' = A \otimes \mathbf{x} \oplus B \otimes \mathbf{u}$, and
- set of initial states $I = \mathcal{X}_0$.

A special symbol τ is used in cases where action names are irrelevant. □

The set of states satisfying each atomic proposition is assumed to be a DBM, i.e. for each $a \in AP$, the set of states $\{\mathbf{x} : a \in L(\mathbf{x})\}$ is a DBM. Furthermore the set of allowed inputs $\mathcal{U}$ is also assumed to be a DBM in the input space $\mathbb{R}^m$. Practically, this enables expressing upper or lower bounds on the separation between input events (schedules). If on the other hand there are no constraints on input events, we define $\mathcal{U} = \mathbb{R}^m$, which is also a DBM.

3.4.1 States: Partitioning Procedure

We construct a partition of S and then each state in the same block is mapped by the abstraction function f to a unique abstract state. More precisely we develop an approach to construct a partition Π_0 of the set of states S, where Π_0 is an AP partition [23, Def. 7.31] and each block is a DBM. The partition Π_0 is computed by using the procedure to generate an AP partition in Section 3.3.1, i.e. $\Pi_0 = \Pi_{AP}$.

Remark Recall that the dynamics that are active in each block of the partition Π_0 in Section 3.3.1 are affine. This fact is used to simplify the computation of transitions in the abstract transition system. As it will be clear in Section 3.4.2, in nonautonomous MPL systems, the computation of transitions in the abstract transition system does not use the dynamics in each block. □

Example Suppose that $AP = \{a\}$ and the set of states satisfying a is the following stripe $\{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 < 3\}$. The resulting AP partition contains three blocks, i.e. $B_1 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 < 0\}$, $B_2 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 < 3\}$, and $B_3 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 3\}$ as shown in Fig. 3.5.

Let us define the set of abstract states $\hat{S}$, the abstraction function f, and the labeling function of the abstract transition system $L_f : \hat{S} \to 2^{AP}$. Since $\Pi_0 = \Pi_{AP}$ contains 3 blocks, $\hat{S} = \{\hat{s}_1, \hat{s}_2, \hat{s}_3\}$. The abstraction function f and the labeling function L_f are defined as follows

$$f(\mathbf{x}) = \begin{cases} \hat{s}_1, & \text{if } x_1 - x_2 < 0, \\ \hat{s}_2, & \text{if } 0 \leq x_1 - x_2 < 3, \\ \hat{s}_3, & \text{if } x_1 - x_2 \geq 3, \end{cases} \qquad L_f(\hat{s}_i) = \begin{cases} \{a\}, & \text{if } i = 2, \\ \emptyset, & \text{if } i = 1, 3. \end{cases}$$ □

3.4.2 Transitions: One-Step Reachability

We investigate a technique to determine the transition relations of the abstract transition system, that is between two blocks of the partition induced by the abstraction function. The (concrete) states associated with an abstract state $\hat{s}$ equals to the inverse image of $\hat{s}$ w.r.t. the abstraction function f, i.e. $f^{-1}(\hat{s}) = \{s : f(s) = \hat{s}\}$. Recall that $f^{-1}(\hat{s})$ is a block or in fact a DBM.

If there exists a transition from an outgoing state s to an incoming state s' in the concrete transition system, i.e. $s \xrightarrow{\gamma} s'$, then there is a transition from $f(s)$ to $f(s')$ in the abstract transition system, i.e. $f(s) \xrightarrow{\gamma}_f f(s')$ [23, Def. 7.51]. Such a transition can be determined by a forward- or backward-reachability approach. According to the former, we calculate $f^{-1}(\hat{s}') \cap \mathit{Post}(f^{-1}(\hat{s}))$, whereas if we use the backward approach we compute $f^{-1}(\hat{s}) \cap \mathit{Pre}(f^{-1}(\hat{s}'))$. The nonemptiness of the resulting set characterizes the presence of a transition from $\hat{s}$ to $\hat{s}'$.

As in the autonomous case, we focus on the forward-reachability approach, since it is computationally more attractive than the backward one. Given an abstract state $\hat{s}$, we employ the PWA representation of the augmented matrix to compute the direct successors as

$$\mathit{Post}(f^{-1}(\hat{s})) = \{\bar{A} \otimes \bar{\mathbf{x}} : \bar{\mathbf{x}} \in f^{-1}(\hat{s}) \times \mathcal{U}\},$$

where $f^{-1}(\hat{s}) \times \mathcal{U}$ denotes the cross product of the sets $f^{-1}(\hat{s})$ and $\mathcal{U}$. Since $f^{-1}(\hat{s}) \times \mathcal{U}$ is a DBM, $\mathit{Post}(f^{-1}(\hat{s}))$ is a union of finitely many DBM. The complete approach to determine the transitions of the abstract transition system is shown in Algorithm 3.3, which incurs a worst-case complexity of $\mathcal{O}((n+m)^3|\hat{S}|^2 q_{\bar{A}})$, where $q_{\bar{A}}$ denotes the number of regions in the PWA system generated by augmented matrix $\bar{A}$.

Example Let us consider $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 = 1\}$ as the set of initial conditions of the nonautonomous MPL system in (2.5). Thus the set of initial abstract states is $I_f = \{\hat{s}_2\}$ [23, Def. 7.51]. Furthermore the set of possible inputs $\mathcal{U} = \{\mathbf{u} \in \mathbb{R}^2 : 0 < u_1 - u_2 < 2\}$. The abstract transition system is shown in Fig. 3.10 (left).

Next we demonstrate the computation of transitions discussed above via VeriSiMPL version 1.4. First we construct an *AP* partition. The following MATLAB script constructs an *AP* partition manually since we have not implemented the procedure to generate an *AP* partition in VeriSiMPL version 1.4:

```
>> D = cell(1,2), D{1} = zeros(3,3,3), D{2} = false(3,3,3)
>> D{1}(:,:,1) = [0 Inf Inf;Inf 0 Inf;Inf 0 0]
>> D{2}(:,:,1) = [true false false;false true false;false false true]
>> D{1}(:,:,2) = [0 Inf Inf;Inf 0 0;Inf 3 0]
>> D{2}(:,:,2) = [true false false;false true true;false false true]
>> D{1}(:,:,3) = [0 Inf Inf;Inf 0 -3;Inf Inf 0]
>> D{2}(:,:,3) = [true false false;false true true;false false true]
```

Then we construct the PWA system generated by the augmented matrix. This can be done

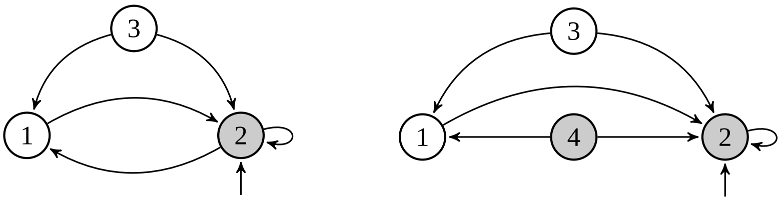

Fig. 3.10: The abstract transition system on the left simulates the concrete transition system, whereas the abstract transition system on the right bisimulates the concrete transition system. The initial state is indicated by an incoming arrow. The grayed states satisfy the atomic proposition a.

by using the function `mpl2pwa` (cf. Section 2.2.4):

```
>> Ampl = [2 5;3 3], Bmpl = [0 -Inf;-Inf 0]
>> [Anon,Bnon,Dnon] = mpl2pwa([Ampl Bmpl])
```

Finally we determine the transitions. In VeriSiMPL version 1.4, the computation of transitions for nonautonomous MPL systems is performed by the function `tsnon_trans`. This function requires a partition of the state space (`D`), PWA system generated by the augmented matrix (`Anon`,`Bnon`,`Dnon`), and the set of possible inputs (`U`). This function returns an adjacency matrix (`adj`). The following MATLAB script defines the set of possible inputs and computes the transitions:

```
>> U = cell(1,2)
>> U{1} = [0 Inf Inf;Inf 0 0;Inf 2 0]
>> U{2} = [true false false;false true false;false false true]
>> adj = tsnon_trans(D,Anon,Bnon,Dnon,U)
```

Suppose that we want to verify an invariant condition $\Box a$ over the abstract transition system. According to [23, p. 107], an invariant condition is satisfied by a transition system if and only if the condition is satisfied by the reachable states. Notice that the abstract transition system in Fig. 3.10 can reach state $\hat{s}_1$ where the proposition a is not satisfied. Thus the invariant condition is not satisfied by the abstract transition system. Recall that this does not imply that the invariant condition is not satisfied by the concrete transition system. $\Box$

3.4.3 Bisimulation-Quotienting Procedure

Having obtained an abstract transition system that simulates the concrete transition system, it makes sense to attempt deriving an abstract transition system that bisimulates the concrete transition system.

Theorem 3.3 Let TS be the concrete transition system generated by a nonautonomous MPL system and TS_f be the abstract transition system induced by an abstraction function $f : S \rightarrow \hat{S}$. The binary relation $\mathcal{R} = \{(f(s), s) : s \in S\}$ is a simulation for (TS_f, TS) if and only if $f^{-1}(\hat{s}) \subseteq Pre(f^{-1}(\hat{s}'))$ for each transition $\hat{s} \longrightarrow_f \hat{s}'$. $\Box$

Proof Notice that conditions 1 and 2(a) in Definition 3.6 are satisfied by $\mathcal{R}$ since $\{(s, f(s)) : s \in S\}$ is a simulation for (TS, TS_f). We will show that condition 2(b) is equivalent to $f^{-1}(\hat{s}) \subseteq Pre(f^{-1}(\hat{s}'))$ for each transition $\hat{s} \longrightarrow_f \hat{s}'$.

According to condition 2(b), for all $(\hat{s}, s) \in \mathcal{R}$ it holds that: if $\hat{s}' \in Post(\hat{s})$, there exists $s' \in Post(s)$ with $(\hat{s}', s') \in \mathcal{R}$. Equivalently, assuming that there is a transition $\hat{s} \longrightarrow_f \hat{s}'$, it holds that: for all $s \in f^{-1}(\hat{s})$, there exists $s' \in Post(s)$ with $s' \in f^{-1}(\hat{s}')$, i.e. $s \in Pre(f^{-1}(\hat{s}'))$. □

The procedure to generate an abstract transition system that bisimulates the concrete transition system works as follows. For each transition $\hat{s} \longrightarrow_f \hat{s}'$ with $f^{-1}(\hat{s}) \setminus Pre(f^{-1}(\hat{s}'))$ is not empty, the set of states $f^{-1}(\hat{s})$ is refined according to $Pre(f^{-1}(\hat{s}'))$. Then the incoming and outgoing transitions are updated. The preceding steps are repeated until all transitions satisfy the condition in Theorem 3.3. Unfortunately, such a procedure in general does not necessarily terminate. As a side note, the procedure to compute $Pre(f^{-1}(\hat{s}'))$ will be discussed in Section 4.3.

With focus on the refinement step, suppose that there exists a transition $\hat{s} \longrightarrow_f \hat{s}'$ such that $f^{-1}(\hat{s}) \setminus Pre(f^{-1}(\hat{s}'))$ is not empty. The refinement step generates a partition of $f^{-1}(\hat{s})$ such that each block is a DBM and each block is either a subset of $Pre(f^{-1}(\hat{s}'))$ or not intersected with $Pre(f^{-1}(\hat{s}'))$. Our approach is as follows. We first construct a partition of $f^{-1}(\hat{s})$ consisting of two blocks, i.e. $f^{-1}(\hat{s}) \cap Pre(f^{-1}(\hat{s}'))$ and $f^{-1}(\hat{s}) \setminus Pre(f^{-1}(\hat{s}'))$. Next the partition is refined such that each block is a DBM.

Example The abstract transition system in Fig. 3.10 (left) simulates the concrete transition system since the transition from $\hat{s}_2$ to $\hat{s}_1$ does not satisfy the condition in Theorem 3.3. One can show that $f^{-1}(\hat{s}_2) \setminus Pre(f^{-1}(\hat{s}_1)) = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 2\}$.

Let us apply the bisimulation-quotienting procedure to the abstract transition system in Fig. 3.10 (left). The set of states $f^{-1}(\hat{s}_2)$ is partitioned into two blocks, i.e. $\{\mathbf{x} : 0 \leq x_1 - x_2 \leq 2\}$ and $\{\mathbf{x} : 2 < x_1 - x_2 < 3\}$. After the refinement step, the partition Π_1 is a set of 4 blocks.

Next we characterize the abstract transition system associated with the refined partition Π_1. Since Π_1 contains 4 blocks, $\hat{S}' = \{\hat{s}'_1, \ldots, \hat{s}'_4\}$. The abstraction function f' and the labeling function $L_{f'}$ are defined as follows:

$$f'(\mathbf{x}) = \begin{cases} \hat{s}'_1, & \text{if } x_1 - x_2 < 0, \\ \hat{s}'_2, & \text{if } 0 \leq x_1 - x_2 \leq 2, \\ \hat{s}'_3, & \text{if } x_1 - x_2 \geq 3, \\ \hat{s}'_4, & \text{if } 2 < x_1 - x_2 < 3, \end{cases} \qquad L_{f'}(\hat{s}'_i) = \begin{cases} \{a\}, & \text{if } i = 2, 4, \\ \emptyset, & \text{if } i = 1, 3. \end{cases}$$

Recall that the set of initial states is $\{\mathbf{x} : x_1 - x_2 = 1\}$. Thus the set of initial abstract states is $I_{f'} = \{\hat{s}'_2\}$. The abstract transition system is depicted in Fig. 3.10 (right) and it bisimulates the concrete transition system since all transitions satisfy the condition in Theorem 3.3.

Let us construct the abstract transition system after the refinement procedure by using VeriSiMPL version 1.4. The refinement procedure has not been implemented in this version of VeriSiMPL. The following MATLAB script defines the refined partition manually:

```
>> D = cell(1,2), D{1} = zeros(3,3,4), D{2} = false(3,3,4)
>> D{1}(:,:,1) = [0 Inf Inf;Inf 0 Inf;Inf 0 0]
>> D{2}(:,:,1) = [true false false;false true false;false false true]
>> D{1}(:,:,2) = [0 Inf Inf;Inf 0 0;Inf 2 0]
>> D{2}(:,:,2) = [true false false;false true true;false true true]
>> D{1}(:,:,3) = [0 Inf Inf;Inf 0 -3;Inf Inf 0]
>> D{2}(:,:,3) = [true false false;false true true;false false true]
>> D{1}(:,:,4) = [0 Inf Inf;Inf 0 -2;Inf 3 0]
>> D{2}(:,:,4) = [true false false;false true false;false false true]
```

Then we construct the PWA system generated by the augmented matrix. This can be done by using the function `mpl2pwa` (cf. Section 2.2.4):

```
>> Ampl = [2 5;3 3], Bmpl = [0 -Inf;-Inf 0]
>> [Anon,Bnon,Dnon] = mpl2pwa([Ampl Bmpl])
```

Finally we define the set of possible inputs and use the function `tsnon_trans` (cf. Section 3.4.2) to determine the transitions as follows:

```
>> U = cell(1,2)
>> U{1} = [0 Inf Inf;Inf 0 0;Inf 2 0]
>> U{2} = [true false false;false true false;false false true]
>> adj = tsnon_trans(D,Anon,Bnon,Dnon,U)
```

Recall that the invariant condition $\Box a$ is not satisfied by the abstract transition system before the refinement, cf. Fig. 3.10 (left). One can check that the invariant condition is satisfied by the abstract transition system after refinement, cf. Fig. 3.10 (right). □

3.5 Implementation: VeriSiMPL

Most algorithms have been implemented as a MATLAB toolbox, “Verification via biSimulations of MPL models” (VeriSiMPL, as in “very simple”) [5], which is freely available for download at http://www.sourceforge.net/projects/verisimpl. MPL systems specified in MATLAB are abstracted to finite-state transition systems. The abstraction procedure runs in MATLAB and leverages sparse representations, fast manipulations based on vector calculus, parallel computing toolbox of MATLAB and optimized data structures such as DBM. The obtained abstraction can be exported to a text file in the PROcess MEta LAnguage (PROMELA) format. This enables the verification of MPL systems against LTL specifications within the SPIN model checker [71].

3.6 Computational Benchmark

To the best of the author’s knowledge, there is no tool that can be used to abstract MPL systems. Thus in order to test the practical efficiency of the proposed algorithms, we compute the runtime required to perform the abstraction of an MPL system into a finite abstract transition system, for increasing dimensions n of the given MPL system. We furthermore keep track of the number of states and of transitions of the obtained abstract transition system, which is directly related to the memory requirement of the technique.

For any given n, we generate row-finite matrices A with 2 finite elements placed uniformly at random in each row, as well as matrices B as column vectors where all elements are finite. The finite elements are uniformly generated integers taking values between 1 and 100. The set of allowed inputs $\mathcal{U}$ is conservatively selected to be equal to $\mathbb{R}$.

The experiments have been run on a 12-core Intel Xeon 3.47 GHz PC with 24 GB of memory. Over 10 independent experiments, Tables 3.1 and 3.2 report the (mean and maximum values for the) time needed to construct the abstract transition system, broken down over the two successive procedures for the generation of the abstract states and the transitions, respectively. The total number of states and of transitions in the abstract transition system are also reported.

Recall that the first step of the procedure (generation of abstract states) consists of the partitioning of the state space (Algorithm 3.2) and, for nonautonomous systems, of the construction of a PWA system over the augmented space (Algorithm 2.1), whereas the second step (generation of transitions) uses forward-reachability analysis to determine transitions between abstract states.

With regards to autonomous systems, as confirmed by Table 3.1, the bottleneck of the abstraction procedure resides on the generation of transitions and depends on the number of partitioning regions that is in the worst case exponential w.r.t. the dimension of the state space. On the other hand, for nonautonomous systems, as reported in Table 3.2, the computation time for generating the transitions is higher than in the autonomous case, since the procedure leverages the PWA system generated by the augmented matrix.

We have also performed similar computations for the case of autonomous systems with full matrices A (in a max-plus sense), which is likely to generate abstract models with more states. Elements are again uniformly distributed integers taking values between 1 and 100. Analogously to the above results, the bottleneck of the abstraction procedure also resides in the generation of the transitions. For an 8-dimensional MPL system over 10 independent experiments, the maximum time needed to compute the abstract transition system amounts to 20.11 minutes, which is made up of 6.90 and 13.21 minutes for generating the partitions and transitions, respectively.

Remark The abstraction and refinement procedure discussed in this chapter can be also used for autonomous and nonautonomous Min-Plus-Linear (MiPL) systems. In VeriSiMPL version 1.4, we have implemented the abstraction procedure for autonomous and nonautonomous MiPL systems. □

3.7 Summary

This chapter has introduced a new technique to generate finite abstractions of autonomous and nonautonomous Max-Plus-Linear (MPL) systems, characterized as finite-state transition systems. The procedure is based on the partitioning (covering) of the state (input) space and on the study of the one-step dynamics to relate partitioning regions. The resulting finite-state abstraction has been shown to either simulate or bisimulate the original MPL system.

Table 3.1: Numerical benchmark for autonomous MPL systems. Each entry represents the mean and maximal values over 10 independent experiments.

size of MPL system	time for generation of abstract states	time for generation of transitions	total number of abstract states	total number of transitions
3	{0.16;0.23} [sec]	{0.47;0.97} [sec]	{3.60;6.00}	{4.30;13.00}
4	{0.21;0.37} [sec]	{0.50;0.89} [sec]	{6.20;12.00}	{11.40;35.00}
5	{0.26;0.33} [sec]	{0.46;1.06} [sec]	{8.60;24.00}	{13.80;90.00}
6	{0.43;0.51} [sec]	{0.47;0.98} [sec]	{19.40;36.00}	{68.50;191.00}
7	{0.90;1.05} [sec]	{0.49;0.91} [sec]	{37.20;84.00}	{289.30;1278.00}
8	{1.58;1.83} [sec]	{0.58;0.97} [sec]	{58.00;160.00}	{512.30;1927.00}
9	{4.09;4.83} [sec]	{0.83;1.44} [sec]	{120.00;208.00}	{1.75;4.35}×10^3
10	{9.49;12.85} [sec]	{3.14;15.47} [sec]	{283.60;768.00}	{1.31;8.35}×10^4
11	{24.85;32.13} [sec]	{15.17;46.56} [sec]	{613.20;1104.00}	{1.87;4.82}×10^4
12	{1.19;1.94} [min]	{1.52;3.61} [min]	{1.20;2.03}×10^3	{4.76;14.08}×10^4
13	{3.53;5.04} [min]	{5.49;15.52} [min]	{1.92;3.81}×10^3	{1.91;8.50}×10^5
14	{12.03;29.65} [min]	{28.21;86.35} [min]	{4.16;8.13}×10^3	{7.83;34.50}×10^5
15	{53.58;78.31} [min]	{1.98;9.45} [hr]	{7.42;19.71}×10^3	{2.05;11.60}×10^6

Table 3.2: Numerical benchmark for nonautonomous MPL systems. Each entry represents the mean and maximal values over 10 independent experiments.

size of MPL system	time for generation of abstract states	time for generation of transitions	total number of abstract states	total number of transitions
3	{0.22;0.29} [sec]	{0.52;1.00} [sec]	{3.60;6.00}	{7.20;16.00}
4	{0.39;0.44} [sec]	{0.51;0.99} [sec]	{6.20;12.00}	{15.30;38.00}
5	{0.88;1.04} [sec]	{0.78;1.28} [sec]	{8.60;24.00}	{21.80;120.00}
6	{2.11;2.63} [sec]	{1.84;3.39} [sec]	{19.40;36.00}	{107.20;364.00}
7	{5.92;8.46} [sec]	{8.93;21.63} [sec]	{37.20;84.00}	{485.00;2520.00}
8	{12.66;18.33} [sec]	{30.55;107.43} [sec]	{58.00;160.00}	{730.30;2578.00}
9	{39.06;55.94} [sec]	{5.39;14.71} [min]	{120.00;208.00}	{2819.40;8742.00}
10	{98.42;141.97} [sec]	{43.21;156.55} [min]	{206.80;432.00}	{6211.60;16996.00}

The computational complexity of the approach has been fully quantified and its performance has been tested on a numerical benchmark, which has displayed a bottleneck that mainly depends on the number of generated partitioning regions. Still, the abstraction procedure comfortably manages models with reasonable size (15-dimensional, in the autonomous case) and can then be employed to study properties of the original MPL system in an original manner.

Chapter 4

Reachability Analysis of Max-Plus-Linear Systems

In this chapter, we discuss a computational approach to reachability analysis of MPL systems. Given a set of initial states, we characterize and compute its “reach tube,” namely the collection of set of reachable states (regarded step-wise as “reach sets”). By an alternative characterization of the MPL dynamics, we show that the exact computation of the reach sets can be performed quickly and compactly by manipulations of DBM, and further derive worst-case bounds on the complexity of these operations. The approach is also extended to backward reachability analysis.

4.1 Related Work

Reachability analysis is a fundamental problem in the area of formal methods, systems theory, and performance and dependability analysis. It is concerned with assessing whether a certain state of a system is attainable from given initial states of the system. The problem is particularly interesting and compelling over models with continuous components – either in time or in (state) space. Over the first class of models, reachability has been widely investigated over discrete-space systems, such as with timed automata [16, 25], Petri nets [76, 96], or hybrid automata [69]. On the other hand, much research has been directed to computationally push the envelope for reachability analysis of continuous-space models. Among the many approaches for deterministic dynamical systems, we report here the use of face lifting [44], the computation of flow-pipes via polyhedral approximations [34], later implemented in CheckMate [32], the formulation as solution of Hamilton-Jacobi equations [95] (related to the study of forward and backward reachability [94]), the use of ellipsoidal techniques [80], later implemented in [79], the use of differential inclusions [18], and finally the use of Taylor models [33]. Techniques that have displayed scalability features (albeit at the expense of precision due to the use of over-approximations) are the use of low-dimensional polytopes [63] and the computation of reachability using support functions [86].

With regards to MPL systems, reachability analysis from a *single* initial condition has been investigated in [38, 58, 61] by leveraging the computation of the reachability matrix, which leads to a parallel with reachability for discrete-time linear dynamical systems. It has been shown in [59, Sec. 4.13] that the reachability problem for autonomous MPL systems with a single initial condition is decidable – this result does not hold for a general, uncountable set of initial conditions. Furthermore, the existing literature does not deal with backward reachability analysis. Under the requirement that the set of initial conditions is expressed as a max-plus polyhedron [60, 120], forward reachability analysis can be performed over the max-plus algebra. Similar results hold for backward reachability analysis of autonomous MPL systems, where in addition the system matrix has to be max-plus invertible. Despite the requirements, computationally the approach based on max-plus polyhedra can be advantageous since its time complexity is polynomial. To the best of the author's knowledge, there are no direct approaches for solving the backward reachability problem of nonautonomous MPL systems in the max-plus algebra. Let us also mention that reachability analysis has been used to determine a static max-plus linear feedback controller for a nonautonomous MPL system such that the trajectories lie within a given target tube [13, Sec. 4.3]. In each event step, the target tube is then defined as a max-plus polyhedron [13, Eqs. (8) and (11)].

In this chapter we extend the forward and backward reachability computations of MPL systems by considering an arbitrary set of initial and final conditions, respectively. Furthermore in both cases, the system matrices do not have to be max-plus invertible.

4.2 Forward Reachability Analysis

The goal of forward reachability analysis is to quantify the set of possible states that can be arrived at under the model dynamics, at a particular event step or over a set of consecutive events, from a set of initial conditions and possibly under the choice of control actions. Recall that the state variables in MPL systems define the time of occurrence of discrete events (cf. Section 2.1). Two main notions can be introduced.

Definition 4.1 (Reach Set) Given an MPL system and a nonempty set of initial positions $\mathcal{X}_0 \subseteq \mathbb{R}^n$, the reach set $\mathcal{X}_N$ at event step $N > 0$ is the set of all states $\{\mathbf{x}(N, \mathbf{x}(0)) : \mathbf{x}(0) \in \mathcal{X}_0\}$ obtained via the MPL dynamics, possibly by application of any of the allowed controls. $\square$

Definition 4.2 (Reach Tube) Given an MPL system and a nonempty set of initial conditions $\mathcal{X}_0 \subseteq \mathbb{R}^n$, the reach tube is defined by the set-valued function $k \mapsto \mathcal{X}_k$ for any given $k > 0$ where $\mathcal{X}_k$ is defined. $\square$

Unless otherwise stated, in this work we focus on *finite-horizon* reachability: in other words, we compute the reach set for a finite index N (cf. Definition 4.1) and the reach tube for $k = 1, \ldots, N$, where $N < \infty$ (cf. Definition 4.2). Thus the reach tube always contains the reach set. While the reach set can be obtained as a by-product of the (sequential) computations used to obtain the reach tube, it can be as well calculated by a tailored procedure (one-shot).

In the computation of the quantities defined above, the set of initial conditions $\mathcal{X}_0 \subseteq \mathbb{R}^n$ and the set of allowed inputs at each event step $\mathcal{U}_k \subseteq \mathbb{R}^m$ are assumed to be a union of finitely many DBM and a single DBM, respectively. As it will become clear later, this assumption will shape the reach set $\mathcal{X}_k$ at any event step $k > 0$ as a union of finitely many DBM. In the more general case of arbitrary sets for $\mathcal{X}_0$ and $\mathcal{U}_k$, these can be over- or under-approximated by DBM. Notice that MPL dynamics are known to be nonexpansive [68, Lem. 3.10]: thus if $\mathcal{X}_0$ is (overapproximated by) a DBM, possible numerical errors associated with forward-reachability computations do not accrue [94]. To pin down notations for the complexity calculations below, we assume that $\mathcal{X}_k$ is a union of q_k DBM and in particular that the set of initial conditions $\mathcal{X}_0$ is a union of q_0 DBM.

4.2.1 Sequential Computation of the Reach Tube

This approach uses the one-step dynamics for autonomous and nonautonomous MPL systems iteratively. In each step, we make use of the DBM representation and the PWA dynamics to compute the successive reach set.

With focus on autonomous MPL systems, given a set of initial conditions $\mathcal{X}_0$, the reach set $\mathcal{X}_k$ is recursively defined as the image of $\mathcal{X}_{k-1}$.

$$\mathcal{X}_k = Im(\mathcal{X}_{k-1}) = \{A \otimes \mathbf{x} : \mathbf{x} \in \mathcal{X}_{k-1}\} = A \otimes \mathcal{X}_{k-1}.$$

In the dynamical systems and automata literature, the mapping *Im* is also known as *Post* [23, Def. 2.3]. From Corollary 3.1, if $\mathcal{X}_{k-1}$ is a union of finitely many DBM, then $\mathcal{X}_k$ is also a union of finitely many DBM. Then by induction, under the assumption that the set of initial states $\mathcal{X}_0$ is a union of finitely many DBM, it can be concluded that the reach set $\mathcal{X}_k$ is a union of finitely many DBM, for each $k \in \mathbb{N}$.

Given a state matrix A and a set of initial conditions $\mathcal{X}_0$, the general procedure for obtaining the reach tube works as follows: first, we construct the PWA system generated by A; then, for each $k = 1,\ldots,N$, the reach set $\mathcal{X}_k$ is obtained by computing $Im(\mathcal{X}_{k-1})$. The reach tube is then obtained by aggregating the reach sets.

The worst-case complexity can be assessed as follows. As discussed above, the complexity to characterize the MPL system via PWA dynamics is $\mathcal{O}(n^{n+3})$. Furthermore, the complexity of computing $Im(\mathcal{X}_{k-1})$ is $\mathcal{O}(q_{k-1}n^{n+3})$, for $k = 1,\ldots,N$. This results in an overall complexity of $\mathcal{O}(n^{n+3}\sum_{k=0}^{N-1} q_k)$. Notice that quantifying the cardinality q_k of the DBM union at each step k is not possible in general (cf. benchmark in Section 4.5).

Let us now look at cases where the structure of the MPL dynamics leads to savings for the computation of the reach tube. Recall that, given an $\mathcal{X}_0$ and a finite $N \in \mathbb{N}$, in order to compute $\mathcal{X}_N$, we need to calculate $\mathcal{X}_1,\ldots,\mathcal{X}_{N-1}$. Whenever the state matrix of an autonomous MPL system is irreducible, implying the existence of a periodic behavior (cf. Proposition 2.1), this can be simplified.

Proposition 4.1 Let $A \in \mathbb{R}_\varepsilon^{n\times n}$ be an irreducible matrix with max-plus eigenvalue $\lambda \in \mathbb{R}$ and cyclicity $c \in \mathbb{N}$. There exists a $k_0(\mathcal{X}_0)$ such that $\mathcal{X}_{k+c} = \lambda^{\otimes c} \otimes \mathcal{X}_k$, for all $k \geq k_0(\mathcal{X}_0)$. □

Recall that $k_0(\mathcal{X}_0) = \max_{\mathbf{x}\in\mathcal{X}_0} k_0(\mathbf{x})$ (cf. Definition 2.5). Thus if the state matrix is irreducible, we only need to compute $\mathcal{X}_1,\ldots,\mathcal{X}_{\min\{k_0(\mathcal{X}_0),N\}}$ in order to calculate $\mathcal{X}_N$, for any $N \in \mathbb{N}$. Furthermore if $\mathcal{X}_0$ is a union of finitely many stripes, the *infinite-horizon* reach tube is also a union of finitely many stripes and can be computed explicitly in finite time, as elaborated in the following statement.

Proposition 4.2 ([10, Th. 1]) Let $A \in \mathbb{R}_\varepsilon^{n\times n}$ be an irreducible matrix with cyclicity $c \in \mathbb{N}$. If $\mathcal{X}_0$ is a union of finitely many stripes, $\bigcup_{i=0}^{k_0(\mathcal{X}_0)+c-1} \mathcal{X}_i = \bigcup_{i=0}^{k} \mathcal{X}_i$, for all $k \geq k_0(\mathcal{X}_0) + c - 1$. □

Example Let us consider the unit square as the set of initial conditions $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ for the autonomous MPL system in (2.2). The reach sets are DBM given by $\mathcal{X}_1 = \{\mathbf{x} \in \mathbb{R}^2 : 1 \leq x_1 - x_2 \leq 2, 5 \leq x_1 \leq 6, 3 \leq x_2 \leq 4\}$, $\mathcal{X}_2 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 1, 8 \leq x_1 \leq 9, 8 \leq x_2 \leq 9\}$, and are shown in Fig. 4.1 (left).

In VeriSiMPL version 1.4, the procedure to determine the reach tube of autonomous MPL systems has been implemented in `mpl_reachtube_for`. The inputs are the PWA system (`A`,`B`,`D`), the initial states (`D0`), and the event horizon (`N`). The initial states are a collection of finitely many DBM and the event horizon is a natural number. The output is the reach tube that is represented by $1 \times (\mathtt{N}+1)$ cell. For each $1 \leq i \leq \mathtt{N}+1$, the i-th element of the reach tube (`D0N`) contains the reach set $\mathcal{X}_{i-1}$, which is a collection of finitely many DBM. The following MATLAB script re-calculates the numerical example in the preceding paragraph:

```
>> Ampl = [2 5;3 3], N = 2
>> [A,B,D] = mpl2pwa(Ampl)
>> D0 = cell(1,2)
>> D0{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D0{2} = [true true true;true true false;true false true]
>> D0N = mpl_reachtube_for(A,B,D,D0,N)
```

The set of initial conditions can also be described as a stripe, for example $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : -1 \leq x_1 - x_2 \leq 1\}$. In this case, the reach sets are stripes given by $\mathcal{X}_1 = \{\mathbf{x} \in \mathbb{R}^2 : 1 \leq x_1 - x_2 \leq 2\}$ and $\mathcal{X}_2 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 1\}$. □

For *nonautonomous* MPL systems, given a set of initial conditions $\mathcal{X}_0$, the reach set $\mathcal{X}_k$ depends on the reach set at event step $k-1$ and on the set of inputs at event step k:

$$\mathcal{X}_k = \overline{Im}(\mathcal{X}_{k-1} \times \mathcal{U}_k) = \{\bar{A} \otimes \bar{\mathbf{x}} : \bar{\mathbf{x}} \in \mathcal{X}_{k-1} \times \mathcal{U}_k\}.$$

We can show by induction that the reach set $\mathcal{X}_k$ is a union of finitely many DBM, for $k \in \mathbb{N}$. In the base case ($k = 1$), since $\mathcal{X}_0$ is a union of finitely many DBM and $\mathcal{U}_1$ is a DBM, then $\mathcal{X}_0 \times \mathcal{U}_1$ is a union of finitely many DBM, which implies that its image $\mathcal{X}_1$ is a union of finitely many DBM (cf. Corollary 3.1). A similar argument holds for the inductive step.

Given a state matrix A, an input matrix B, a set of initial conditions $\mathcal{X}_0$, and a sequence of sets of inputs $\mathcal{U}_1, \ldots, \mathcal{U}_N$, the general procedure for obtaining the reach tube works as follows: first, we construct the PWA system generated by $\bar{A}$; then for each $k = 1, \ldots, N$, the reach set $\mathcal{X}_k$ is obtained by computing the image of $\mathcal{X}_{k-1} \times \mathcal{U}_k$ w.r.t. the PWA system.

Let us quantify the complexity of the procedure. Constructing the PWA system can be done in $\mathcal{O}((n+m)^{n+3})$. For each $k = 1, \ldots, N$, the complexity of computing $\mathcal{X}_k$ critically depends on the image computation and is $\mathcal{O}(q_{k-1}(n+m)^{n+3})$. The overall complexity is $\mathcal{O}((n+m)^{n+3} \sum_{k=0}^{N-1} q_k)$.

Example Let us consider the unit square as the set of initial conditions $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ for the nonautonomous MPL system in (2.5). The set of possible inputs is given by $\mathcal{U}_1 = \{\mathbf{u} \in \mathbb{R}^2 : 4 \leq u_1 \leq 5, 4 \leq u_2 \leq 5\}$ and $\mathcal{U}_2 = \{\mathbf{u} \in \mathbb{R}^2 : 8 \leq u_1 \leq 9, 8 \leq u_2 \leq 9\}$. The reach sets are DBM given by $\mathcal{X}_1 = \{\mathbf{x} \in \mathbb{R}^2 : 5 \leq x_1 \leq 6, 4 \leq x_2 \leq 5\}$, $\mathcal{X}_2 = \{\mathbf{x} \in \mathbb{R}^2 : 9 \leq x_1 \leq 10, 8 \leq x_2 \leq 9\}$, and are shown in Fig. 4.1 (right).

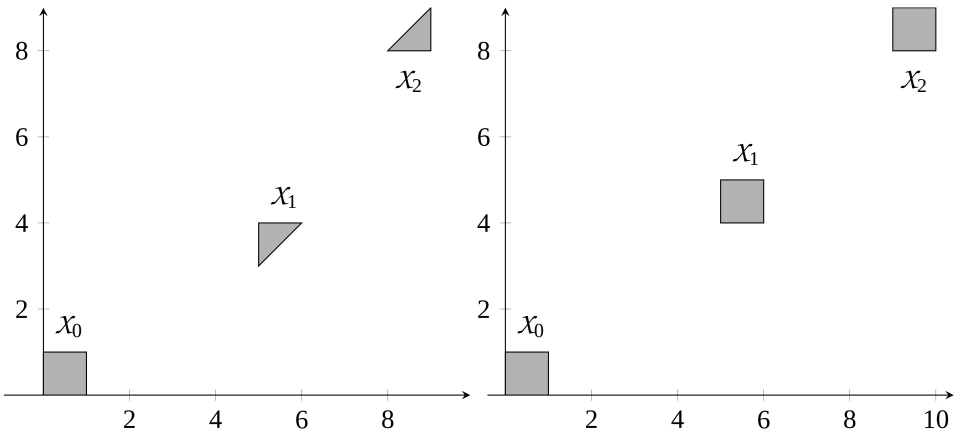

Fig. 4.1: The left plot represents forward reach tube for the autonomous MPL system in (2.2) *over 2 event steps. The right plot represents forward reach tube for the nonautonomous MPL system in* (2.5) *over 2 event steps. The initial states are denoted by* X_0*. The sets* X_1 *and* X_2 *represent the states reachable in 1 and 2 steps, respectively.*

Let us re-calculates the numerical example in the preceding paragraph by using VeriSiMPL version 1.4. First we construct the PWA system generated by the augmented matrix using the function `mpl2pwa` (cf. Section 2.2.4):

```
>> Ampl = [2 5;3 3]
>> Bmpl = [0 -Inf;-Inf 0]
>> [Anon,Bnon,Dnon] = mpl2pwa([Ampl Bmpl])
```

Then we define the initial states and the set of possible inputs at the first two steps:

```
>> D0 = cell(1,2)
>> D0{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D0{2} = [true true true;true true false;true false true]
>> U = cell(1,2)
>> U{1} = zeros(3,3,2)
>> U{1}(:,:,1) = [0 5 5;-4 0 Inf;-4 Inf 0]
>> U{1}(:,:,2) = [0 9 9;-8 0 Inf;-8 Inf 0]
>> U{2} = false(3,3,2)
>> U{2}(:,:,1) = [true true true;true true false;true false true]
>> U{2}(:,:,2) = U{2}(:,:,1)
```

In VeriSiMPL version 1.4, the procedure to compute the reach tube of nonautonomous MPL systems has been implemented in the function `mplnon_reachtube_for`. The inputs are the PWA system generated by the augmented matrix (`Anon`,`Bnon`,`Dnon`), the initial states (`D0`), and the set of possible inputs (`U`). For each step i, the set of possible inputs $\mathcal{U}_i$ is the i-th DBM in `U`. This function returns the reach tube (`D0N`). The following MATLAB script computes the reach tube for 2 event steps:

```
>> D0N = mplnon_reachtube_for(Anon,Bnon,Dnon,D0,U)
```

□

4.2.2 One-Shot Computation of the Reach Set

In this section we design a procedure for computing the reach set for a specific event step N using a tailored (one-shot) procedure. Let us focus on autonomous MPL systems: given a set of initial conditions $\mathcal{X}_0$, we compute the reach set at event step N using

$$\mathcal{X}_N = (Im \circ \cdots \circ Im)(A) = Im^N(A) = \{A^{\otimes N} \otimes \mathbf{x} : \mathbf{x} \in \mathcal{X}_0\}.$$

Using Corollary 3.1, it can be seen that the reach set $\mathcal{X}_N$ is a union of finitely many DBM. Given a state matrix A, a set of initial conditions $\mathcal{X}_0$ with $\mathcal{X}_0$ being a union of finitely many DBM, and a finite index N, the general procedure for obtaining $\mathcal{X}_N$ is: 1) computing $A^{\otimes N}$; then 2) constructing the PWA system generated by it; finally 3) computing the image of $\mathcal{X}_0$ w.r.t. the obtained PWA system.

The worst-case complexity of computing the N-th max-algebraic power of an $n \times n$ matrix (cf. Section 2.1) is $\mathcal{O}(\lceil \log_2(N) \rceil n^3)$. Since $\mathcal{X}_0$ is in general a union of q_0 DBM, the overall complexity of the procedure is $\mathcal{O}(\lceil \log_2(N) \rceil n^3 + q_0 n^{n+3})$. In comparison with the complexity for computing the N-step reach tube, which amounted to an $\mathcal{O}(n^{n+3} \sum_{k=0}^{N-1} q_k)$, the one-shot procedure appears to be advantageous. However, notice that the bottleneck lies on the (exponential) complexity of Algorithm 2.1, which is applied to two different matrices ($A^{\otimes N}$ and A, respectively). Thus while in general comparing the performance of the sequential and one-shot approaches is difficult, Proposition 4.3 suggests that under some dynamical assumptions the number of PWA regions generated by $A^{\otimes N}$ is higher than that generated by A.

Proposition 4.3 Let $R_{\mathbf{g}}$ and $R_{\mathbf{g}'}$ be regions generated by $A \in \mathbb{R}_{\varepsilon}^{n \times n}$. If $Im(R_{\mathbf{g}}) \subseteq R_{\mathbf{g}'}$, then $R_{\mathbf{g}} \subseteq R_{\mathbf{g}''}$ for some region $R_{\mathbf{g}''}$ generated by $A^{\otimes 2}$. □

Proof In this proof, the coefficients $\mathbf{g}, \mathbf{g}', \mathbf{g}''$ are treated as functions from $\{1, \dots, n\}$ to $\{1, \dots, n\}$, e.g. $\mathbf{g} : i \mapsto g_i$, for $i = 1, \dots, n$. Recall that the affine dynamics in $R_{\mathbf{g}}$ are

$$x_i(k-1) = x_{\mathbf{g}(i)}(k-2) + A(i, \mathbf{g}(i));$$

and the ones in $R_{\mathbf{g}'}$ are

$$x_i(k) = x_{\mathbf{g}'(i)}(k-1) + A(i, \mathbf{g}'(i)).$$

Hence, the affine dynamics in $R_{\mathbf{g}''}$ can be formulated as a composition of the affine dynamics in $R_{\mathbf{g}'}$ and $R_{\mathbf{g}}$ as

$$\begin{aligned} x_i(k) &= x_{\mathbf{g}(\mathbf{g}'(i))}(k-2) + A(i, \mathbf{g}'(i)) + A(\mathbf{g}'(i), \mathbf{g}(\mathbf{g}'(i))), \\ &= x_{\mathbf{g}''(i)}(k-2) + A^{\otimes 2}(i, \mathbf{g}''(i)). \end{aligned}$$

Notice that $\mathbf{g}'' = \mathbf{g} \circ \mathbf{g}'$, where $\circ$ denotes the function composition operator. □

Of course obtaining a higher number of PWA regions relates to obtaining a reach set expressed with a higher number of DBM. The result above can be generalized to $A^{\otimes N}$ as follows. Let $R_{\mathbf{g}^{(0)}},\ldots,R_{\mathbf{g}^{(N-1)}}$ be regions generated by A. If $Im^i(R_{\mathbf{g}^{(0)}}) \subseteq R_{\mathbf{g}^{(i)}}$, for each $i = 1,\ldots,N-1$, then it can be shown by induction that there exists a region $R_{\mathbf{g}^{(N)}}$ generated by $A^{\otimes N}$, such that $R_{\mathbf{g}^{(0)}} \subseteq R_{\mathbf{g}^{(N)}}$, where $\mathbf{g}^{(N)} = \mathbf{g}^{(0)} \circ \cdots \circ \mathbf{g}^{(N-1)}$.

On the side, let us remark that if the MPL dynamics are characterized by an irreducible matrix A, then the above figures should substitute the quantity N with $\min\{N, k_0(A)\}$.

Implementation In VeriSiMPL version 1.4, the one-shot procedure for autonomous MPL systems has been implemented in the function `mpl_reachset_for`. The inputs are the state matrix (`Ampl`), the initial states (`D0`), and the event horizon. This function returns the reach set (`DN`) as a 1×2 cell: the first element is the set of initial states and the second one is the reach set at the desired event step. Recall that both the initial states and the reach set are a collection of finitely many DBM. The following MATLAB script computes the reach set of the autonomous MPL system in (2.2) where the initial states are $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$:

```
>> Ampl = [2 5;3 3]
>> D0 = cell(1,2)
>> D0{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D0{2} = [true true true;true true false;true false true]
>> DN = mpl_reachset_for(Ampl,D0,2)
```

□

A similar technique can be applied to *nonautonomous* MPL systems. Given a set of initial conditions $\mathcal{X}_0$, the reach set at event step N is computed by using the following formula:

$$\mathcal{X}_N = [A^{\otimes N}, A^{\otimes(N-1)} \otimes B, \ldots, B] \otimes (\mathcal{X}_0 \times \mathcal{U}_1 \times \cdots \times \mathcal{U}_N).$$

From Corollary 3.1, the reach set $\mathcal{X}_N$ is again a union of finitely many DBM, since $\mathcal{X}_0 \times \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$ is a union of finitely many DBM. Recall that $\mathcal{X}_0$ is a union of q_0 DBM and $\mathcal{U}_1,\ldots,\mathcal{U}_N$ are DBM.

Given a state matrix A, an input matrix B, a set of initial conditions $\mathcal{X}_0$, a sequence of sets of inputs $\mathcal{U}_1,\ldots,\mathcal{U}_N$, the general procedure for obtaining $\mathcal{X}_N$ is: 1) generating $[A^{\otimes N}, A^{\otimes(N-1)} \otimes B, \ldots, B]$; then 2) constructing the PWA system generated by it; finally 3) computing the image of $\mathcal{X}_0 \times \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$ w.r.t. the PWA system.

Let us determine the complexity of the approach. In order to generate the matrix $[A^{\otimes N}, A^{\otimes(N-1)} \otimes B, \ldots, B]$, first we compute $A^{\otimes i}$, for $i = 2,\ldots,N$; then $A^{\otimes i} \otimes B$, for $i = 1,\ldots,N-1$, which leads to a worst-case complexity $\mathcal{O}(Nn^3 + Nn^2m)$. Since the size of the obtained matrix is $n \times (n+mN)$, the complexity of the second and third steps is $\mathcal{O}((n+mN)^{n+3})$ and $\mathcal{O}(q_0(n+mN)^{n+3})$, respectively. Unfortunately, this approach is not tractable for problems over long event horizons, since the maximum number of regions of the PWA system is $(n+mN)^n$ and grows exponentially w.r.t. the event horizon N. In this instance, using the sequential procedure (cf. Section 4.2.1) can be advantageous.

4.3 Backward Reachability Analysis

The objective of backward reachability analysis is to determine the set of states that enter a given set of final conditions, possibly under the choice of control inputs. This setup is of practical importance, for instance in seeking the set of initial conditions leading to a set of undesired states for any choice of the inputs, as well as in the transient analysis of irreducible MPL systems. Similar to the forward instance, two main notions are first introduced.

Definition 4.3 (Backward Reach Set) Given an MPL system and a nonempty set of final positions $\mathcal{X}_0 \subseteq \mathbb{R}^n$, the backward reach set $\mathcal{X}_{-N}$ is the set of all states $\mathbf{x}(-N)$ that lead to $\mathcal{X}_0$ in N steps of the MPL dynamics, possibly by application of any of the allowed controls. $\Box$

Definition 4.4 (Backward Reach Tube) Given an MPL system and a nonempty set of final positions $\mathcal{X}_0 \subseteq \mathbb{R}^n$, the backward reach tube is defined by the set-valued function $k \mapsto \mathcal{X}_{-k}$ for any given $k > 0$ where $\mathcal{X}_{-k}$ is defined. $\Box$

Similar to the forward reachability instance, the set of final conditions $\mathcal{X}_0 \subseteq \mathbb{R}^n$ and the set of control actions at each event step $\mathcal{U}_{-k} \subseteq \mathbb{R}^m$ are assumed to be a union of finitely many DBM and a single DBM, respectively. In particular, we denote by q_{-k} the cardinality of the set of DBM representing $\mathcal{X}_{-k}$ and assume that the set of final conditions $\mathcal{X}_0$ is a union of q_0 DBM.

4.3.1 Sequential Computation of the Backward Reach Tube

Let us focus on autonomous MPL systems: given a set of final conditions $\mathcal{X}_0$, for each $k = 1, \ldots, N$ we determine the states that enter $\mathcal{X}_0$ in k event steps by the following recursion:

$$\mathcal{X}_{-k} = Im^{-1}(\mathcal{X}_{-k+1}) = \{\mathbf{x} \in \mathbb{R}^n : A \otimes \mathbf{x} \in \mathcal{X}_{-k+1}\}.$$

The mapping Im^{-1} is also known in the literature as *Pre* [23, Def. 2.3]. Whenever $\mathcal{X}_0$ is a union of finitely many DBM, by Corollary 3.1 it follows that the backward reach set $\mathcal{X}_{-k}$ is a union of finitely many DBM, for each $k > 0$. As in the forward reachability case, the procedure for obtaining the backward reach tube leverages the dynamics of the PWA system associated with matrix A and the recursion above.

The complexity of computing $Im^{-1}(\mathcal{X}_{-k+1})$ at any $k \in \{1, \ldots, N\}$ is $\mathcal{O}(q_{-k+1} n^{n+3})$. This results in an overall worst-case complexity of $\mathcal{O}(n^{n+3} \sum_{k=1}^{N} q_{-k+1})$, where in general it is not feasible to precisely quantify the cardinality q_{-k+1} of the DBM union set at step k.

In general, given an $\mathcal{X}_0$, in order to calculate $\mathcal{X}_{-N}$, where N is finite, we have to determine $\mathcal{X}_{-1}, \ldots, \mathcal{X}_{-N+1}$, except if the autonomous MPL system is irreducible. The following result is directly shown by the definition of $k_\emptyset$.

Proposition 4.4 Let $A \in \mathbb{R}_\varepsilon^{n \times n}$ be an irreducible matrix with cyclicity $c \in \mathbb{N}$. If $\mathcal{X}_0 \cap E(A^{\otimes c})$ is empty, $\mathcal{X}_{-k}$ is empty for all $k \geq k_\emptyset(\mathcal{X}_0)$. $\Box$

Recall that $k_\emptyset(\mathcal{X}_0) = \max_{\mathbf{x} \in \mathcal{X}_0} k_\emptyset(\mathbf{x})$ (cf. Definition 2.5). Notice that if $\mathcal{X}_0 \cap E(A^{\otimes c})$ is empty, from Proposition 4.4, $\mathcal{X}_{-k}$ is empty for $k \geq k_\emptyset(\mathcal{X}_0)$. On the other hand if $\mathcal{X}_0 \cap E(A^{\otimes c})$ is not empty, the backward reach set at or after $k_\emptyset(\mathcal{X}_0)$ steps depends only on $\mathcal{X}_0 \cap E(A^{\otimes c})$, i.e. it does not depend on $\mathcal{X}_0 \setminus (\mathcal{X}_0 \cap E(A^{\otimes c}))$. More precisely in the case of $\mathcal{X}_0 \cap E(A^{\otimes c})$ is not empty and $k \geq k_\emptyset(\mathcal{X}_0)$, we have $Im^k(\mathcal{X}_{-k}) \subseteq \mathcal{X}_0 \cap E(A^{\otimes c})$, thus $k_0(\mathcal{X}_{-k}) \leq k$. Recall that $k_0(\mathcal{X}_{-k}) = \max_{\mathbf{x} \in \mathcal{X}_{-k}} k_0(\mathbf{x})$ (cf. Definition 2.5).

Theorem 4.1 Let $A \in \mathbb{R}_{\varepsilon}^{n\times n}$ be an irreducible matrix with max-plus eigenvalue $\lambda \in \mathbb{R}$ and cyclicity $c \in \mathbb{N}$, then $\lambda^{\otimes(-c)} \otimes \mathcal{X}_{-k} \subseteq \mathcal{X}_{-k-c}$, for all $k \geq k_0(\mathcal{X}_0)$. □

Proof If $\mathcal{X}_0 \cap E(A^{\otimes c})$ is empty, the proposition is trivially satisfied (cf. Proposition 4.4). Next, we assume that $\mathcal{X}_0 \cap E(A^{\otimes c})$ is not empty and that $k \geq k_0(\mathcal{X}_0)$.

We will prove that each element of $\lambda^{\otimes(-c)} \otimes \mathcal{X}_{-k}$ enters the set of final conditions in $k+c$ steps, i.e. $A^{\otimes(k+c)} \otimes \lambda^{\otimes(-c)} \otimes \mathcal{X}_{-k} \subseteq \mathcal{X}_0$. Observe that since $A^{\otimes k_0(\mathcal{X}_{-k})} \otimes \mathcal{X}_{-k} \subseteq E(A^{\otimes c})$, from Proposition 2.1 we conclude that $A^{\otimes(k_0(\mathcal{X}_{-k})+c)} \otimes \mathcal{X}_{-k} = A^{\otimes k_0(\mathcal{X}_{-k})} \otimes \mathcal{X}_{-k} \otimes \lambda^{\otimes c}$. The preceding observation and the fact that $k_0(\mathcal{X}_{-k}) \leq k$ (see the discussion before this theorem) are used in the following steps:

$$\begin{aligned} A^{\otimes(k+c)} \otimes \mathcal{X}_{-k} \otimes \lambda^{\otimes(-c)} &= A^{\otimes(k-k_0(\mathcal{X}_{-k}))} \otimes (A^{\otimes(k_0(\mathcal{X}_{-k})+c)} \otimes \mathcal{X}_{-k}) \otimes \lambda^{\otimes(-c)} \\ &= (A^{\otimes(k-k_0(\mathcal{X}_{-k}))} \otimes A^{\otimes k_0(\mathcal{X}_{-k})}) \otimes \mathcal{X}_{-k} \\ &= A^{\otimes k} \otimes \mathcal{X}_{-k} \\ &\subseteq \mathcal{X}_0. \end{aligned}$$ □

Remark Since the result in Theorem 4.1 is not as strong as Proposition 4.1, for backward reachability we do not obtain a result similar to that in Proposition 4.2. □

Example Let us consider the unit square as the set of final conditions $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ for the autonomous MPL system in (2.2). The backward reach sets are the union of finitely many DBM given by $\mathcal{X}_{-1} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 3, x_1 = -2\} \cup \{\mathbf{x} \in \mathbb{R}^2 : -3 \leq x_1 \leq -2, -5 \leq x_2 \leq -4\}$, $\mathcal{X}_{-2} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 \leq -7, -8 \leq x_2 \leq -7\}$, and are shown in Fig. 4.2.

In VeriSiMPL version 1.4, the procedure for computing the backward reach tube of autonomous MPL systems has been implemented in the function `mpl_reachtube_back`. The inputs are the PWA system (`A`,`B`,`D`), the final states (`D0`), the event horizon, and the dimension of the domain of the PWA dynamics. The output is the backward reach tube that is represented by a $1 \times (\mathtt{N}+1)$ cell, where `N` denotes the event horizon. For each $1 \leq i \leq \mathtt{N}+1$, the i-th element of the backward reach tube (`DN0`) contains the backward reach set $\mathcal{X}_{-i+1}$ which is a collection of finitely many DBM. The following MATLAB script re-computes the numerical example in the preceding paragraph:

```
>> Ampl = [2 5;3 3]
>> [A,B,D] = mpl2pwa(Ampl)
>> D0 = cell(1,2)
>> D0{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D0{2} = [true true true;true true false;true false true]
>> DN0 = mpl_reachtube_back(A,B,D,D0,2,size(Ampl,2))
```

Let us also consider the case of a stripe as the set of final conditions: $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : -1 \leq x_1 - x_2 \leq 1\}$. In this case, the backward reach sets are stripes described by $\mathcal{X}_{-1} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 1\}$ and $\mathcal{X}_{-2} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \leq 1\}$. □

For *nonautonomous* MPL systems, given a set of final conditions $\mathcal{X}_0$, the backward reach set $\mathcal{X}_{-k}$ depends on the backward reach set and on the set of inputs at event step $-k+1$:

$$\mathcal{X}_{-k} = \{\mathbf{x} \in \mathbb{R}^n : \exists \mathbf{u} \in \mathcal{U}_{-k+1} \text{ s.t. } \bar{A} \otimes [\mathbf{x}^T, \mathbf{u}^T]^T \in \mathcal{X}_{-k+1}\}.$$

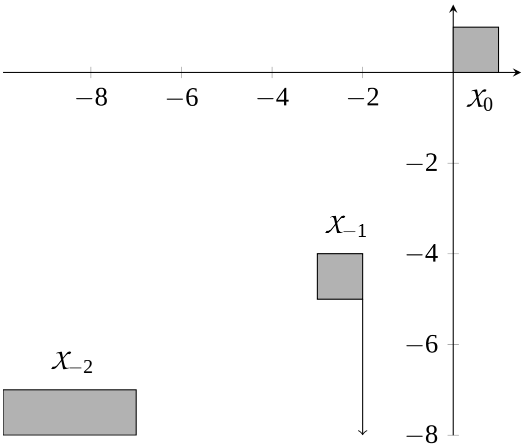

Fig. 4.2: Backward reach tube for the autonomous MPL system in (2.2) *over 2 event steps. The final states are denoted by $\mathcal{X}_0$. The sets $\mathcal{X}_{-1}$ and $\mathcal{X}_{-2}$ represent the states that reach $\mathcal{X}_0$ in 1 and 2 steps, respectively. The down-pointing arrow in $\mathcal{X}_{-1}$ indicates a half-line: that set can be expressed as a union of two DBM. The rectangle ($\mathcal{X}_{-2}$) at the bottom left is unbounded in the left direction.*

A practical procedure for computing the set $\mathcal{X}_{-k}$ is as follows: 1) compute the inverse image of $\mathcal{X}_{-k+1}$ w.r.t. the PWA system generated by $\overline{A}$, i.e. $\{\overline{\mathbf{x}} \in \mathbb{R}^{n+m} : \overline{A} \otimes \overline{\mathbf{x}} \in \mathcal{X}_{-k+1}\}$; then 2) intersect the inverse image with $\mathbb{R}^n \times \mathcal{U}_{-k+1}$; and finally 3) project the intersection over the state variables. As in the forward reachability case, it can be shown by using Corollary 3.1 that the backward reach set $\mathcal{X}_{-k}$ is a union of finitely many DBM, for $k \in \mathbb{N}$.

Example Let us consider the unit square as the set of final conditions $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ for the nonautonomous MPL system in (2.5). The set of possible inputs is given by $\mathcal{U}_0 = \{\mathbf{u} \in \mathbb{R}^2 : 0 \leq u_1 \leq 1, 0 \leq u_2 \leq 1\}$ and $\mathcal{U}_{-1} = \{\mathbf{u} \in \mathbb{R}^2 : -4 \leq u_1 \leq -3, -4 \leq u_2 \leq -3\}$. The backward reach sets are DBM given by $\mathcal{X}_{-1} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 \leq -2, x_2 \leq -4\}$, $\mathcal{X}_{-2} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 \leq -7, x_2 \leq -7\}$.

In VeriSiMPL version 1.4, the procedure to compute the backward reach tube of nonautonomous MPL systems has been implemented in the function `mplnon_reachtube_back`. The inputs are the PWA system generated by the augmented matrix (`Anon`,`Bnon`,`Dnon`), the final states (`D0`), and the set of possible inputs (`U`). For each step i, the set of possible inputs $\mathcal{U}_{-i+1}$ is the i-th DBM in `U`. The output is the backward reach tube (`DN0`). The following MATLAB script re-calculates the numerical example in the preceding paragraph:

```
>> Ampl = [2 5;3 3], Bmpl = [0 -Inf;-Inf 0]
>> [Anon,Bnon,Dnon] = mpl2pwa([Ampl Bmpl])
>> D0 = cell(1,2)
>> D0{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D0{2} = [true true true;true true false;true false true]
>> U = cell(1,2)
>> U{1} = zeros(3,3,2)
>> U{1}(:,:,1) = [0 1 1;0 0 Inf;0 Inf 0]
>> U{1}(:,:,2) = [0 -3 -3;4 0 Inf;4 Inf 0]
>> U{2} = false(3,3,2)
>> U{2}(:,:,1) = [true true true;true true false;true false true]
>> U{2}(:,:,2) = U{2}(:,:,1)
>> DN0 = mplnon_reachtube_back(Anon,Bnon,Dnon,D0,U)
```

□

4.3.2 One-Shot Computation of the Backward Reach Set

With focus on autonomous MPL systems, given a state matrix A, a set of final conditions $\mathcal{X}_0$ and a finite index N, the states that are able to enter $\mathcal{X}_0$ in N event steps are obtained similarly to those for the forward reachability case:

$$\mathcal{X}_{-N} = \{\mathbf{x} \in \mathbb{R}^n : A^{\otimes N} \otimes \mathbf{x} \in \mathcal{X}_0\}.$$

Further, by Corollary 3.1 it can be seen that the backward reach set $\mathcal{X}_{-N}$ is a union of finitely many DBM. Notice that because the complexity of computing the image and inverse image w.r.t. the MPL dynamics is the same (cf. Section 3.2.1), since the complexity of the approach critically depends on this operation, the overall complexity associated with the one-shot computation of the backward reach set amounts to that for the forward instance.

Implementation In VeriSiMPL version 1.4, the one-shot procedure for computing the backward reach set of autonomous MPL systems has been implemented in the function `mpl_reachset_back`. The inputs are the state matrix (`Ampl`), the final states (`D0`), and the event horizon. The output variable `D_N` is a 1×2 cell: the first element is the set of final states and the second one is the backward reach set at the desired event step. Recall that both the final states and the backward reach set are a collection of finitely many DBM. The following MATLAB script computes the backward reach set of the autonomous MPL system in (2.2) where the final states are $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le 1, 0 \le x_2 \le 1\}$:

```
>> Ampl = [2 5;3 3]
>> D0 = cell(1,2)
>> D0{1} = [0 1 1;0 0 Inf;0 Inf 0]
>> D0{2} = [true true true;true true false;true false true]
>> D_N = mpl_reachset_back(Ampl,D0,2)
```

□

For *nonautonomous* MPL systems, given a set of final conditions $\mathcal{X}_0$, the states that are able to enter $\mathcal{X}_0$ in N event steps are computed by using the following formula:

$$\mathcal{X}_{-N} = \{\mathbf{x}(-N) \in \mathbb{R}^n : \exists \mathbf{u}(-N+1) \in \mathcal{U}_{-N+1}, \ldots, \mathbf{u}(0) \in \mathcal{U}_0 \text{ s.t. } \mathbf{x}(0) \in \mathcal{X}_0\}.$$

Given a state matrix A, an input matrix B, a set of final conditions $\mathcal{X}_0$ that is a union of finitely many DBM, a sequence of sets of inputs $\mathcal{U}_0, \ldots, \mathcal{U}_{-N+1}$, the general procedure for obtaining $\mathcal{X}_{-N}$ is: 1) generating $[A^{\otimes N}, A^{\otimes (N-1)} \otimes B, \ldots, B]$; then 2) constructing the PWA system generated by it; 3) computing the inverse image of $\mathcal{X}_0$ w.r.t. the PWA system; 4) intersecting the inverse image with $\mathbb{R}^n \times \mathcal{U}_{-N+1} \times \cdots \times \mathcal{U}_0$; and finally 5) projecting the intersection w.r.t. the state variables. The backward reach set $\mathcal{X}_{-N}$ is a union of finitely many DBM. The complexity of the approach is the same as the corresponding for the forward case.

4.4 Applications

4.4.1 Safety Analysis

We consider the following safety problem (or in fact invariance): given an unsafe set, determine whether the states of an MPL system starting from a given initial set enter the unsafe set during the event interval $k = 0, \ldots, N$. This problem can be solved either by using forward- or backward-reachability analysis.

With focus on the forward-reachability analysis, we check whether the intersection of the N-step forward reach tube and the unsafe set is empty. The system is safe if and only if the intersection is empty.

With regards to the backward-reachability analysis, we compute the N-step backward reach tube, where the unsafe set is tagged as the set of final conditions, and then checking whether the intersection of the backward reach tube and the set of initial conditions is empty. If the intersection is empty, the system is deemed to be safe. If instead the system is not safe (namely, if the intersection is not empty), then the obtained intersection denotes the subset of the set of initial conditions leading to "unsafe dynamics."

Example Considering the autonomous MPL system in (2.2), suppose that there is a requirement on the departure times at station 2 to be at least three time units before those at station 1 and at most the same times as those at station 1. The safe set corresponds to $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 < 3\}$. The unsafe set is defined as the complement of the safe set, i.e. $\mathbb{R}^2 \setminus \mathcal{X} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 < 0\} \cup \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \geq 3\}$. Let us consider initial states of the MPL system that coincides with the safe set, i.e. $\mathcal{X}_0 = \mathcal{X}$.

By forward reachability computation, we obtain that $\mathcal{X}_1 = \{\mathbf{x} \in \mathbb{R}^2 : -1 < x_1 - x_2 \leq 2\}$ and that $\mathcal{X}_k = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 2\}$ for $k = 2, \ldots$ (cf. Proposition 4.2). Thus the system is not safe. By backward reachability computation, we obtain that $\mathcal{X}_{-1} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 > 2\}$ and that $\mathcal{X}_{-k} = \emptyset$ for $k = 2, \ldots$ (cf. Proposition 4.4). Thus the subset of the initial states leading to the unsafe set is $\{\mathbf{x} \in \mathbb{R}^2 : 2 < x_1 - x_2 < 3\}$. □

4.4.2 Transient Analysis

Classical results in the literature on transient analysis of MPL systems can be enhanced by computing a partition of $\mathbb{R}^n$ based on the length of the transient part k_0 via backward reachability analysis, as described next. First the set of final conditions $\mathcal{X}'_0$ is defined as the complete set of periodic behaviors $E(A^{\otimes c}) = \{\mathbf{x} \in \mathbb{R}^n : k_0(\mathbf{x}) = 0\}$. The eigenspace $E(A^{\otimes c})$ is a union of finitely many DBM, since $E(A^{\otimes c})$ [22, Sec. 3.7.2] is a max-plus cone and each max-plus cone can be expressed as a union of finitely many DBM (cf. Proposition 3.2). Then for each $k \in \mathbb{N}$, the backward reach set is obtained by

$$\mathcal{X}'_{-k} = \begin{cases} Im^{-1}(\mathcal{X}'_0) \setminus \mathcal{X}'_0, & \text{if } k = 1, \\ Im^{-1}(\mathcal{X}'_{-k+1}), & \text{if } k > 1. \end{cases}$$

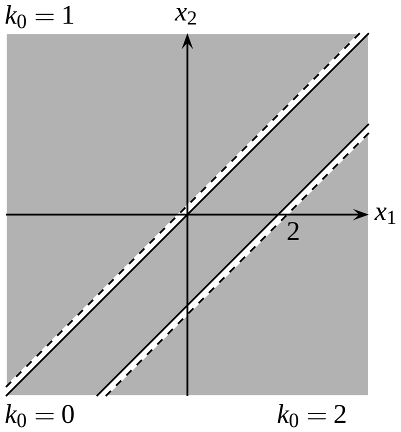

Fig. 4.3: Partition of $\mathbb{R}^2$ *for the MPL system in* (2.2) *based on the length of transient part* k_0.

Notice that the complete periodic behavior is a subset of its inverse image, i.e. $E(A^{\otimes c}) \subseteq Im^{-1}(E(A^{\otimes c}))$. Further one can see that $\mathcal{X}'_{-k} = \{\mathbf{x} \in \mathbb{R}^n : k_0(\mathbf{x}) = k\}$, for each $k \in \mathbb{N} \cup \{0\}$. The procedure is finite in time, since $\mathcal{X}'_1 \cap E(A^{\otimes c})$ is empty (cf. Proposition 4.4). More precisely, $\mathcal{X}'_{-k}$ is empty for $k \geq k_{\emptyset}(\mathcal{X}'_1) + 1$.

Example Let us demonstrate the procedure on the autonomous MPL system (2.2). Recall that the states associated with $k_0 = 0$ encompass the complete periodic behavior $\mathcal{X}'_0 = E(A^{\otimes c}) = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 2\}$. By using the procedure, the states corresponding to $k_0 = 1$ are given by $\mathcal{X}'_1 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 \leq 2\} \setminus \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 - x_2 \leq 2\} = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 < 0\}$. Finally the set of states such that $k_0 = 2$ can be computed by using the backward reachability analysis, which yields $\mathcal{X}'_2 = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 > 2\}$. The graphical representation of the state space partition is shown in Fig. 4.3. □

4.5 Numerical Benchmark

4.5.1 Implementation and Setup of the Benchmark

We have implemented the technique for forward and backward reachability computations on MPL systems in the VeriSiMPL ("very simple") software toolbox, which is freely available at [5].

In order to test the practical efficiency of the proposed algorithms we compute the runtime needed to determine the reach tube of an autonomous MPL system, for event horizon $N = 10$ and an increasing dimension n of the MPL system. The experiments reported here have been run on a 12-core Intel Xeon 3.47 GHz PC with 24 GB of memory. We also keep track of the number of regions of the PWA system generated from the MPL system. For any given n, we generate matrices A with 2 finite elements (in a max-plus sense) that are randomly placed in each row. The finite elements are randomly generated integers between 1 and 100. The test over a number of randomly generated dynamics goes against

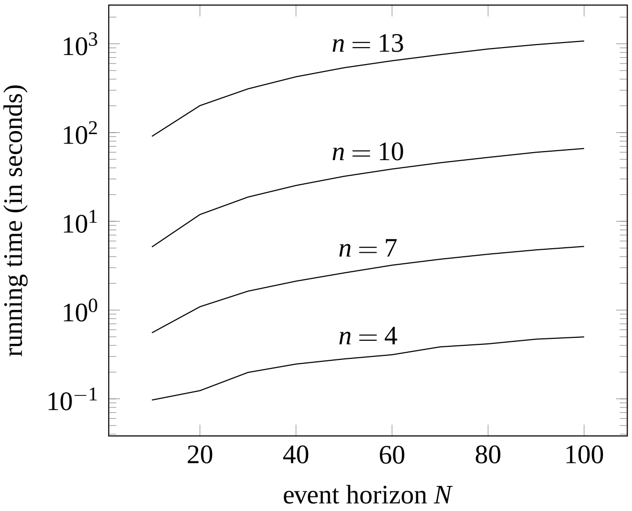

Fig. 4.4: Time needed to generate reach tube of autonomous MPL systems for different models size and event horizons.

biasing the experimental outcomes and allows claiming the applicability of our technique over general MPL systems. The set of initial conditions is selected as the unit hypercube, i.e. $\{\mathbf{x} \in \mathbb{R}^n : 0 \leq x_1 \leq 1, \ldots, 0 \leq x_n \leq 1\}$.

Over 10 independent experiments, Table 4.1 reports the average time needed to generate the PWA system and to compute the reach tube, as well as the corresponding average number of regions. As confirmed by Table 4.1, the time needed to compute the reach tube is monotonically increasing w.r.t. the dimension of the MPL system (as we commented previously this is not the case for the cardinality of the DBM union in the reach sets, which hinges on the specific dynamics of the MPL systems). For a fixed model size and dynamics, the growth of the computation time for forward reachability is linear with the event horizon as also shown in Fig. 4.4. We have also performed reachability computations for the case of the set of initial conditions described as a stripe, which has led to results that are quite analogue to those in Table 4.1. Further, the nonautonomous and backward-reachability cases can be handled similarly.

4.5.2 Comparison with an Alternative Computation

To the best of the author's knowledge, there does not exist any generally valid approach for forward reachability computation over MPL systems. This problem can be only alternatively assessed by leveraging the PWA characterization of the model dynamics (cf. Section 2.2.4). Forward reachability analysis of PWA systems can be best computed by the Multi-Parametric Toolbox (MPT, version 2.0[1]) [82]. However, the toolbox has some implementation requirements: the state space matrix A has to be invertible – this is in general not the case for MPL systems; the reach sets $\mathcal{X}_k$ have to be bounded – in our case the reach sets can be unbounded, particularly when expressed as stripes; further, MPT deals only with

---

[1] When we did the comparison, MPT version 3.0 [70] was not released yet.

Table 4.1: Numerical benchmark, autonomous MPL system: computation of the reach tube (average over 10 experiments)

size of MPL system	generation time for PWA system	number of regions of PWA system	generation time for reach tube	number of DBM of X_{10} (q_{10})
3	0.09 [sec]	5.80	0.09 [sec]	4.20
4	0.09 [sec]	12.00	0.13 [sec]	6.10
5	0.14 [sec]	22.90	0.20 [sec]	6.10
6	0.25 [sec]	42.00	0.25 [sec]	3.40
7	0.52 [sec]	89.60	0.72 [sec]	13.40
8	0.91 [sec]	145.00	0.73 [sec]	3.20
9	2.24 [sec]	340.80	2.25 [sec]	4.10
10	4.73 [sec]	700.80	8.23 [sec]	12.30
11	10.42 [sec]	1.44×10^3	15.49 [sec]	3.20
12	20.67 [sec]	2.87×10^3	117.98 [sec]	25.60
13	46.70 [sec]	5.06×10^3	5.27 [min]	16.90
14	82.94 [sec]	9.28×10^3	15.80 [min]	59.90
15	3.48 [min]	2.01×10^4	25.76 [min]	10.10
16	7.90 [min]	4.91×10^4	84.79 [min]	23.50
17	15.45 [min]	9.07×10^4	3.17 [hr]	68.70
18	29.13 [min]	1.58×10^5	5.82 [hr]	21.00
19	67.07 [min]	3.48×10^5	7.13 [hr]	5.00

Table 4.2: Time needed to generate the reach tube of a 10-dimensional autonomous MPL system for different event horizons (average over 10 experiments)

event horizon	20	40	60	80	100
VeriSiMPL	11.02 [sec]	17.94 [sec]	37.40 [sec]	51.21 [sec]	64.59 [sec]
MPT	47.61 [min]	1.19 [hr]	2.32 [hr]	3.03 [hr]	3.73 [hr]

full-dimensional polytopes – whereas the reach sets of interest may not necessarily be so; finally, MPT handles convex regions and over-approximates the reach sets X_k when necessary – our approach computes instead the reach sets exactly.

We have been concerned with benchmarking the proposed reachability computations with the described alternative. For the sake of comparison, we have constructed artificial examples (with invertible dynamics) and run both procedures in parallel, with focus on computation time rather than the obtained reach tubes. MPT can handle, in a reasonable time frame, models with dimension up to 10: in this instance (as well as lower-dimensional ones) our approach performs better (cf. Table 4.2). Notice that this is despite MPT being implemented as object code in the *C* language, whereas VeriSiMPL runs as interpreted code in MATLAB: this leaves quite some margin of improvement to our techniques and software.

4.6 Summary

This chapter has discussed a new computational technique for reachability analysis of max-plus-linear systems, which in essence amounts to exact and fast manipulations of difference-bound matrices through piecewise affine dynamics. The discussed procedure scales over 20-dimensional models thanks to a space partitioning approach that is adapted to the underlying model dynamics, as well as to a compact representation and fast manipulation of the quantities that come into play.

Chapter 5

Verification of Properties for Network Calculus Elements via Finite Abstractions

In this chapter we develop a framework for formal verification of properties for network calculus elements. Specifically, we leverage abstraction techniques developed in Chapter 3 to determine an upper bound on the backlog and virtual delay in a network. Suppose that we want to verify whether the backlog of a network is bounded by B'_{max}. Our approach works as follows. Initially we discretize the arrival and service curves with a period sufficiently small to capture the required details. More precisely the period is selected to be less than or equal to the sampling interval of all devices in the network. If the period is too large, we may lose some accuracy in measuring the backlog and virtual delay. We then characterize the dynamics as a switching MiPL system under some mild assumptions. Next we construct an abstract transition system based on B'_{max}. If the LTL formula representing the backlog is bounded by B'_{max} is verified, the backlog of the switching MiPL system is also bounded by B'_{max}. However, if the LTL formula is not verified, the backlog of the switching MiPL system may still be bounded by B'_{max}. A similar approach can be used to verify the virtual delay bound of a network.

5.1 Related Work

Our main contribution is to bridge the modeling framework of network calculus to the world of formal verification and synthesis. In order to do this we heavily rely on research already available in the area of formal methods applied to PWA systems. This is an area that has been quite active in the past decade providing abstractions satisfying approximate simulations [102], control synthesis methods [19, 118], analysis of stabilizability problems [93], or verification of general LTL formulae [117, 119]. In this work we decide to rely (with small modifications) on the work from Chapter 3, instead of other available options, for two reasons: it produces exact simulations relations; and produces formula-free abstractions, which allow for the modularity we look for in order to enable the joint analysis of control systems and communication networks. Thus, this work should be seen as a stepping stone towards the analysis of more complex problems involving networked control systems.

5.2 Network Calculus

In network calculus, a data flow is described by means of a cumulative function R, defined as the number of bits seen in time interval $[0,t]$. By convention, we assume that all flows are causal (i.e. $R(0) = 0$), unless otherwise specified.

An arrival curve specifies the maximum amount of arrivals allowed in a given time interval.

Definition 5.1 (Arrival Curve [85, Def. 1.2.1]) A flow R is said to be upper-constrained by an arrival curve α iff α is a non-negative wide-sense increasing function such that:

$$R(t) - R(s) \leq \alpha(t-s) \quad \forall 0 \leq s \leq t.$$

A function f is wide-sense increasing iff $f(s) \leq f(t)$ for all $s \leq t$. □

In communication networks, an element that forces a flow to conform to a certain arrival curve is called a shaper. An element that only checks whether the input conforms to an arrival curve without affecting the flow is called a policer. One of the most widely used classes of arrival curves is the class of affine arrival curves, defined by $\alpha_{r,b}(t) = rt + b$ for $t > 0$ and 0 otherwise. The parameters r and b are called rate and burstiness, respectively. Affine arrival curves are physically realizable by leaky buckets [29].

Service guarantees provided by servers to their input flows are characterized in network calculus by service curves. Servers can abstract physical network elements such as links, routers, and schedulers.

Definition 5.2 (Service Curve [85, Def. 1.3.1]) A system offers a service curve β iff β is a non-negative wide-sense increasing function with $\beta(0) = 0$ and

$$R^*(t) \geq \inf_{0 \leq s \leq t} \{R(s) + \beta(t-s)\} \quad \forall t \geq 0, \tag{5.1}$$

where R and R^* are input and output flows, respectively. □

One of the most widely used classes of service curves is the class of latency-rate service curves, defined by $\beta_{c,d}(t) = c(t-d)$ for $t > d$ and 0 otherwise. The parameters c and d are called rate and delay, respectively. The latency-rate server[1] is equivalent to concatenating a maximum-delay server and a guaranteed-rate server in series. A maximum-delay server is characterized by $\beta_d(t) = +\infty$ for $t > d$ and 0 otherwise. A guaranteed-rate server is characterized by $\beta_c(t) = ct$ for $t > 0$ and 0 otherwise.

The two most important properties that need to be analyzed in a communication network are the backlog and the virtual delay.

Definition 5.3 (Backlog [85, Def. 1.1.1]) The backlog at time t is the amount of data held inside the system, computed as: $B(t) = R(t) - R^*(t)$. □

---

[1] Service guarantees provided by the latency-rate server are characterized by a latency-rate service curve. Similar explanations hold for maximum-delay and guaranteed-rate servers.

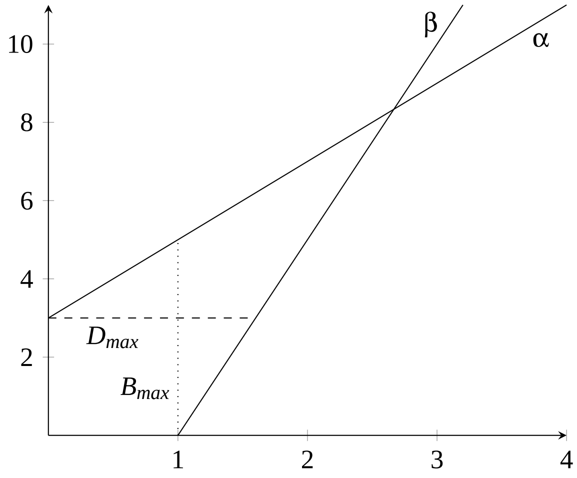

Fig. 5.1: Graphical representation of the arrival curve α *and service curve* β*. The dotted line represents the maximum vertical distance between* α *and* β*. The dashed line represents the maximum horizontal distance between* α *and* β*.*

Definition 5.4 (Virtual Delay [85, Def. 1.1.1]) The virtual delay at time t is the time spent inside the system by an arrival at time t if only earlier arrivals were processed before it, and is computed as: $d(t) = \inf\{\tau \geq 0 : R(t) \leq R^*(t+\tau)\}$. □

The backlog is bounded by

$$B_{max} = \sup_{s \geq 0}\{\alpha(s) - \beta(s)\},$$

which is the maximum vertical distance between α and β [85, Th. 1.4.1]. Correspondingly, the virtual delay is bounded by

$$D_{max} = \sup_{s \geq 0}\{\inf\{\tau \geq 0 : \alpha(s) \leq \beta(s+\tau)\}\},$$

which is the maximum horizontal distance between α and β [85, Th. 1.4.2]. In a communication network where servers use a first-in-first-out servicing policy, one can identify the virtual delay with the actual delay that packets experience.

Example Consider a latency-rate server with delay $d = 1$ and rate $c = 5$. We assume that the input flow is constrained by an affine arrival curve with burstiness $b = 3$ and rate $r = 2$. The graphical representation of the arrival and service curves is depicted in Fig. 5.1. One can see that the maximum backlog B_{max} is 5 and the maximum virtual delay D_{max} is 8/5. □

5.3 Min-Plus State-Space Formulation

We consider a single server and a policer that checks whether the input flow conforms with an affine arrival curve (cf. Fig. 5.2). The servers considered are characterized by latency-rate service curves. We consider worst-case scenarios by forcing the inequality in (5.1) to be an equality.

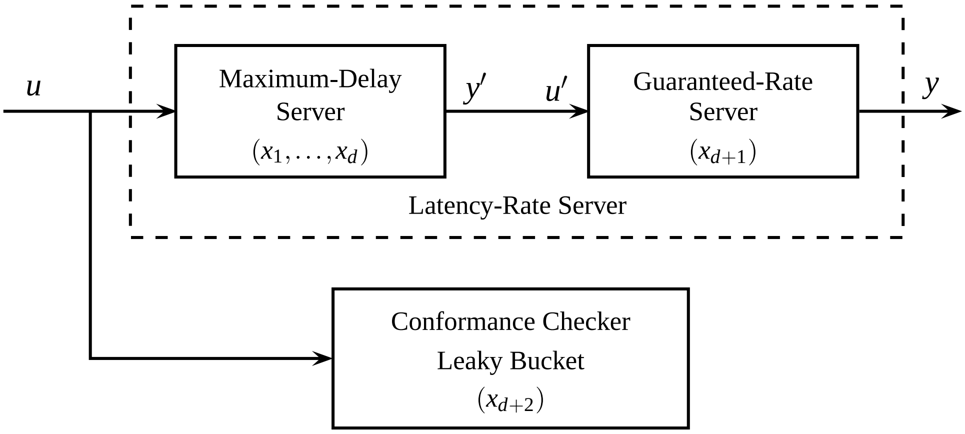

Fig. 5.2: Block diagram of the network calculus elements.

Remark A common approach in network calculus is using a shaper to guarantee that the flow entering a server conforms with a certain arrival curve. On the other hand, in this thesis we use a policer that checks whether the input conforms to an arrival curve without affecting the flow. Equivalently, the flow entering a server is not required to conform with a certain arrival curve. □

Recall that a latency-rate server is equivalent to concatenating a maximum-delay server and a guaranteed-rate server in series (cf. Section 5.2). Without loss of generality, we assume that the output of a maximum-delay server is fed to a guaranteed-rate server (cf. Fig. 5.2).

A discrete-time maximum-delay server is characterized by the service curve

$$\beta_d(k) = \begin{cases} +\infty, & \text{if } k > d, \\ 0, & \text{if } k \le d. \end{cases}$$

It delays the input d time units. The dynamics of the maximum-delay server is $y(k) = u(k-d)$, which can be written as a set of first-order recurrence relations (2.7) by introducing auxiliary variables

$$\begin{aligned} x_1(k) &= u(k), \\ x_i(k) &= x_{i-1}(k-1), \quad i = 2,\dots,d, \\ y'(k) &= x_d(k-1). \end{aligned}$$

For a discrete-time guaranteed-rate server, the service curve is

$$\beta_c(k) = \begin{cases} ck, & \text{if } k > 0, \\ 0, & \text{if } k \le 0. \end{cases}$$

At each time step, if the current backlog is greater than c, then the server dispatches c data units, otherwise the server will dispatch all of them. The dynamics can be represented by a single state x_{d+1} representing the amount of dispatched data:

$$\begin{aligned} x_{d+1}(k) &= c \otimes x_{d+1}(k-1) \oplus' u'(k), \\ y(k) &= x_{d+1}(k). \end{aligned}$$

The discrete-time service curve of a latency-rate server is

$$\beta_{c,d}(k) = \begin{cases} c(k-d), & \text{if } k > d, \\ 0, & \text{if } k \le d. \end{cases}$$

First, the server delays the input d time units, then the server dispatches them at most c data units at each time step. Since the output of a maximum-delay server is fed to a guaranteed-rate server, we have $u'(k) = y'(k)$ (cf. Fig. 5.2). The states of the latency-rate server are defined as the states of both servers, i.e. $x_1, \dots, x_{d+1}$. The corresponding dynamics are

$$\begin{aligned} x_1(k) &= u(k), \\ x_i(k) &= x_{i-1}(k-1), \quad i = 2, \dots, d, \\ x_{d+1}(k) &= x_d(k-1) \oplus' c \otimes x_{d+1}(k-1), \\ y(k) &= x_{d+1}(k). \end{aligned} \tag{5.2}$$

To characterize the incoming traffic, we employ affine arrival curves. Conformance to such curves can be checked by using a policer in the form of a leaky bucket [29, p. 4]. A discrete-time leaky bucket is composed of a token buffer (bucket) with a token generation rate r and a buffer size b. If the buffer is not full, r tokens are leaked into the buffer every time. When data arrives, enough tokens must be available in the buffer in order to allow the data to move forward. If the amount of data arrivals is greater than the amount of tokens in the buffer, a traffic violation is detected and the flow is considered non-conformant. Based on this description, one can represent the content of the bucket by some state h. The trajectory $h(k)$ thus obeys the following:

$$h(k) = \min\{h(k-1) + r - a(k), b\} \qquad h(0) = b, \tag{5.3}$$

where $a(k) = u(k) - u(k-1)$ is the number of data arrivals at time k and initially the buffer is assumed to be full. As a result, the conformance checking by a leaky bucket is transformed to observing whether $h(k) \ge 0$ holds for all $k \in \mathbb{Z}$ or not. The condition $h(k) \ge 0$ for all $k \in \mathbb{Z}$ means that the amount of data arrivals is smaller than or equal to the amount of tokens in the buffer at each time step.

The dynamics (5.3) can be expressed as an MiPL system, by letting $x_{d+2}(k) = h(k) + u(k)$:

$$x_{d+2}(k) = r \otimes x_{d+2}(k-1) \oplus' b \otimes u(k). \tag{5.4}$$

Thus, the conformance condition becomes $x_{d+2}(k) - x_1(k) \ge 0$ for all $k \in \mathbb{Z}$.

The following non-autonomous MiPL system represents the combined dynamics of the server (5.2) and conformance checker (5.4):

$$\begin{aligned} x_1(k) &= u(k), \\ x_i(k) &= x_{i-1}(k-1), \quad i = 2, \dots, d, \\ x_{d+1}(k) &= x_d(k-1) \oplus' c \otimes x_{d+1}(k-1), \\ x_{d+2}(k) &= r \otimes x_{d+2}(k-1) \oplus' b \otimes u(k). \end{aligned} \tag{5.5}$$

Recall that $x_1(k) = u(k)$ and the other state variables $x_2(k), \dots, x_d(k)$ are used as memory for the maximum-delay server, i.e. $u(k-1), \dots, u(k-d+1)$. The variable x_{d+1} is used to represent the amount of dispatched data.

The non-autonomous MiPL system (5.5) generates many more trajectories than the network elements in Fig. 5.2 can actually generate. One of the reasons for this sort of conservatism is that the current input $u(k)$ in (5.5) is not necessarily greater than or equal to the previous input $u(k-1)$ for all k, whereas the input flow is a wide-sense increasing function (cf. Definition 5.1). In order to mitigate this issue, we assume that the amount of data arrivals at each time takes values in a set of finitely many nonnegative real numbers, i.e. $u(k)-u(k-1) \in \{a^{(1)},\dots,a^{(n_m)}\}$. Thus the input can be defined explicitly as

$$x_1(k) = u(k) = a^{(\ell(k))} \otimes x_1(k-1),$$

where the mode $\ell(k) \in \{1,\dots,n_m\}$ characterizes the amount of data arrivals at time step k. In this case the mode that is active at each step is chosen in a purely nondeterministic fashion, i.e. the outcome is not known a priori. It follows that the dynamics can be formulated as an autonomous switching MiPL system where the state matrix at mode $\ell(k)$ is given by

$$A^{(\ell(k))} = \left[\begin{array}{cccccc|c}
a^{(\ell(k))} & +\infty & +\infty & \cdots & +\infty & +\infty & +\infty \\
0 & +\infty & +\infty & \cdots & +\infty & +\infty & +\infty \\
+\infty & 0 & +\infty & \cdots & +\infty & +\infty & +\infty \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\
+\infty & +\infty & +\infty & \ddots & +\infty & +\infty & +\infty \\
+\infty & +\infty & +\infty & \ddots & 0 & c & +\infty \\
\hline
a^{(\ell(k))}+b & +\infty & +\infty & \cdots & +\infty & +\infty & r
\end{array}\right].$$

We assume initially the backlog is zero and the bucket of the policer is full. In the switching MiPL system, the initial conditions are characterized by $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^{d+2} : x_1 = x_{d+1}, x_{d+2} - x_1 = b\}$, which is equivalent to $\{\mathbf{x} \in \mathbb{R}^{d+2} : x_1 = \cdots = x_{d+1}, x_{d+2} - x_1 = b\}$ as shown in Proposition 5.1. Notice that the set of initial states is a DBM (cf. Definition 3.1).

Proposition 5.1 Let us consider a latency-rate server and an affine arrival curve characterized by $\beta_{c,d}$ and $\alpha_{r,b}$, respectively. The following states

$$\{\mathbf{x} \in \mathbb{R}^{d+2} : x_1 = \cdots = x_{d+1}, x_{d+2} - x_1 = b\}$$

characterize that the backlog is zero and the bucket of the policer is full. □

Proof Recall the condition that the backlog is zero and the bucket of the policer is full is represented by $\{\mathbf{x} \in \mathbb{R}^{d+2} : x_1 = x_{d+1}, x_{d+2} - x_1 = b\}$. Since $x_2,\dots,x_d$ are delayed inputs, we have $x_1 \geq \cdots \geq x_d$, which implies $x_d \leq x_{d+1}$. Notice that x_d denotes the amount of data that has been in the system for at least d time units. Then x_{d+1} represents the amount of data that has been dispatched. The condition $x_d < x_{d+1}$ represents that the server was dispatching data that has been in the system for less than d time units. This condition violates the expected operation of the server since the data can only be dispatched after staying in the system for at least d time units. Thus we have $x_d = x_{d+1}$, which implies $x_1 = \cdots = x_{d+1}$. □

Example Let us illustrate the approach discussed in this section on a simple example. Suppose that the latency-rate server is characterized by delay $d = 1$ and rate $c = 5$. The input flow is constrained by an affine arrival curve with burstiness $b = 3$ and rate $r = 2$. The amount of data arrivals at each period takes values in $\{0,1,2,3,4,5\}$.

Next we construct the autonomous switching MiPL system. The number of modes is 6, i.e. $n_m = 6$, and the amount of data arrivals is described by $a^{(\ell)} = \ell - 1$ for $\ell = 1,\ldots,n_m$. The state matrix at mode ℓ is

$$A^{(\ell)} = \begin{bmatrix} \ell-1 & +\infty & +\infty \\ 0 & 5 & +\infty \\ \ell+2 & +\infty & 2 \end{bmatrix}. \tag{5.6}$$

The initial states are $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 = 0, x_3 - x_1 = 3\}$. □

5.4 Abstraction of Autonomous Switching Min-Plus-Linear Systems

Recall that the idea of abstraction is to replace a model to be verified by a smaller abstract model and to verify the latter instead of the original one, where both models are expressed as transition systems. Therefore let us introduce a transition system related to the autonomous switching MiPL systems generated by network calculus elements.

Definition 5.5 (Transition Systems Associated with Autonomous Switching MiPL Systems) Consider an autonomous switching MiPL system (2.8) with $\mathcal{X}_0$ as the set of initial conditions and a set of atomic propositions AP together with the corresponding labeling function L. The associated transition system TS is a tuple $(S, Act, \longrightarrow, I, AP, L)$ where

- set of states $S = \mathbb{R}^n$,
- set of actions $Act = \{\tau\}$,
- there exists a transition relation $\mathbf{x} \xrightarrow{\tau} \mathbf{x}'$ if there exists a mode ℓ such that $\mathbf{x}' = A^{(\ell)} \otimes' \mathbf{x}$, and
- set of initial states $I = \mathcal{X}_0$.

In cases where action names are irrelevant, we use a special symbol τ. □

Recall that the mode that is active in each step is an environmentally nondeterministic, i.e. the mode cannot be controlled. Thus we do not define the set of actions as the set of possible modes in Definition 5.5. As it will be clear in Section 5.4.1, the set of states satisfying each atomic proposition is a DBM, i.e. for each $a \in AP$, the set of states $\{\mathbf{x} : a \in L(\mathbf{x})\}$ is a DBM in the state space.

5.4.1 States: Partitioning Procedure

We construct a partition of S and then the abstraction function f maps each state in the same block to a unique abstract state. Each non-empty subset of S is called block. More precisely we develop an approach to construct a partition Π_0 of the set of states S, where Π_0 is an AP partition, each block is a DBM, and the dynamics in each block is switched affine. A partition is an *AP* partition if the labeling function L maps each state in the same block to a unique subset of atomic propositions [23, Def. 7.31]. The approach is as follows. We first determine an *AP* partition of S, denoted by Π_{AP}, where each block is a DBM. Then we determine a partition Π_{SAD} of S where each block is a DBM and the dynamics in each block are switched affine.[2] Finally the partition Π_0 is defined as the refinement of Π_{AP} and Π_{SAD}, i.e. $\mathcal{R}_{\Pi_0} = \mathcal{R}_{\Pi_{AP}} \cap \mathcal{R}_{\Pi_{SAD}}$. The notation $\mathcal{R}_{\Pi}$ denotes the equivalence relation induced by partition Π [23, Rem. 7.30].

The set of atomic propositions *AP* is defined as $\{\mathsf{CI}, \mathsf{BB}, \mathsf{DB}\}$, which stand for conformant input, backlog bounded, and virtual delay bounded, respectively. The atomic proposition CI is true if the input conforms with the arrival curve, namely the number of tokens in the bucket is nonnegative:

$$\{\mathbf{x} \in \mathbb{R}^{d+2} : \mathsf{CI} \in L(\mathbf{x})\} = \{\mathbf{x} \in \mathbb{R}^{d+2} : x_{d+2} - x_1 \geq 0\}.$$

Whenever the backlog is less than or equal to the maximum backlog B'_{max}, then BB is true. This corresponds to

$$\{\mathbf{x} \in \mathbb{R}^{d+2} : \mathsf{BB} \in L(\mathbf{x})\} = \{\mathbf{x} \in \mathbb{R}^{d+2} : x_1 - x_{d+1} \leq B'_{max}\}.$$

Finally DB is true whenever the virtual delay is less than the maximum delay D'_{max}:

$$\{\mathbf{x} \in \mathbb{R}^{d+2} : \mathsf{DB} \in L(\mathbf{x})\} = \{\mathbf{x} \in \mathbb{R}^{d+2} : u(k - D'_{max}) - y(k) \leq 0\}.$$

Notice that we need to store the input for D'_{max} steps. If $D'_{max} > d$, additional state variables can be added. The partition Π_{AP} is computed by using the procedure described in Section 3.3.1.

Example Considering the autonomous switching MiPL system in (5.6), let us determine the partition Π_{AP}. The states satisfying CI and BB are given by $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_3 \leq 0\}$ and $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \leq 5\}$, respectively. Here we select $B'_{max} = 5$. One can show that the partition Π_{AP} contains the following 4 blocks: $L^{-1}(\emptyset) = \{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, x_1 - x_3 > 0\}$, $L^{-1}(\{\mathsf{CI}\}) = \{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, x_1 - x_3 \leq 0, x_2 - x_3 < -5\}$, $L^{-1}(\{\mathsf{BB}\}) = \{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \leq 5, x_1 - x_3 > 0, x_2 - x_3 > -5\}$, and $L^{-1}(\{\mathsf{BB}, \mathsf{CI}\}) = \{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \leq 5, x_1 - x_3 \leq 0\}$ (cf. the left plot in Fig. 5.3). In this example, we focus on the verification of backlog bound property. Verifying the virtual delay property can be done similarly. ☐

With regards to the partition Π_{SAD}, we propose the following approach. Let $\Pi_{AD}^{(\ell)}$ denote the *AD* partition generated by $A^{(\ell)}$ using the procedure in Section 3.3.1 for all $\ell = 1, \ldots, n_m$. The partition Π_{SAD} is defined as the refinement of the preceding *AD* partitions, i.e. $\mathcal{R}_{\Pi_{SAD}} = \cap_{\ell=1}^{n_m} \mathcal{R}_{\Pi_{AD}^{(\ell)}}$. One can show that each block of Π_{SAD} is a DBM and the corresponding dynamics are switched affine.

Example Let us compute the partition Π_{SAD} generated by the autonomous switching MiPL system in (5.6). Skipping the details, the partition Π_{SAD} contains 14 blocks: $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, x_1 - x_3 < -6\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -6 \leq x_1 - x_3 < -5, x_2 - x_3 > -11\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -5 \leq x_1 - x_3 < -4, x_2 - x_3 > -10\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -4 \leq$

---

[2] *SAD* stands for "switched affine dynamics" and *SAD* does not represent the multiplication of matrix S, matrix A, and matrix D, unless stated explicitly.

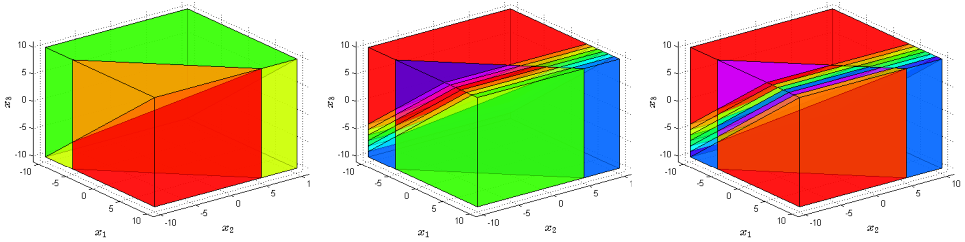

Fig. 5.3: The left, middle, and right plots are the graphical representation of Π_{AP}, Π_{SAP}, and Π_0 respectively.

$x_1 - x_3 < -3, x_2 - x_3 > -9\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -3 \leq x_1 - x_3 < -2, x_2 - x_3 > -8\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -2 \leq x_1 - x_3 < -1, x_2 - x_3 > -7\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, x_1 - x_3 \geq -1, x_2 - x_3 > -6\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, x_1 - x_3 < -6, x_2 - x_3 < -11\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, -6 \leq x_1 - x_3 < -5, x_2 - x_3 < -10\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, -5 \leq x_1 - x_3 < -4, x_2 - x_3 < -9\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, -4 \leq x_1 - x_3 < -3, x_2 - x_3 < -8\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, -3 \leq x_1 - x_3 < -2, x_2 - x_3 < -7\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, -2 \leq x_1 - x_3 < -1, x_2 - x_3 < -6\}$, and $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 \geq 5, x_1 - x_3 \geq -1\}$ as depicted in Fig. 5.3 (middle).

Recall that the partition Π_0 is the refinement of Π_{AP} and Π_{SAD}. Partition Π_0 contains 24 blocks: $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, x_1 - x_3 < -6\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -6 \leq x_1 - x_3 < -5, x_2 - x_3 > -11\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -5 \leq x_1 - x_3 < -4, x_2 - x_3 > -10\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -4 \leq x_1 - x_3 < -3, x_2 - x_3 > -9\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -3 \leq x_1 - x_3 < -2, x_2 - x_3 > -8\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -2 \leq x_1 - x_3 < -1, x_2 - x_3 > -7\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, x_1 - x_3 > 0, x_2 - x_3 > -5\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 < 5, -1 \leq x_1 - x_3 \leq 0, x_2 - x_3 > -6\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, x_1 - x_3 < -6, x_2 - x_3 < -11\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, x_1 - x_3 < -6, x_2 - x_3 < -11\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, -6 \leq x_1 - x_3 < -5, x_2 - x_3 < -10\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, -6 \leq x_1 - x_3 < -5, x_2 - x_3 < -10\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, -5 \leq x_1 - x_3 < -4, x_2 - x_3 < -9\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, -5 \leq x_1 - x_3 < -4, x_2 - x_3 < -9\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, -4 \leq x_1 - x_3 < -3, x_2 - x_3 < -8\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, -4 \leq x_1 - x_3 < -3, x_2 - x_3 < -8\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, -3 \leq x_1 - x_3 < -2, x_2 - x_3 < -7\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, -3 \leq x_1 - x_3 < -2, x_2 - x_3 < -7\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, -2 \leq x_1 - x_3 < -1, x_2 - x_3 < -6\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, -2 \leq x_1 - x_3 < -1, x_2 - x_3 < -6\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, x_1 - x_3 > 0\}$, $\{\mathbf{x} \in \mathbb{R}^3 : x_1 - x_2 > 5, -1 \leq x_1 - x_3 \leq 0, x_2 - x_3 < -5\}$, $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, x_1 - x_3 > 0, x_2 - x_3 > -5\}$, and $\{\mathbf{x} \in \mathbb{R}^3 : 5 \leq x_1 - x_2 \leq 5, -1 \leq x_1 - x_3 \leq 0\}$, as shown in Fig. 5.3 (right). $\square$

5.4.2 Transitions: One-Step Reachability

We investigate a technique to determine the transition relations of the abstract transition system. The transition relates two abstract states. Each abstract state is associated with a block via the abstraction function. More specifically the set of (concrete) states associated with an abstract state $\hat{s}$ is equal to the inverse image of $\hat{s}$ w.r.t. the abstraction function f, i.e. $f^{-1}(\hat{s}) = \{s : f(s) = \hat{s}\}$. Recall that $f^{-1}(\hat{s})$ is a block or in fact a DBM.

If there exists a transition from an outgoing state s to an incoming state s' in the concrete transition system, i.e. $s \xrightarrow{\gamma} s'$, then there is a transition from $f(s)$ to $f(s')$ in the

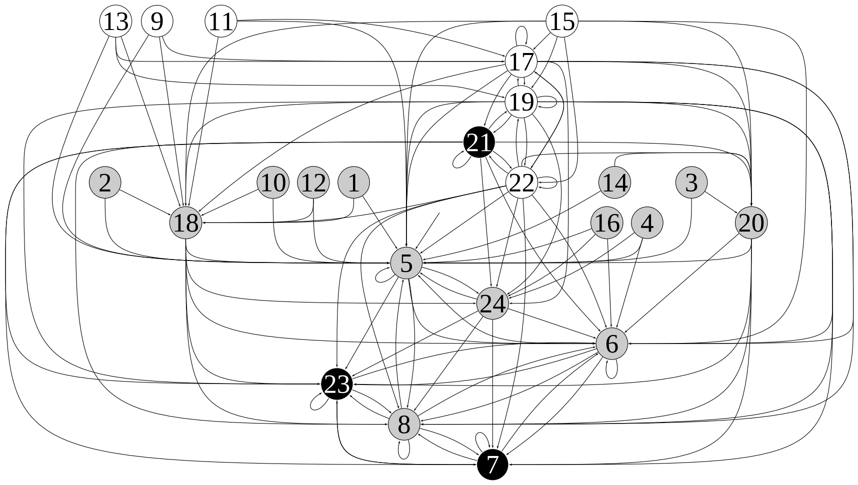

Fig. 5.4: Abstract transition system generated by the autonomous switching MiPL system in (5.6)*. The initial state is 5. The states satisfying CI and BB are the gray ones. The states that satisfy CI and that do not satisfy BB are the white ones. Finally the states that do not satisfy CI are the black ones.*

abstract transition system, i.e. $f(s) \xrightarrow{\gamma}_f f(s')$ (cf. Section 3.2.4). Such a transition can be determined by a forward- or backward-reachability approach. According to the former, we calculate $f^{-1}(\hat{s}') \cap Post(f^{-1}(\hat{s}))$, whereas if we use the backward approach we compute $f^{-1}(\hat{s}) \cap Pre(f^{-1}(\hat{s}'))$. The nonemptiness of the resulting set characterizes the presence of a transition from $\hat{s}$ to $\hat{s}'$.

We focus on the forward-reachability approach, since it is computationally more attractive than the backward one. Given an abstract state $\hat{s}$, we employ the PWSA representation of the system matrices to compute the direct successors as

$$Post(f^{-1}(\hat{s})) = \bigcup_{\ell=1}^{n_m} \{A^{(\ell)} \otimes \mathbf{x} : \mathbf{x} \in f^{-1}(\hat{s})\},$$

Since $f^{-1}(\hat{s})$ is a DBM, $Post(f^{-1}(\hat{s}))$ is a union of finitely many DBM (cf. Corollary 3.1). The complete approach to determine the transitions of the abstract transition system is shown in Algorithm 3.3.

Example The abstract transition system generated by the autonomous switching MiPL system in (5.6) is depicted in Fig. 5.4. ☐

Remark Having obtained an abstract transition system that simulates the concrete transition system, it makes sense to attempt deriving an abstract transition system that bisimulates the concrete transition system. The refinement procedure discussed in Section 3.4.3 can be used. Recall that such a procedure in general does not necessarily terminate. ☐

5.5 Formal Verification of Switching Min-Plus-Linear Systems

Let us now construct LTL formulae to check the backlog and virtual delay of the abstract transition system. We would like to check whether the backlog and virtual delay are bounded by the given maximum values, under the condition that the input conforms with the arrival curve. In other words, if the input is conformant, then the backlog (respectively, virtual delay) has to be at most B'_{max} (respectively, D'_{max}). The LTL formulae for backlog and virtual delay can be written as

$$\varphi_1 = \square \mathsf{CI} \Rightarrow \square \mathsf{BB} \qquad \text{and} \qquad \varphi_2 = \square \mathsf{CI} \Rightarrow \square \mathsf{DB}, \tag{5.7}$$

respectively. Unary operators bind stronger than the binary ones [23, p. 232]. Furthermore the LTL formulae in (5.7) are a liveness property [23, Def. 3.33]. Intuitively speaking, this means that any finite prefix can be extended so that the resulting infinite trace satisfies the property under consideration. Furthermore the LTL formulae in (5.7) are not a safety property since the only property that is both a safety and a liveness property is nonrestrictive [23, Lem. 3.35], i.e. it allows all possible behaviors.

To find the maximum backlog via model checking, we employ the following procedure. We first select a value of B'_{max} and generate an abstraction. If the abstraction does not satisfy φ_1, then we repeat the procedure with a higher value of B'_{max}, else we stop. The value of B'_{max} when the LTL property is verified will be the guaranteed upper bound of the backlog.

Example One can show that the LTL formula representing the backlog bound is verified by the abstract transition system in Fig. 5.4. However in general, given that the abstract transition system only simulates the original switching MiPL system, one cannot expect to verify the exact maximum bound for the backlog, but only a conservative bound. □

5.6 Summary

In this chapter we have proposed an approach to automatically verify network properties that can be critical for the correct functioning of control systems. In particular our approach allows to obtain delay bounds for aperiodic traffic sources. It is important to keep in mind that more complex specifications in LTL are amenable to be verified using the abstraction procedure we have proposed.

Chapter 6

Finite Abstractions of Stochastic Max-Plus-Linear Systems

This chapter investigates the use of finite abstractions to study the finite-horizon probabilistic invariance problem over Stochastic Max-Plus-Linear (SMPL) systems. SMPL systems are probabilistic extensions of discrete-event MPL systems that are employed in the engineering practice for timing and synchronization studies. We construct finite abstractions by re-formulating the SMPL system as a discrete-time Markov process, then tailoring formal abstraction techniques in the literature to generate a finite-state Markov Chain (MC), together with precise guarantees on the introduced approximation level. This finally allows probabilistic model checking of the obtained MC against the finite-horizon probabilistic invariance specification. The approach is practically implemented via a dedicated software, and elucidated in this chapter over numerical examples.

6.1 Related Work

Only a few approaches have been developed in the literature to study the steady-state behavior of SMPL systems, for example employing Lyapunov exponents and asymptotic growth rates [20–22, 57, 62, 92, 111]. The Lyapunov exponent of an SMPL system is the analogue of the max-plus eigenvalue for an autonomous MPL system. The Lyapunov exponent of SMPL systems under some assumptions has been studied in [111], and later extended to approximate computations under other technical assumptions in [62, p. 251]. The application of model predictive control and system identification to SMPL systems is studied in [54, 55]. In contrast, our work focuses on one-step properties of SMPL systems and is based on developing finite-state abstractions: this is parallel to the approach in Chapter 3 for (deterministic) MPL systems. To the best of the author's knowledge, this contribution represents the first work on finite-state abstractions of SMPL systems.

Verification techniques and tools for deterministic, discrete-time, finite-state systems have been widely investigated and developed in the past decades [78]. The application of formal methods to stochastic models is typically limited to discrete-state structures, either in continuous or in discrete time [24, 83]. Continuous-space models on the other hand require the use of finite abstractions, as it is classically done for example with finite bisimulations of timed automata, which can be computed via the known region construction [16]. With focus on stochastic models, numerical schemes based on Markov Chain (MC) approximations of stochastic systems have been introduced [30, 81], and applied to the approximate study of probabilistic reachability or invariance in [77, 103], however these finite abstractions do not come with explicit error bounds. In [3], a technique has been introduced to instead provide formal abstractions of discrete-time, continuous-space Markov models [2], with the objective of investigating their probabilistic invariance by employing probabilistic model checking over a finite MC. In view of scalability and of generality, the approach has been improved and optimized in [52]. Interestingly the procedure has been shown [1] to introduce an approximate probabilistic bisimulation of the concrete model [49].

6.2 The Probabilistic Invariance Problem

Let us consider events that are scheduled to occur regularly, that is let us select a time interval between consecutive events that is a positive given constant, say d. We call this a *regular schedule* and assume that it does not affect the time of occurrence of all events, e.g. any event may occur ahead of the regular schedule. In this chapter, we consider an N-step finite-horizon probabilistic invariance problem w.r.t. a regular schedule: more specifically, for each possible time of occurrence of initial event ($\mathbf{x}(0)$), we are interested in determining the probability that the time of occurrence of k-th event ($\mathbf{x}(k)$) remains close to the corresponding time of the regular schedule, for $k \in \{0,\ldots,N\}$. For instance, we may want to determine the probability that the time of occurrence of the first 3 events is at least 5 time units ahead of the given regular schedule, as well as at most 5 time units behind it. The invariant set is then defined as the desired time of occurrence w.r.t. the regular schedule.

The techniques in [3, 52], developed to provide the characterization and the computation of the probabilistic invariance problem over general Markov processes, can be directly applied to the SMPL system (2.9). However, in order to prevent the growth of the invariant set as the event horizon N increases (which in general leads to a decrease in computational performance), we reformulate the SMPL system based on the given regular schedule, so that a fixed invariant set is obtained. Since we are interested in the delay of event occurrences with respect to the given schedule, we introduce new variables defined as the difference between the states of the original SMPL system and the regular schedule. More precisely, first we define a vector $\mathbf{s}$ that characterizes the regular schedule. The dynamics of $\mathbf{s}$ are determined by the time duration $d \in \mathbb{R}$ between consecutive events and the arbitrary initial condition $\mathbf{s}(0) \in \mathbb{R}^n$, i.e. $\mathbf{s}(k) = d \otimes \mathbf{s}(k-1)$. As mentioned, new states are defined as the difference between the states of the original SMPL system (2.9) and the regular schedule $\mathbf{s}$, i.e. $\mathbf{z}(k) = \mathbf{x}(k) - \mathbf{s}(k)$ for $k \in \mathbb{N} \cup \{0\}$. The dynamics of the newly introduced SMPL system are then given by

$$\mathbf{z}(k) = [A(k) + D] \otimes \mathbf{z}(k-1), \tag{6.1}$$

where $D = [d_{ij}]_{i,j} \in \mathbb{R}^{n\times n}$ (i.e. d_{ij} is the entry of matrix D at row i and column j), $d_{ij} = s_j(0) - s_i(0) - d$, and $\mathbf{z}(k) = [z_1(k)\ldots z_n(k)]^T \in \mathbb{R}^n$. Notice that $A_{ij}(k) \otimes d_{ij}$ are independent for all $k \in \mathbb{N}$ and $i,j \in \{1,\ldots,n\}$. The density (resp., distribution) function of $A_{ij}(k) \otimes d_{ij}$ corresponds to the density (resp., distribution) function of $A_{ij}(k)$ shifted forward by d_{ij} units. The independent variable k again denotes an increasing event index, whereas the state variable $\mathbf{z}(k)$ defines the delay w.r.t. the schedule of occurrence of k-th event. If the delay is positive then the event occurs behind the schedule, whereas if the delay is negative then the event occurs ahead of the schedule. The next theorem shows that, much like the original model in (2.9), the new SMPL system can be described as a discrete-time homogeneous Markov process.

Theorem 6.1 (cf. Theorem 2.1) The SMPL system in (6.1) is fully characterized by the following conditional density function

$$t_z(\bar{\mathbf{z}}|\mathbf{z}) = \prod_{i=1}^{n} t_i(\bar{z}_i|\mathbf{z}), \ \text{ where}$$

$$t_i(\bar{z}_i|\mathbf{z}) = \sum_{j=1}^{n} \left[t_{ij}(\bar{z}_i - d_{ij} - z_j) \prod_{k=1,k\neq j}^{n} T_{ik}(\bar{z}_i - d_{ik} - z_k) \right], \ \text{ for all } i \in \{1,\ldots,n\},$$

for $\bar{\mathbf{z}}, \mathbf{z} \in \mathbb{R}^n$. Recall that the density function $t_{ij}(\cdot)$ is the derivative of the associated distribution function $T_{ij}(\cdot)$ w.r.t. its argument for all $i, j \in \{1,\ldots,n\}$. □

Remark If the time interval between consecutive occurrences is not the same for all $i \in \{1,\ldots,n\}$, then we obtain a time-inhomogeneous Markov process. In this case, the computational complexity of the procedure will greatly increase. □

Employing the introduced SMPL system (6.1), the problem can be formulated as the following N-step invariance probability

$$P_{\mathbf{z}_0}(\mathcal{A}) = \Pr\{\mathbf{z}(k) \in \mathcal{A} \text{ for all } k = 0,\ldots,N|\mathbf{z}(0) = \mathbf{z}_0\},$$

where $\mathcal{A}$ is called the invariant set and is assumed to be Borel measurable.

Let $\mathbb{I}_{\mathcal{X}} : \mathbb{R}^n \to \{0,1\}$ denote the indicator function of set $\mathcal{X} \subseteq \mathbb{R}^n$, i.e. $\mathbb{I}_{\mathcal{X}}(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathcal{X}$ and $\mathbb{I}_{\mathcal{X}}(\mathbf{x}) = 0$ if $\mathbf{x} \notin \mathcal{X}$. The next proposition provides a theoretical framework to study the problem.

Proposition 6.1 ([2, Lem. 1]) Consider value functions $V_k : \mathbb{R}^n \to [0,1]$, for $k \in \{0,\ldots,N\}$, computed through the following backward recursion:

$$V_k(\mathbf{z}) = \mathbb{I}_{\mathcal{A}}(\mathbf{z}) \int_{\mathcal{A}} V_{k+1}(\bar{\mathbf{z}}) t_z(\bar{\mathbf{z}}|\mathbf{z}) d\bar{\mathbf{z}}, \qquad \text{for all } \mathbf{z} \in \mathbb{R}^n,$$

initialized with $V_N(\mathbf{z}) = \mathbb{I}_{\mathcal{A}}(\mathbf{z})$ for all $\mathbf{z} \in \mathbb{R}^n$. Then $P_{\mathbf{z}_0}(\mathcal{A}) = V_0(\mathbf{z}_0)$. □

For any $k \in \{0,\ldots,N\}$, notice that $V_k(\mathbf{z})$ represents the probability that an execution of the SMPL system (6.1) remains within the invariant set $\mathcal{A}$ over the residual event horizon $\{k,\ldots,N\}$, starting from $\mathbf{z}$ at event step k.

This result characterizes the finite-horizon probabilistic invariance problem as a dynamic programming problem. Since an explicit analytical solution to the problem is generally impossible to find, we leverage the techniques developed in [3, 52] to provide a numerical computation with exact associated error bounds. This is elaborated in the next section.

6.3 Abstraction by a Finite State Markov Chain

We tailor the abstraction procedure presented in [3, Sec. 3.1] towards the goal of generating a finite-state MC $(\hat{S}, \hat{T})$ from a given SMPL system and an invariant set $\mathcal{A}$, then employ it to approximately compute the probabilistic invariance of interest.

Let $\hat{S} = \{\hat{s}_1, \dots, \hat{s}_{m+1}\}$ be a set of finitely many abstract states and $\hat{T} : \hat{S} \times \hat{S} \to [0,1]$ a related transition probability matrix, such that $\hat{T}(\hat{s}_i, \hat{s}_j)$ characterizes the probability of transitioning from state $\hat{s}_i$ to state $\hat{s}_j$ and thus induces a conditional discrete probability distribution over the finite space $\hat{S}$.

Given a bounded invariant set $\mathcal{A}$, Algorithm 6.1 provides a procedure to abstract an SMPL system by a finite state MC. The set $\hat{\mathcal{A}} = \{\hat{s}_1, \dots, \hat{s}_m\}$ denotes the discrete invariant set. In Algorithm 6.1, $f : \mathbb{R}^n \to \hat{S}$ represents the abstraction function, i.e. a map that associates to any concrete state $\mathbf{z} \in \mathbb{R}^n$ the corresponding abstract state $f(\mathbf{z}) \in \hat{S}$. Furthermore the concretization function $f^{-1}(\hat{s}) = \{\mathbf{z} : f(\mathbf{z}) = \hat{s}\}$ associates to any abstract state $\hat{s} \in \hat{S}$ the corresponding continuous partition set. Without loss of generality, we define $\hat{\mathcal{A}} = \{\hat{s}_1, \dots, \hat{s}_m\}$ as the abstract invariant set, i.e. the set of abstract states associated with the concrete invariant set $\mathcal{A}$. Additionally, notice that an absorbing discrete state $\hat{s}_{m+1}$ is added to the state space of the MC in order to render the transition probability matrix $\hat{T}$ stochastic.

Algorithm 6.1 Generation of a finite-state MC from an SMPL system and an invariant set
Input: An SMPL system in (6.1) and an invariant set $\mathcal{A}$
Output: A finite-state MC $(\hat{S}, \hat{T})$

1. select a finite partition of set $\mathcal{A}$ of cardinality m, as $\mathcal{A} = \cup_{i=1}^{m} \mathcal{A}_i$
2. define $\hat{\mathcal{A}} = \{\hat{s}_1, \dots, \hat{s}_m\}$ and take $\hat{S} = \hat{\mathcal{A}} \cup \{\hat{s}_{m+1}\}$ as the finite state-space of the MC ($\hat{s}_{m+1}$ is an absorbing state, as explained in the text)
3. define abstraction function as $f(\mathbf{z}) = \hat{s}_i$ if $\mathbf{z} \in \mathcal{A}_i$ for $i \in \{1, \dots, m\}$ and $f(\mathbf{z}) = s_{m+1}$ if $\mathbf{z} \in \mathbb{R}^n \setminus \mathcal{A}$
4. for each $i \in \{1, \dots, m\}$, select a single representative point $\mathbf{z}_i \in \mathcal{A}_i$
5. compute the transition probability matrix $\hat{T}$ as

$$\hat{T}(\hat{s}_i, \hat{s}_j) = \begin{cases} \int_{f^{-1}(\hat{s}_j)} t_z(\overline{\mathbf{z}}|\mathbf{z}_i) d\overline{\mathbf{z}}, & \text{if } 1 \leq j \leq m \text{ and } 1 \leq i \leq m, \\ 1 - \sum_{\bar{\hat{s}} \in \mathcal{A}_p} \int_{f^{-1}(\bar{\hat{s}})} t_z(\overline{\mathbf{z}}|\mathbf{z}_i) d\overline{\mathbf{z}}, & \text{if } j = m+1 \text{ and } 1 \leq i \leq m, \\ 1, & \text{if } j = i = m+1, \\ 0, & \text{if } 1 \leq j \leq m \text{ and } i = m+1, \end{cases}$$

Remark The bottleneck of Algorithm 6.1 lies in the computation of transition probability matrix $\hat{T}$, due to the integration of kernel t_z. This integration can be circumvented if the distribution functions $T_{ij}(\cdot)$ for all $i, j \in \{1, \dots, n\}$ have an explicit analytical form, e.g. an exponential distribution.

The procedure in Algorithm 6.1 has been shown [1] to introduce an approximate probabilistic bisimulation of the concrete model [49], i.e. the SMPL systems (6.1).

Algorithm 6.1 can be applied to abstract an SMPL system as a finite-state MC, regardless of the particular invariant set $\mathcal{A}$. However the quantification of the abstraction error in Section 6.4 requires that the invariant set $\mathcal{A}$ is bounded. □

Considering the obtained finite-state, discrete-time MC $(\hat{S}, \hat{T})$ and the discretized invariant set $\hat{\mathcal{A}} \subset \hat{S}$, the probabilistic invariance problem amounts to evaluating the probability that a finite execution associated with the initial condition $\hat{s}_0 \in \hat{S}$ remains within the abstract invariant set $\hat{\mathcal{A}}$ during the given event horizon. This can be stated as the following probability:

$$\hat{P}_{\hat{s}_0}(\hat{\mathcal{A}}) = \Pr\{\hat{s}(k) \in \hat{\mathcal{A}} \text{ for } k = 0, \ldots, N | \hat{s}(0) = \hat{s}_0\},$$

where $\hat{s}(k)$ denotes the discrete state of the MC at step k.

The solution of this finite-horizon probabilistic invariance problem over the MC abstraction can be determined via a discrete version of Proposition 6.1.

Proposition 6.2 Consider value functions $\hat{V}_k : \hat{S} \to [0,1]$, for $k \in \{0, \ldots, N\}$, computed through the following backward recursion:

$$\hat{V}_k(\hat{s}) = \mathbb{I}_{\hat{\mathcal{A}}}(\hat{s}) \sum_{\bar{\hat{s}} \in \hat{S}} \hat{V}_{k+1}\left(\bar{\hat{s}}\right) \hat{T}\left(\hat{s}, \bar{\hat{s}}\right), \qquad \text{for all } \hat{s} \in \hat{S},$$

initialized with $\hat{V}_N(\hat{s}) = \mathbb{I}_{\hat{\mathcal{A}}}(\hat{s})$ for all $\hat{s} \in \hat{S}$. Then $\hat{P}_{\hat{s}_0}(\hat{\mathcal{A}}) = \hat{V}_0(\hat{s}_0)$. □

For any $k \in \{0, \ldots, N\}$, notice that $\hat{V}_k(\hat{s})$ represents the probability that an execution of the finite-state MC remains within the discrete invariant set $\hat{\mathcal{A}}$ over the residual event horizon $\{k, \ldots, N\}$, starting from $\hat{s}$ at event step k.

The quantities in Proposition 6.2 can be easily computed by linear algebra. It is of interest to provide a quantitative comparison between the discrete outcome obtained by Proposition 6.2 and the continuous solution that results from Proposition 6.1: in other words, we are interested in deriving bounds on the abstraction error. The following section accomplishes this goal.

6.4 Quantification of the Abstraction Error

This section starts by precisely defining the error related to the abstraction procedure, which is due to the approximation of a continuous concrete model with a finite discrete one. Then a bound of the approximation error in [52] is recalled, and applied to the probabilistic invariance problem under some structural assumptions, namely in the case of Lipschitz continuous density functions, or alternatively piecewise Lipschitz continuous density functions.

The approximation error is defined as the maximum difference between the outcomes obtained by Propositions 6.1 and 6.2 for any pair of initial conditions $\mathbf{z}_0 \in \mathbb{R}^n$ and $f(\mathbf{z}_0) \in \hat{S}$. Since an exact computation of this error is not possible in general, we resort to determining an upper bound of the approximation error, which is denoted as E. More formally we are interested in determining E that satisfies

$$|P_{\mathbf{z}_0}(\mathcal{A}) - \hat{P}_{f(\mathbf{z}_0)}(\hat{\mathcal{A}})| \le E, \qquad \text{for all } \mathbf{z}_0 \in \mathcal{A}. \tag{6.2}$$

We raise the following assumption on the SMPL system. Recall that the density function of $A_{ij}(k) \otimes d_{ij}$ in (6.1) corresponds to the density function of $A_{ij}(k)$ in (2.9) shifted d_{ij} units forward.

Assumption 6.1 The density functions $t_{ij}(\cdot)$ for $i,j \in \{1,\dots,n\}$ are bounded:

$$t_{ij}(z) \leq M_{ij}, \qquad \text{for all } z \in \mathbb{R}. \qquad \square$$

Assumption 6.1 implies the distribution functions $T_{ij}(\cdot)$ for $i,j \in \{1,\dots,n\}$ are Lipschitz continuous. Recall that the (global) Lipschitz constant of a one-dimensional function can be computed as the maximum of the absolute value of the first derivative of the function. Thus

$$|T_{ij}(z) - T_{ij}(z')| \leq M_{ij}|z - z'|, \qquad \text{for all } z, z' \in \mathbb{R}.$$

For computation of the bound on approximation error, we use the following result based on [52], which has inspired most of this work.

Proposition 6.3 ([52, pp. 933-934]) Suppose Assumption 6.1 holds and the density function $t_z(\bar{\mathbf{z}}|\mathbf{z})$ satisfies the condition

$$\int_{\mathcal{A}} |t_z(\bar{\mathbf{z}}|\mathbf{z}) - t_z(\bar{\mathbf{z}}|\mathbf{z}')| d\bar{\mathbf{z}} \leq H\|\mathbf{z} - \mathbf{z}'\|_2, \qquad \text{for all } \mathbf{z}, \mathbf{z}' \in \mathcal{A},$$

then an upper bound on the approximation error in (6.2) is $E = NH\delta$, where N is the event horizon, δ is the partition diameter, H is a constant scalar. $\square$

The partition diameter δ in Proposition 6.3 is defined in [3, Sec. 3.1]. The notation $\|\cdot\|_2$ denotes the 2-norm operator. In the remainder of this subsection, we first determine the constant H for Lipschitz continuous density functions, then generalize the result to piecewise Lipschitz continuous density functions.

6.4.1 Lipschitz Continuous Density Functions

Assumption 6.2 The density functions $t_{ij}(\cdot)$ for $i,j \in \{1,\dots,n\}$ are Lipschitz continuous, namely there exist finite and positive constants h_{ij}, such that

$$|t_{ij}(z) - t_{ij}(z')| \leq h_{ij}|z - z'|, \quad \forall z, z' \in \mathbb{R}. \qquad \square$$

Under Assumptions 6.1 and 6.2, the conditional density function $t_z(\bar{\mathbf{z}}|\mathbf{z})$ is Lipschitz continuous. This opens up the application of the results in [3, 52] for the approximate solution of the probabilistic invariance problem. Notice that the Lipschitz constant of $t_z(\bar{\mathbf{z}}|\mathbf{z})$ may be large, which implies a rather conservative upper bound on the approximation error. To improve this bound, we can instead directly use Proposition 6.3 presented before – an option also discussed in [52]. In particular we present three technical lemmas that are essential for the computation of the constant H, with proofs appearing in the Appendix. After the derivation of the improved bound, the obtained results are applied to a numerical example.

Lemma 6.2 Any one-dimensional continuous distribution function $T(\cdot)$ satisfies the inequality

$$\int_{\mathbb{R}} |T(\bar{z} - z) - T(\bar{z} - z')| d\bar{z} \leq |z - z'|, \qquad \text{for all } z, z' \in \mathbb{R}. \qquad \square$$

Lemma 6.3 Suppose the random vector $\bar{\mathbf{z}}$ can be organized as $\bar{\mathbf{z}} = [\bar{\mathbf{z}}_1^T, \bar{\mathbf{z}}_2^T]^T$, so that its conditional density function is the multiplication of conditional density functions of $\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2$ as:

$$f(\bar{\mathbf{z}}|\mathbf{z}) = f_1(\bar{\mathbf{z}}_1|\mathbf{z}) f_2(\bar{\mathbf{z}}_2|\mathbf{z}).$$

Then for a given set $\mathcal{A} \in \mathcal{B}(\mathbb{R}^n)$[1] it holds that

$$\int_{\mathcal{A}} |f(\bar{\mathbf{z}}|\mathbf{z}) - f(\bar{\mathbf{z}}|\mathbf{z}')| d\bar{\mathbf{z}} \le \sum_{i=1}^{2} \int_{proj_i(\mathcal{A})} |f_i(\bar{\mathbf{z}}_i|\mathbf{z}) - f_i(\bar{\mathbf{z}}_i|\mathbf{z}')| d\bar{\mathbf{z}}_i,$$

where $proj_i(\cdot)$ represents the projection operator on the i-th axis. □

Lemma 6.4 Suppose the vector $\mathbf{z}$ can be organized as $\mathbf{z} = [\mathbf{z}_1^T, \mathbf{z}_2^T]^T$, and that the density function of the conditional random variable $(\bar{z}|\mathbf{z})$ is of the form

$$f(\bar{z}|\mathbf{z}) = f_1(\bar{z}, \mathbf{z}_1) f_2(\bar{z}, \mathbf{z}_2),$$

where $f_1(\bar{z}, \mathbf{z}_1), f_2(\bar{z}, \mathbf{z}_2)$ are bounded non-negative functions with $M_1 = \sup_{\bar{z}_1, \mathbf{z}_1} f_1(\bar{z}_1, \mathbf{z}_1)$ and $M_2 = \sup_{\bar{z}_1, \mathbf{z}_2} f_2(\bar{z}_1, \mathbf{z}_2)$. Then for a given set $\mathcal{C} \in \mathcal{B}(\mathbb{R})$:

$$\begin{aligned} &\int_{\mathcal{C}} |f(\bar{z}|\mathbf{z}_1, \mathbf{z}_2) - f(\bar{z}|\mathbf{z}_1', \mathbf{z}_2')| d\bar{z} \\ &\qquad \le M_2 \int_{\mathcal{C}} |f_1(\bar{z}, \mathbf{z}_1) - f_1(\bar{z}, \mathbf{z}_1')| d\bar{z} + M_1 \int_{\mathcal{C}} |f_2(\bar{z}, \mathbf{z}_2) - f_2(\bar{z}, \mathbf{z}_2')| d\bar{z}. \end{aligned}$$ □

Theorem 6.5 Under Assumptions 6.1 and 6.2, the constant H in Proposition 6.3 is

$$H = \sum_{i,j=1}^{n} H_{ij} + (n-1) M_{ij},$$

where $H_{ij} = \mathcal{L}_i h_{ij}$, and where the constant $\mathcal{L}_i = \mathcal{L}(proj_i(\mathcal{A}))$ is the Lebesgue measure of the projection of the invariant set onto the i-th axis. □

Proof Using Lemma 6.3 on the multiplicative structure of the conditional density function we have:

$$\int_{\mathcal{A}} |t_z(\bar{\mathbf{z}}|\mathbf{z}) - t_z(\bar{\mathbf{z}}|\mathbf{z}')| d\bar{\mathbf{z}} \le \sum_{i=1}^{n} \int_{proj_i(\mathcal{A})} |t_i(\bar{z}_i|\mathbf{z}) - t_i(\bar{z}_i|\mathbf{z}')| d\bar{z}_i,$$

and employing the triangle inequality for the additive structure of $t_i(\bar{z}_i|\mathbf{z})$ and utilizing Lemma 6.4 and Assumption 6.1 we obtain:

$$\begin{aligned} &\le \sum_{i,j=1}^{n} \int_{proj_i(\mathcal{A})} |t_{ij}(\bar{z}_i - d_{ij} - z_j) - t_{ij}(\bar{z}_i - d_{ij} - z_j')| d\bar{z}_i \\ &+ \sum_{i,j=1}^{n} \sum_{k=1, k\neq j}^{n} M_{ij} \int_{proj_i(\mathcal{A})} |T_{ik}(\bar{z}_i - d_{ik} - z_k) - T_{ik}(\bar{z}_i - d_{ik} - z_k')| d\bar{z}_i. \end{aligned}$$

---

[1] The notation $\mathcal{B}(\mathbb{R}^n)$ represents the collection of Borel sets that are a subset of $\mathbb{R}^n$.

Finally, by Assumption 6.2 and Lemma 6.2 we obtain

$$\leq \sum_{i,j=1}^{n} h_{ij}\mathcal{L}(proj_i(\mathcal{A}))|z_j - z'_j| + \sum_{i,j=1}^{n}\sum_{k=1,k\neq j}^{n} M_{ij}|z_k - z'_k|$$
$$\leq \left(\sum_{i,j=1}^{n} H_{ij} + (n-1)M_{ij}\right)\|\mathbf{z}-\mathbf{z}'\|_2 = H\|\mathbf{z}-\mathbf{z}'\|_2. \qquad \square$$

We now elucidate the above results on a case study, and select a beta distribution to characterize delays. A motivation for employing a beta distribution is that its density function has bounded support. Thus by scaling and shifting the density function, we can construct a distribution taking positive real values within an interval. Recall that this distribution is used to model processing or transportation times, and as such it can only take positive values. Furthermore, the beta distribution can be used to approximate the normal distribution with arbitrary accuracy.

Definition 6.1 (Beta Distribution) The general formula for the density function of the beta distribution is

$$t(x;\alpha,\beta,a,b) = \frac{(x-a)^{\alpha-1}(b-x)^{\beta-1}}{B(\alpha,\beta)(b-a)^{\alpha+\beta-1}}, \qquad \text{if } a \leq x \leq b,$$

and 0 otherwise, where $\alpha,\beta > 0$ are the shape parameters; $[a,b]$ is the support of the density function; and $B(\cdot,\cdot)$ is the beta function. A random variable X characterized by this distribution is denoted by $X \sim Beta(\alpha,\beta,a,b)$. $\square$

The case where $a = 0$ and $b = 1$ is called the standard beta distribution. Let us remark that the density function of the beta distribution is unbounded if any of the shape parameters belongs to the interval $(1,2)$. We remark that if the shape parameters are positive integers, the beta distribution has a piecewise polynomial density function, which has been used for system identification of SMPL systems in [55, Sec. 4.3].

Example We apply the results in Theorem 6.5 to the following two-dimensional SMPL system (2.9), where $A_{ij}(\cdot) \sim Beta(\alpha_{ij},\beta_{ij},a_{ij},b_{ij})$,

$$\begin{bmatrix}\alpha_{11} & \alpha_{12}\\ \alpha_{21} & \alpha_{22}\end{bmatrix} = \begin{bmatrix}2 & 4\\ 2 & 2\end{bmatrix}, \begin{bmatrix}\beta_{11} & \beta_{12}\\ \beta_{21} & \beta_{22}\end{bmatrix} = \begin{bmatrix}5 & 2\\ 2 & 4\end{bmatrix}, \begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix} = \begin{bmatrix}0 & 2\\ 2 & 0\end{bmatrix}, \begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{bmatrix} = \begin{bmatrix}7 & 6.5\\ 4 & 9\end{bmatrix}.$$

Skipping the details of the direct calculations, the supremum and the Lipschitz constant of the density functions are respectively

$$\begin{bmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{bmatrix} = \begin{bmatrix}1536/4375 & 15/32\\ 3/4 & 15/64\end{bmatrix}, \quad \begin{bmatrix}h_{11} & h_{12}\\ h_{21} & h_{22}\end{bmatrix} = \begin{bmatrix}30/49 & 80/81\\ 3/2 & 20/81\end{bmatrix}.$$

Considering a regular schedule with $\mathbf{s}(0) = [0,0]^T$ and $d = 4$, selecting invariant set $\mathcal{A} = \{\mathbf{z} \in \mathbb{R}^2 : -5 \leq z_1 \leq 5, -5 \leq z_2 \leq 5\}$, and event horizon $N = 5$, according to Theorem 6.5 we obtain an error $E = 176.4\delta$. In order to obtain an approximation error bounded by $E = 0.1$, we would need to discretize set $\mathcal{A}$ uniformly with 24942 bins for each dimension (step 1 of Algorithm 6.1). The obtained finite-state MC has $24942^2 + 1$ discrete states (step 2). The representative points have been selected at the center of the squares obtained by uniform discretization (step 4). The procedure to construct transition probability matrix (step 5) works as follows. For each $i,j \in \{1,\ldots,24942^2+1\}$, we compute $\hat{T}(\hat{s}_i,\hat{s}_j)$, which consists of four possible cases. If $1 \le i,j \le 24942^2$, then $\hat{T}(\hat{s}_i,\hat{s}_j)$ is defined as the probability of transitioning from the i-th representative point $\mathbf{z}_i$ to the j-th partition set $\mathcal{A}_j$. If $1 \le i \le 24942^2$ and $j = 24942^2+1$, then $\hat{T}(\hat{s}_i,\hat{s}_j)$ is defined as the probability of transitioning from the i-th representative point $\mathbf{z}_i$ to the complement of the invariant set $\mathbb{R}^n \setminus \mathcal{A}$. Since the discrete state $\hat{s}_{24942^2+1}$ is absorbing, then $\hat{T}(\hat{s}_{24942^2+1},\hat{s}_j) = 1$ if $j = 24942^2+1$, and it is equal to 0 otherwise. The solution of the invariance problem obtained over the abstract model (cf. Proposition 6.2) is computed via the software tool FAUST^2 [53] and is depicted in Fig. 6.2 (left). □

6.4.2 Piecewise Lipschitz Continuous Density Functions

It is clear that the structural assumptions raised in the previous section pose limitations on the applicability of the ensuing results. For the sake of generality, we now extend the previous results to the more general case encompassed by the following requirement.

Assumption 6.3 The density functions $t_{ij}(\cdot)$ for $i,j \in \{1,\ldots,n\}$ are piecewise Lipschitz continuous, namely there exist partitions $\mathbb{R} = \cup_{k=1}^{m_{ij}} D_{ij}^k$ and corresponding finite and positive constants h_{ij}^k, such that

$$\begin{aligned} & t_{ij}(z) = \sum_{k=1}^{m_{ij}} t_{ij}^k(z)\mathbb{I}_{D_{ij}^k}(z), && \text{for all } z \in \mathbb{R}, \\ & |t_{ij}^k(z) - t_{ij}^k(z')| \le h_{ij}^k |z - z'|, && \text{for all } k \in \{1,\ldots,m_{ij}\} \text{ and } z,z' \in D_{ij}^k. \end{aligned}$$ □

The notation k used in Assumption 6.3 is not a power and is not an event index (2.9), but it denotes the index of a set in the partition of cardinality $\sum_{i,j} m_{ij}$. Notice that if Assumption 6.3 holds and the density functions are Lipschitz continuous, then Assumption 6.2 is automatically satisfied with $h_{ij} = \max_k h_{ij}^k$. In other words, with Assumption 6.3 we allow relaxing Assumption 6.2 to hold only within arbitrary sets partitioning the state space of the SMPL system. In fact, we could limit the assumptions to the invariant set.

Under Assumptions 6.1 and 6.3, we now present a result extending Theorem 6.5 for the computation of the constant H.

Theorem 6.6 Under Assumptions 6.1 and 6.3, the constant H in Proposition 6.3 is

$$H = \sum_{i,j=1}^{n} H_{ij} + (n-1)M_{ij},$$

where $H_{ij} = \mathcal{L}_i \max_k h_{ij}^k + \sum_k |J_{ij}^k|$ and $\mathcal{L}_i = \mathcal{L}(proj_i(\mathcal{A}))$. The notation $J_{ij}^k = \lim_{z \downarrow c_{ij}^k} t_{ij}(z) - \lim_{z \uparrow c_{ij}^k} t_{ij}(z)$ denotes the jump distance of the density function $t_{ij}(\cdot)$ at the k-th discontinuity point c_{ij}^k.[2] □

---

[2] The jump distance is defined as the limit of $t_{ij}(z)$ as z approaches c_{ij}^k “from the right” subtracted by the limit of $t_{ij}(z)$ as z approaches c_{ij}^k “from the left.”

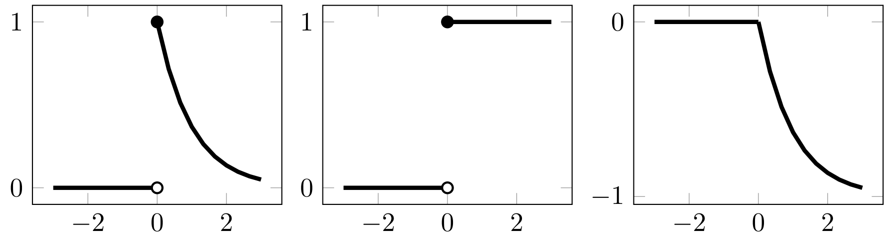

Fig. 6.1: (Left plot) The density function of exponential distribution with mean 1. (Middle plot) The unit step function $\theta(\cdot)$*. (Right plot) The continuous part of the function on the left plot.*

Proof We can follow exactly the same steps as in the proof of Theorem 6.5. The only difference is in the computation of constant H_{ij} for the inequality

$$\int_{proj_i(\mathcal{A})} |t_{ij}(\bar{z}_i - d_{ij} - z_j) - t_{ij}(\bar{z}_i - d_{ij} - z'_j)| d\bar{z}_i \leq H_{ij}|z_j - z'_j|, \tag{6.3}$$

for all $z_j, z'_j \in proj_j(\mathcal{A})$. We show that such a constant exists for piecewise Lipschitz continuous density functions and compute it based on Assumption 6.3. Define two functions $g^d_{ij}(z) = \sum_{k=1}^{m_{ij}-1} \mathcal{J}^k_{ij}\theta(z - c^k_{ij})$ and $g^c_{ij}(z) = t_{ij}(z) - g^d_{ij}(z)$, where $\mathcal{J}^k_{ij} = \sum_{q=1}^{k} J^q_{ij}$, $\theta(\cdot)$ denotes the unit step function, and $\{c^k_{ij} : k = 1, \dots, m_{ij} - 1\}$ are the discontinuity points of the density function $t_{ij}(\cdot)$. Then the density function is decomposed into $t_{ij}(z) = g^c_{ij}(z) + g^d_{ij}(z)$ where g^c_{ij} is its continuous part and g^d_{ij} is a piecewise constant function containing its jumps (cf. Fig. 6.1). It is clear that

$$\int_{proj_i(\mathcal{A})} |g^d_{ij}(\bar{z} - d_{ij} - z) - g^d_{ij}(\bar{z} - d_{ij} - z')| d\bar{z} \leq \sum_{k=1}^{m-1} |\mathcal{J}^k_{ij}||z - z'|,$$

$$\int_{proj_i(\mathcal{A})} |g^c_{ij}(\bar{z} - d_{ij} - z) - g^c_{ij}(\bar{z} - d_{ij} - z')| d\bar{z} \leq \mathcal{L}_i \max_k h^k_{ij}|z - z'|.$$

Adding both sides using the triangle inequality leads to the desired value for H_{ij}. □

We now display the obtained results with a numerical example.

Example Let us clarify the approach used in the proof of Theorem 6.6 on a simple example. Consider the density function of the exponential distribution with mean 1, i.e. $t(z) = \exp\{-z\}$ if $z \geq 0$ and 0 otherwise, as shown in Fig. 6.1 (left). Notice that the density function is piecewise Lipschitz continuous (cf. Assumption 6.3). Furthermore the density presents one discontinuity point $c^1 = 0$ with associated jump distance equal to $J^1 = 1$. Notice that $m = 2$ since $\mathbb{R}$ is partitioned into two sets. By using the formula in the proof, one gets $\mathcal{J}^1 = 1$ and furthermore the density function can be decomposed into a piecewise constant function $g^d(z) = \theta(z)$ and a continuous function $g^c(z) = 0\mathbb{I}_{z<0} + (\exp\{-z\} - 1)\mathbb{I}_{z\geq 0}$, as depicted in Fig. 6.1 (middle and right). □

In some cases, it is possible to obtain a smaller value for H_{ij} by substituting the density function directly into the inequality in (6.3). Furthermore H_{ij} may be independent of the size of the invariant set. For instance, if the delay is modeled by an exponential distribution as in page 13, then $A_{ij}(\cdot)$ for all $i,j \in \{1,\dots,n\}$ follows a shifted exponential distribution, i.e. $A_{ij}(\cdot) \sim SExp(\mu_{ij},\varsigma_{ij})$. In this case, $H_{ij} = \mu_{ij}^{-1} + \mu_{ij}^{-2}\mathcal{L}_i$, as per Theorem 6.6. However if we compute directly the left-hand side of (6.3), we get the quantity $H_{ij} = 2\mu_{ij}^{-1}$, which is independent of the shape of the invariant set. This fact is now proven in general, for a class of distribution functions, in Theorem 6.7. Let us first introduce the following definition.

Definition 6.2 (Shifted Exponential Distribution) The density function of an exponential distribution shifted by ς is given by

$$t(x;\mu,\varsigma) = \mu^{-1}\exp\{-\mu^{-1}(x-\varsigma)\}\theta(x-\varsigma),$$

where $\theta(\cdot)$ is the unit step function. A random variable X characterized by this distribution is denoted by $X \sim SExp(\mu,\varsigma)$. □

The proof of the following theorem can be found in the Appendix.

Theorem 6.7 Any random sequence $A_{ij}(\cdot) \sim SExp(\mu_{ij},\varsigma_{ij})$ satisfies inequality (6.3) with $H_{ij} = 2\mu_{ij}^{-1}$. □

Given the previous result, the bound related to the invariance-related abstraction error over SMPL systems with $A_{ij}(\cdot) \sim SExp(\mu_{ij},\varsigma_{ij})$ can be improved and explicitly shown as follows. The maximum value of the density function $t_{ij}(\cdot)$ equals μ_{ij}^{-1}, i.e. $M_{ij} = \mu_{ij}^{-1}$ for all $i,j \in \{1,\dots,n\}$. By Theorem 6.6 and Proposition 6.3, the bound of the approximation error is then

$$E = (n+1)N\delta\sum_{i,j}\mu_{ij}^{-1}.$$

Let us go back to the example in page 88 and adapt it according to Definition 6.2 and Theorem 6.7.

Example Consider the following two-dimensional SMPL system (2.9), where $A_{ij}(\cdot) \sim SExp(\mu_{ij},\varsigma_{ij})$ and

$$\begin{bmatrix}\mu_{11} & \mu_{12}\\ \mu_{21} & \mu_{22}\end{bmatrix} = \begin{bmatrix}2 & 3\\ 1 & 3\end{bmatrix}, \quad \begin{bmatrix}\varsigma_{11} & \varsigma_{12}\\ \varsigma_{21} & \varsigma_{22}\end{bmatrix} = \begin{bmatrix}0 & 2\\ 2 & 0\end{bmatrix}.$$

Considering a regular schedule with $\mathbf{s}(0) = [0,0]^T$ and $d = 4$, selecting invariant set $\mathcal{A} = \{\mathbf{z} \in \mathbb{R}^2 : -5 \le z_1 \le 5, -5 \le z_2 \le 5\}$, and event horizon $N = 5$, we get $E = 32.5\delta$. In order to obtain a desired error $E = 0.1$, we need to use 4597 bins for each dimension on a uniform discretization of the set $\mathcal{A}$. The solution of the invariance problem over the abstract model is presented in Fig. 6.2 (right).

Let us now validate this outcome. We have computed 1000 sample trajectories, with an initial condition that has been uniformly generated from the level set corresponding to the probability 0.3, namely within the set $\{\mathbf{z} : \hat{P}_{f(\mathbf{z})}(\hat{\mathcal{A}}) \ge 0.3\}$. Given the error bound $E = 0.1$, we would expect that the trajectories are invariant with a likelihood greater than 0.2. Among the cohort, we have found that 374 trajectories stay inside the invariant set for the given 5 steps, which is aligned with the guarantee we have derived.

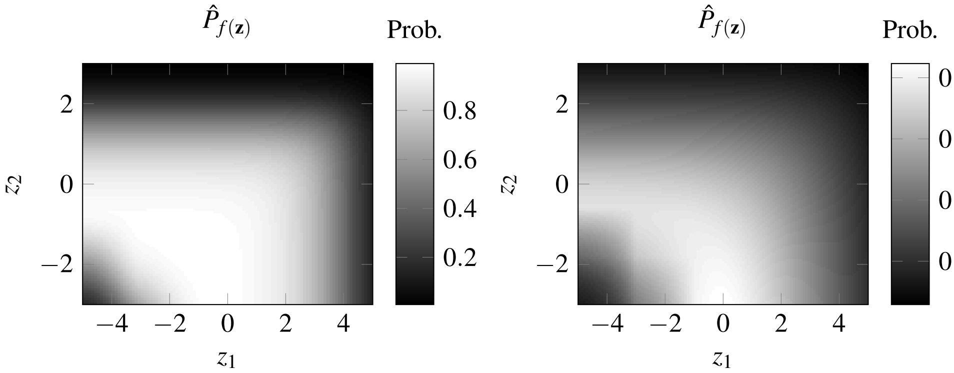

Fig. 6.2: The left and right plots show solution of the finite-horizon probabilistic invariance problem for two-dimensional SMPL systems with beta (cf. page 88) and exponential (cf. page 91) distributions, respectively. The plots have been obtained by computing the problem over finite abstractions obtained by uniform discretization of the set of interest and selection of central representative points.

Furthermore we have compared the approximate solution against the following empirical approach: for each representative point, we generate 1000 sample trajectories starting from it and compute the ratio of the number of trajectories that stay in the invariant set for 5 steps to the total number of trajectories (1000). The maximum absolute difference between the approximate solution and the empirical approach for all representative points is 0.0565, which aligns with the error bound of 0.1.

We have also performed these two comparisons for the SMPL system in page 88. The results are quite analogous to the ones obtained in this example. □

6.5 Summary

This chapter has employed finite abstractions to study the finite-horizon probabilistic invariance problem over Stochastic Max-Plus-Linear (SMPL) systems. We have assumed that each random variable has a fixed support, which implies that the topology of the SMPL system is fixed over time. Along this line, we are interested to relax this assumption in order to obtain results that are robust against small topological changes.

Appendix

Proof (of Lemma 6.2) We prove the inequality for the case $z' > z$. For the other case, the proof is similar. Consider any arbitrary $a, b \in \mathbb{R}$. Since the distribution function is non-decreasing we can write

$$\begin{aligned}\int_a^b |T(\bar{z}-z) - T(\bar{z}-z')|d\bar{z} &= \int_a^b T(\bar{z}-z)d\bar{z} - \int_a^b T(\bar{z}-z')d\bar{z}\\ &= g(z) - g(z'),\end{aligned}$$

where $g(z) = \int_a^b T(\overline{z}-z)d\overline{z} = \int_{a-z}^{b-z} T(u)du$. By using the fundamental theorem of calculus, we obtain

$$|g'(z)| = |T(a-z) - T(b-z)| \leq 1.$$

Finally based on the mean value theorem, we can write $|g(z)-g(z')| \leq |z-z'|$. The inequality holds for any interval $[a,b]$, then it also holds over $\mathbb{R}$. □

Proof (of Lemma 6.3) In the following derivation, we use the triangle inequality and the following properties of a density function: it is a positive function and its integral is bounded by one. Hence,

$$\begin{aligned}
&\int_{\mathcal{A}} |f(\overline{\mathbf{z}}|\mathbf{z}) - f(\overline{\mathbf{z}}|\mathbf{z}')|d\overline{\mathbf{z}} = \int_{\mathcal{A}} |f_1(\overline{\mathbf{z}}_1|\mathbf{z})f_2(\overline{\mathbf{z}}_2|\mathbf{z}) - f_1(\overline{\mathbf{z}}_1|\mathbf{z}')f_2(\overline{\mathbf{z}}_2|\mathbf{z}')|d\overline{\mathbf{z}} \\
&\leq \int_{\mathcal{A}} |f_1(\overline{\mathbf{z}}_1|\mathbf{z})f_2(\overline{\mathbf{z}}_2|\mathbf{z}) - f_1(\overline{\mathbf{z}}_1|\mathbf{z}')f_2(\overline{\mathbf{z}}_2|\mathbf{z})|d\overline{\mathbf{z}} \\
&+ \int_{\mathcal{A}} |f_1(\overline{\mathbf{z}}_1|\mathbf{z}')f_2(\overline{\mathbf{z}}_2|\mathbf{z}) - f_1(\overline{\mathbf{z}}_1|\mathbf{z}')f_2(\overline{\mathbf{z}}_2|\mathbf{z}')|d\overline{\mathbf{z}} \\
&= \int_{\mathcal{A}} |f_1(\overline{\mathbf{z}}_1|\mathbf{z}) - f_1(\overline{\mathbf{z}}_1|\mathbf{z}')|f_2(\overline{\mathbf{z}}_2|\mathbf{z})d\overline{\mathbf{z}} \\
&+ \int_{\mathcal{A}} |f_2(\overline{\mathbf{z}}_2|\mathbf{z}) - f_2(\overline{\mathbf{z}}_2|\mathbf{z}')|f_1(\overline{\mathbf{z}}_1|\mathbf{z}')d\overline{\mathbf{z}} \\
&\leq \int_{proj_1(\mathcal{A})} |f_1(\overline{\mathbf{z}}_1|\mathbf{z}) - f_1(\overline{\mathbf{z}}_1|\mathbf{z}')|d\overline{\mathbf{z}}_1 \int_{proj_2(\mathcal{A})} f_2(\overline{\mathbf{z}}_2|\mathbf{z})d\overline{\mathbf{z}}_2 \\
&+ \int_{proj_2(\mathcal{A})} |f_2(\overline{\mathbf{z}}_2|\mathbf{z}) - f_2(\overline{\mathbf{z}}_2|\mathbf{z}')|d\overline{\mathbf{z}}_2 \int_{proj_1(\mathcal{A})} f_1(\overline{\mathbf{z}}_1|\mathbf{z}')d\overline{\mathbf{z}}_1 \\
&\leq \int_{proj_1(\mathcal{A})} |f_1(\overline{\mathbf{z}}_1|\mathbf{z}) - f_1(\overline{\mathbf{z}}_1|\mathbf{z}')|d\overline{\mathbf{z}}_1 + \int_{proj_2(\mathcal{A})} |f_2(\overline{\mathbf{z}}_2|\mathbf{z}) - f_2(\overline{\mathbf{z}}_2|\mathbf{z}')|d\overline{\mathbf{z}}_2 \qquad \square
\end{aligned}$$

Proof (of Lemma 6.4) By using the triangle inequality, we obtain the following inequality:

$$\begin{aligned}
&\int_C |f(\overline{z}|\mathbf{z}_1,\mathbf{z}_2) - f(\overline{z}|\mathbf{z}_1',\mathbf{z}_2')|d\overline{z} = \int_C |f_1(\overline{z},\mathbf{z}_1)f_2(\overline{z},\mathbf{z}_2) - f_1(\overline{z},\mathbf{z}_1')f_2(\overline{z},\mathbf{z}_2')|d\overline{z} \\
&\leq \int_C |f_1(\overline{z},\mathbf{z}_1)f_2(\overline{z},\mathbf{z}_2) - f_1(\overline{z},\mathbf{z}_1')f_2(\overline{z},\mathbf{z}_2)|d\overline{z} \\
&+ \int_C |f_1(\overline{z},\mathbf{z}_1')f_2(\overline{z},\mathbf{z}_2) - f_1(\overline{z},\mathbf{z}_1')f_2(\overline{z},\mathbf{z}_2')|d\overline{z} \\
&\leq \int_C |f_1(\overline{z},\mathbf{z}_1) - f_1(\overline{z},\mathbf{z}_1')|f_2(\overline{z},\mathbf{z}_2)d\overline{z} + \int_C |f_2(\overline{z},\mathbf{z}_2) - f_2(\overline{z},\mathbf{z}_2')|f_1(\overline{z},\mathbf{z}_1')d\overline{z} \\
&\leq M_2 \int_C |f_1(\overline{z},\mathbf{z}_1) - f_1(\overline{z},\mathbf{z}_1')|d\overline{z} + M_1 \int_C |f_2(\overline{z},\mathbf{z}_2) - f_2(\overline{z},\mathbf{z}_2')|d\overline{z}. \qquad \square
\end{aligned}$$

Proof (of Theorem 6.7) We will show that the following inequality holds:

$$\int_{proj_i(\mathcal{A})} |t_{ij}(\overline{z_i} - d_{ij} - z_j;\mu_{ij},\varsigma_{ij}) - t_{ij}(\overline{z_i} - d_{ij} - z_j';\mu_{ij},\varsigma_{ij})|d\overline{z_i} \leq 2\mu_{ij}^{-1}|z_j - z_j'|,$$

for all $z_j, z_j' \in proj_j(\mathcal{A})$.

Without loss of generality, let us assume $z_j \leq z_j'$ (since the integrand and the expression on the right-hand side are symmetric w.r.t. z_j and z_j'). It follows that the integrand is a piecewise continuous function of $\overline{z_i}, z_j, z'_j$:

$$\begin{cases} \mu_{ij}^{-1}\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-z'_j-\varsigma_{ij})\}-\mu_{ij}^{-1}\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-z_j-\varsigma_{ij})\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{if } \overline{z_i}\geq z'_j+d_{ij}+\varsigma_{ij}, \\ \mu_{ij}^{-1}\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-z_j-\varsigma_{ij}), \quad \text{if } z_j+d_{ij}+\varsigma_{ij}\leq\overline{z_i}\leq z'_j+d_{ij}+\varsigma_{ij}, \\ 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{if } \overline{z_i}\leq z_j+d_{ij}+\varsigma_{ij}. \end{cases}$$

Thus the overall bounds can be computed based on the bounds of the first two subfunctions. We will prove that the first two subfunctions are bounded by $\mu_{ij}^{-1}|z_j-z'_j|$. Let us focus on the first subfunction:

$$\begin{aligned} &\mu_{ij}^{-1}\int_{z'_j+d_{ij}+\varsigma_{ij}}^{+\infty}\Big(\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-z'_j-\varsigma_{ij})\}-\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-z_j-\varsigma_{ij})\}\Big)\,d\overline{z_i} \\ &=\mu_{ij}^{-1}(\exp\{\mu_{ij}^{-1}z'_j\}-\exp\{\mu_{ij}^{-1}z_j\})\int_{z'_j+d_{ij}+\varsigma_{ij}}^{+\infty}\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-\varsigma_{ij})\}d\overline{z_i} \\ &=(\exp\{\mu_{ij}^{-1}z'_j\}-\exp\{\mu_{ij}^{-1}z_j\})\ \exp\{-\mu_{ij}^{-1}z'_j\} \\ &=1-\exp\{-\mu_{ij}^{-1}(z'_j-z_j)\} \\ &\leq\mu_{ij}^{-1}|z_j-z'_j|. \end{aligned}$$

The last inequality holds because $\mu_{ij}^{-1}(z'_j-z_j)\geq 0$ and $1-\exp\{-z\}\leq z$ for all $z\geq 0$. Then we continue to the second subfunction:

$$\begin{aligned} &\mu_{ij}^{-1}\int_{z_j+d_{ij}+\varsigma_{ij}}^{z'_j+d_{ij}+\varsigma_{ij}}\exp\{-\mu_{ij}^{-1}(\overline{z_i}-d_{ij}-z_j-\varsigma_{ij})\}d\overline{z_i} \\ &=-\exp\{-\mu_{ij}^{-1}(z'_j-z_j)\}+1 \\ &\leq\mu_{ij}^{-1}|z_j-z'_j|. \end{aligned}$$

□

Chapter 7

Conclusions and Future Research

In this thesis we have discussed finite abstractions of MPL systems, switching MiPL systems, and SMPL systems. Furthermore we have discussed reachability computations of MPL systems. We have applied the abstraction techniques to verify some properties of communication networks. In this chapter we summarize our main contributions and formulate future research directions.

7.1 Conclusions

Our main contributions are:

- **Formal verification of MPL systems.** In Chapter 3 we have designed a novel abstraction procedure applicable to autonomous and nonautonomous MPL systems. The finite abstraction has been proven to simulate the original MPL system. We have derived conditions under which the existence of a finite abstraction that bisimulates the original MPL system is guaranteed. Furthermore, we have devised a procedure to obtain such an abstraction. Compared to the broad existing literature in this area, this novel approach represents a brand new way of looking at general analysis of MPL models. Finally, the abstraction algorithms have been implemented and released in the VeriSiMPL tool.

- **Reachability computations of MPL systems.** Chapter 4 has discussed reachability computations of MPL systems, where the initial or final states are assumed to be expressed as unions of finitely many DBM. This work extends related results in the literature, since every max-plus polyhedron can be expressed as a union of finitely many DBM. The reachability algorithms have been efficiently implemented in the VeriSiMPL tool and shown to outperform alternative implementations.

- **Automatic verification of network properties.** In Chapter 5 we have discussed the automatic verification of backlog and virtual delay bounds for network calculus models via finite-state abstractions based on min-plus-linear (MiPL) models, thus elucidating an application of the theory developed above. First we have formulated a switching MiPL system from network calculus element. Then we have abstracted the switching MiPL system via its PWSA representation. The abstraction procedure is an extension of the one developed in Chapter 3. If we can verify a virtual delay or a backlog bound over the abstraction, this represents also a bound for the virtual delay or backlog of the switching MiPL system.

- **Finite abstractions of SMPL systems.** In Chapter 6 we have studied the finite-horizon probabilistic invariance problem over SMPL systems. The abstraction techniques are formal in the sense that they provide explicit error bounds. The error bound is defined as an upper bound on the maximum absolute difference between the exact solution and the one obtained from the abstraction. These results are distinguished from the existing literature on SMPL analysis and are computational since they leverage an existing software toolbox.

7.2 Recommendations for Future Research

In this section we discuss some interesting topics that can be considered for future research.

- **Specifications.** In Chapters 3 and 5 we have discussed an abstraction procedure for MPL systems and switching MiPL systems, respectively. Both abstraction procedures are formula based and preserve a wide range of LTL formulae. Considering other specifications such as Computation Tree Logic (CTL) [23, Def. 6.1], CTL* [23, Def. 6.80], and metric temporal logic [84] represents a first meaningful goal to extend our results. Similarly we are interested in extending the probabilistic invariance problem considered in Chapter 6 to more complex properties such as reach avoid or to general specifications expressed in Probabilistic CTL (PCTL) [23, Def. 10.36], PCTL* [23, Def. 10.59].

- **Max-plus polyhedra and polytopes.** It is clear that obtaining an abstract transition system with a smaller number of abstract states is desirable from a computational point of view. One way to achieve this is by using an abstraction procedure based on max-plus polyhedra [60], which has been studied under the names of semimodules [39] or idempotent spaces [88]. In this case there is no need to generate a PWA system and to refine the partition based on the affine dynamics. On top of that the computation of transitions will be faster because the dynamics are linear in the max-plus algebra. However there is an issue in partitioning the state space, which is $\mathbb{R}^n$, because max-plus polyhedra are necessarily closed. More precisely it is not possible to construct a non-trivial partition of $\mathbb{R}^n$ such that each block is a closed set.

  We are also interested in using polytopes in the abstraction procedure. Polytopes are more expressive than DBM, i.e. every DBM is a polytope. However the time complexity of many polytope operations is exponential.

- **Abstraction and verification techniques.** In Chapters 3 and 5 we restrict the specifications to LTL formulae. To check whether an abstraction satisfies an LTL formula, we use automata-based LTL model checking implemented in SPIN [71]. If we consider a specification expressed as a CTL formula, we can utilize symbolic model checking [35, Ch. 6], which is more efficient than the enumerative one [23, Sec. 6.4].

For SMPL systems, we have been exploring the existence of distributions associated to an analytical solution to the finite-horizon probabilistic invariance problem. This can be advantageous since the invariant set may be unbounded which limits the abstraction approach we have resorted to. Additionally this approach should not suffer from the curse of dimensionality since it does not explicitly employ a partitioning procedure.

- **Tools.** Currently VeriSiMPL is implemented in MATLAB. We are currently implementing tailored formula-based abstractions discussed above in the Java programming language so that VeriSiMPL can smoothly run on any platform. Furthermore we are planning to leverage symbolic model checking by using binary decision diagrams [114].

  The abstraction procedure for switching MiPL systems is currently implemented as a collection of MATLAB functions and scripts. These MATLAB files use some functionalities of VeriSiMPL. We are planning to integrate them with VeriSiMPL. Furthermore we will generalize the abstraction procedure to support the verification of any LTL formula, i.e. not restricted to verification of virtual delay and backlog bounds.

  Currently the procedure for computing the approximate solution of finite-horizon probabilistic invariance problem over SMPL systems is implemented in some MATLAB files. We are planning to integrate them with FAUST^2 [53]. FAUST^2 is a software tool that generates formal abstractions of (possibly non-deterministic) discrete-time Markov processes defined over uncountable (continuous) state spaces.

- **Models.** In this thesis we have extended the abstraction procedure for MPL systems to switching MiPL systems. There are some models that are related to MPL systems, such as (stochastic) MiPL systems, (stochastic) switching MPL systems [109, 110], stochastic switching MiPL systems, (stochastic) max-min-plus systems [72, 99], (stochastic) max-min-plus-scaling systems [97]. Improving the abstraction procedure to those models and looking towards extensions to new models is something we deem worth looking at.

- **Applications.** We have applied the abstraction techniques to verify some properties of communication networks. We are looking for some new applications for our techniques, such as optimal scheduling of multiple sheets in a printer [14], legged locomotion [89] and systems biology [28]. In the long run, we are also interested in large-scale applications such as railway network [68].

Bibliography

[1] A. Abate. Approximation metrics based on probabilistic bisimulations for general state-space Markov processes: a survey. *Electronic Notes in Theoretical Computer Sciences*, pages 3–25, 2014.

[2] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[3] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16(6):624–641, 2010.

[4] P.A. Abdulla and A. Nylén. Timed Petri Nets and BQOs. In José-Manuel Colom and Maciej Koutny, editors, *Applications and Theory of Petri Nets*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–70. Springer, Heidelberg, 2001.

[5] D. Adzkiya and A. Abate. VeriSiMPL: Verification via biSimulations of MPL models. In K. Joshi, M. Siegle, M. Stoelinga, and P.R. D'Argenio, editors, *Proc. 10th Int. Conf. Quantitative Evaluation of Systems (QEST'13)*, volume 8054 of *Lecture Notes in Computer Science*, pages 253–256. Springer, Heidelberg, September 2013. URL http://sourceforge.net/projects/verisimpl/.

[6] D. Adzkiya, B. De Schutter, and A. Abate. Abstraction and verification of autonomous max-plus-linear systems. In *Proc. 31st Amer. Control Conf. (ACC'12)*, pages 721–726, Montreal, CA, June 2012.

[7] D. Adzkiya, B. De Schutter, and A. Abate. Finite abstractions of nonautonomous max-plus-linear systems. In *Proc. 32nd Amer. Control Conf. (ACC'13)*, pages 4387–4392, Washington, DC, June 2013.

[8] D. Adzkiya, B. De Schutter, and A. Abate. Finite abstractions of max-plus-linear systems. *IEEE Trans. Autom. Control*, 58(12):3039–3053, December 2013.

[9] D. Adzkiya, B. De Schutter, and A. Abate. Computational techniques for reachability analysis of max-plus-linear systems. Technical report, Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, November 2013. Provisionally accepted for *Automatica*.

[10] D. Adzkiya, B. De Schutter, and A. Abate. Forward reachability computation for autonomous max-plus-linear systems. In E. Ábrahám and K. Havelund, editors, *Tools*

and Algorithms for the Construction and Analysis of Systems (TACAS'14), volume 8413 of *Lecture Notes in Computer Science*, pages 248–262. Springer, Heidelberg, 2014.

[11] D. Adzkiya, B. De Schutter, and A. Abate. Backward reachability of autonomous max-plus-linear systems. In *Proc. 12th Int. Workshop Discrete Event Systems*, pages 117–122, Cachan, May 2014.

[12] D. Adzkiya, S. Esmaeil Zadeh Soudjani, and A. Abate. Finite abstractions of stochastic max-plus-linear systems. In G. Norman and W. Sanders, editors, *Proc. 11th Int. Conf. Quantitative Evaluation of Systems (QEST'14)*, volume 8657 of *Lecture Notes in Computer Science*, pages 74–89. Springer International Publishing, 2014.

[13] M. Ahmane and L. Truffet. Idempotent versions of Haar's lemma: links between comparison of discrete event systems with different state spaces and control. *Kybernetika*, 43(3):369–391, 2007.

[14] M. Alirezaei, T.J.J. Van den Boom, and R. Babuška. Max-plus algebra for optimal scheduling of multiple sheets in a printer. In *Proc. 31st Amer. Control Conf. (ACC'12)*, pages 1973–1978, June 2012.

[15] X. Allamigeon, S. Gaubert, and É. Goubault. Inferring min and max invariants using max-plus polyhedra. In M. Alpuente and G. Vidal, editors, *Static Analysis*, volume 5079 of *Lecture Notes in Computer Science*, pages 189–204. Springer, Heidelberg, 2008.

[16] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[17] R. Alur, T. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proc. IEEE*, 88(7):971–984, July 2000.

[18] E. Asarin, G. Schneider, and S. Yovine. Algorithmic analysis of polygonal hybrid systems, part I: Reachability. *Theoretical Computer Science*, 379(12):231–265, 2007.

[19] E. Aydin Gol, M. Lazar, and C. Belta. Language-guided controller synthesis for linear systems. IEEE Trans. Autom. Control, 2014. to be published.

[20] H. Ayhan and F. Baccelli. Expansions for joint Laplace transform of stationary waiting times in (max,+)-linear systems with Poisson input. *Queueing Systems*, 37(1-3): 291–328, 2001.

[21] F. Baccelli and D. Hong. Analytic expansions of max-plus Lyapunov exponents. *The Annals of Applied Probability*, 10(3):779–827, August 2000.

[22] F. Baccelli, G. Cohen, G.J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity, An Algebra for Discrete Event Systems*. John Wiley and Sons, 1992.

[23] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[24] C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continous-time Markov chains. In J.C.M. Baeten and S. Mauw, editors, *Proc. 10th Int. Conf. Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–162. Springer, Heidelberg, 1999.

[25] G. Behrmann, A. David, and K.G. Larsen. A tutorial on UPPAAL. In M. Bernardo and F. Corradini, editors, *Formal Methods for the Design of Real-Time Systems (SFM-RT'04)*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, Heidelberg, September 2004.

[26] R.E. Bellman. On a routing problem. *Quart. Appl. Math.*, 16:87–90, 1958.

[27] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Autom. Control*, 45(10):1864–1876, October 2000.

[28] C. A. Brackley, D. S. Broomhead, M. C. Romano, and M. Thiel. A max-plus model of ribosome dynamics during mRNA translation. *Journal of Theoretical Biology*, 303 (0):128–140, June 2012.

[29] C.-S. Chang. *Performance Guarantees in Communication Networks*. Springer-Verlag, London, UK, 2000.

[30] P. Chaput, V. Danos, P. Panangaden, and G. Plotkin. Approximating Markov processes by averaging. *Journal of the ACM*, 61(1):5:1–5:45, January 2014.

[31] B. Charron-Bost, M. Függer, and T. Nowak. Transience bounds for distributed algorithms. In V. Braberman and L. Fribourg, editors, *Formal Modeling and Analysis of Timed Systems (FORMATS'13)*, volume 8053 of *Lecture Notes in Computer Science*, pages 77–90. Springer, Heidelberg, 2013.

[32] CheckMate [Online]. URL http://users.ece.cmu.edu/~krogh/checkmate/.

[33] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In N. Sharygina and H. Veith, editors, *Computer Aided Verification (CAV'13)*, volume 8044 of *Lecture Notes in Computer Science*, pages 258–263. Springer, Heidelberg, 2013.

[34] A. Chutinan and B.H. Krogh. Computational techniques for hybrid system verification. *IEEE Trans. Autom. Control*, 48(1):64–75, January 2003.

[35] E.M. Clarke, Jr., O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

[36] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. McGettrick, and J.-P. Quadrat. Numerical computation of spectral elements in max-plus algebra. In *Proc. IFAC Conf. Systems Structure and Control*, pages 699–706, Nantes, FR, July 1998.

[37] G. Cohen, D. Dubois, J.-P. Quadrat, and M. Viot. A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE Trans. Autom. Control*, 30(3):210–220, March 1985.

[38] G. Cohen, S. Gaubert, and J.-P. Quadrat. Max-plus algebra and system theory: Where we are and where to go now. *Annual Reviews in Control*, 23(0):207–219, 1999.

[39] G. Cohen, S. Gaubert, and J.-P. Quadrat. Duality and separation theorems in idempotent semimodules. *Linear Algebra and its Applications*, 379(0):395–422, 2004.

[40] R.A. Cuninghame-Green. *Minimax Algebra*, volume 166 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Germany, 1979.

[41] R.A. Cuninghame-Green. Minimax algebra and applications. *Fuzzy Sets and Systems*, 41(3):251–267, June 1991.

[42] B. Dahlan. Finite abstractions of network calculus elements for formal verification of network bounds. Master's thesis, Delft University of Technology, 2013.

[43] B. Dahlan, D. Adzkiya, A. Abate, and M. Mazo, Jr. Verification of properties for network calculus elements via finite abstractions. In *Proc. 53th IEEE Conf. Decision and Control (CDC'14)*, 2014. submitted.

[44] T. Dang and O. Maler. Reachability analysis via face lifting. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control (HSCC'98)*, volume 1386 of *Lecture Notes in Computer Science*, pages 96–109. Springer, Heidelberg, 1998.

[45] B. De Schutter. On the ultimate behavior of the sequence of consecutive powers of a matrix in the max-plus algebra. *Linear Algebra and its Applications*, 307(1-3): 103–117, March 2000.

[46] B. De Schutter and T.J.J. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, July 2001.

[47] R.E. de Vries. *On the Asymptotic Behavior of Discrete Events Systems*. PhD thesis, Delft University of Technology, 1992.

[48] J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, New York, NY, USA, 1995.

[49] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. *Information and Computation*, 179(2):163 – 193, 2002.

[50] M. Di Loreto, S. Gaubert, R. Katz, and J. Loiseau. Duality between invariant spaces for max-plus linear discrete event systems. *SIAM Journal on Control and Optimization*, 48(8):5606–5628, 2010.

[51] D.L. 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*, chapter 17, pages 197–212. Springer, Heidelberg, 1990.

[52] S. Esmaeil Zadeh Soudjani and A. Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956, 2013.

[53] S. Esmaeil Zadeh Soudjani, C. Gevaerts, and A. Abate. FAUST^2: Formal Abstractions of Uncountable-STate STochastic processes, 2014. URL http://sourceforge.net/projects/faust2/. arXiv:1403.3286.

[54] S. Farahani, T.J.J. van den Boom, H. van der Weide, and B. De Schutter. An approximation approach for model predictive control of stochastic max-plus linear systems. In *Proc. 10th Int. Workshop Discrete Event Systems*, pages 386–391, Berlin, DE, August/September 2010.

[55] S. Farahani, T.J.J. van den Boom, and B. De Schutter. Exact and approximate approaches to the identification of stochastic max-plus-linear systems. *Discrete Event Dynamic Systems: Theory and Applications*, pages 1–25, April 2013.

[56] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, June 1962.

[57] S. Gaubert and D. Hong. Series expansions of Lyapunov exponents and forgetful monoids. Technical Report RR-3971, INRIA, July 2000.

[58] S. Gaubert and R.D. Katz. Reachability and invariance problems in max-plus algebra. In L. Benvenuti, A. De Santis, and L. Farina, editors, *Positive Systems*, volume 294 of *Lecture Notes in Control and Information Science*, chapter 4, pages 15–22. Springer, Heidelberg, April 2003.

[59] S. Gaubert and R.D. Katz. Reachability problems for products of matrices in semirings. *International Journal of Algebra and Computation*, 16(3):603–627, 2006.

[60] S. Gaubert and R.D. Katz. The Minkowski theorem for max-plus convex sets. *Linear Algebra and its Applications*, 421(2-3):356–369, 2007.

[61] M.J. Gazarik and E.W. Kamen. Reachability and observability of linear systems over max-plus. *Kybernetika*, 35(1):2–12, 1999.

[62] R.M.P. Goverde, B. Heidergott, and G. Merlet. A coupling approach to estimating the Lyapunov exponent of stochastic max-plus linear systems. *European Journal of Operational Research*, 210(2):249–257, 2011.

[63] Z. Han and B.H. Krogh. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 287–301. Springer, Heidelberg, 2006.

[64] L. Hardouin, C.A. Maia, B. Cottenceau, and M. Lhommeau. Observer design for (max , +) linear systems. *IEEE Trans. Autom. Control*, 55(2):538–543, February 2010.

[65] M. Hartmann and C. Arguelles. Transience bounds for long walks. *Mathematics of Operations Research*, 24(2):414–439, May 1999.

[66] W. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[67] B. Heidergott. *Max-Plus Linear Stochastic Systems and Perturbation Analysis (The International Series on Discrete Event Dynamic Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, 2006.

[68] B. Heidergott, G.J. Olsder, and J.W. van der Woude. *Max Plus at Work–Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press, 2006.

[69] T.A. Henzinger and V. Rusu. Reachability verification for hybrid automata. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control (HSCC'98)*, volume 1386 of *Lecture Notes in Computer Science*, pages 190–204. Springer, Heidelberg, 1998.

[70] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. European Control Conf.*, pages 502–510, July 2013. URL http://control.ee.ethz.ch/~mpt.

[71] G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.

[72] A. Jean-Marie and G.J. Olsder. Analysis of stochastic min-max-plus systems: Results and conjectures. *Mathematical and Computer Modelling*, 23(11-12):175–189, 1996.

[73] M. Johansson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Autom. Control*, 43(4):555–559, April 1998.

[74] R. D. Katz. Max-plus (A,B)-invariant spaces and control of timed discrete-event systems. *IEEE Trans. Autom. Control*, 52(2):229–241, February 2007.

[75] T. A M Kevenaar and D.M.W. Leenaerts. A comparison of piecewise-linear model descriptions. *IEEE Trans. Circuits Syst. I*, 39(12):996–1004, December 1992.

[76] M. Kloetzer, C. Mahulea, C. Belta, and M. Silva. An automated framework for formal verification of timed continuous Petri nets. *IEEE Trans. Ind. Informat.*, 6(3): 460–471, 2010.

[77] K. Koutsoukos and D. Riley. Computational methods for reachability analysis of stochastic hybrid systems. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 377–391. Springer, Heidelberg, 2006.

[78] R. P. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton Series in Computer Science. Princeton University Press, 1994.

[79] A.A. Kurzhanskiy and P. Varaiya. Ellipsoidal toolbox. Technical report, EECS Department, University of California, Berkeley, May 2006.

[80] A.A. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Trans. Autom. Control*, 52(1):26–38, January 2007.

[81] H. J. Kushner and P.G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, New York, 2001.

[82] M. Kvasnica, P. Grieder, and M. Baotić. Multi-parametric toolbox (MPT), 2004. URL http://control.ee.ethz.ch/~mpt/.

[83] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In C. Palamidessi, editor, *Proc. 11th Int. Conf. Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 123–137. Springer, Heidelberg, 2000.

[84] Y. Lakhneche and J. Hooman. Reasoning about durations in metric temporal logic. In H. Langmaack, W.-P. Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, chapter 24, pages 488–510. Springer, Heidelberg, 1994.

[85] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, Heidelberg, 2001.

[86] C. Le Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification (CAV'09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554. Springer, Heidelberg, 2009.

[87] D.M.W. Leenaerts and W.M.G. van Bokhoven. *Piecewise Linear Modeling and Analysis*. Kluwer Academic Publishers, Boston, 1998.

[88] G.L. Litvinov, V.P. Maslov, and G.B. Shpiz. Idempotent functional analysis: An algebraic approach. *Mathematical Notes*, 69(5-6):696–729, 2001.

[89] G.A.D. Lopes, B. Kersbergen, T.J.J. van den Boom, B. De Schutter, and R. Babuška. Modeling and control of legged locomotion via switching max-plus models. IEEE Trans. Robot., 2014. to be published.

[90] C.A. Maia, C.R. Andrade, and L. Hardouin. On the control of max-plus linear system subject to state restriction. *Automatica*, 47(5):988–992, 2011.

[91] J. Mairesse. A graphical approach of the spectral theory in the (max,+) algebra. *IEEE Trans. Autom. Control*, 40(10):1783–1789, October 1995.

[92] G. Merlet. Cycle time of stochastic max-plus linear systems. *Electronic Journal of Probability*, 13(12):322–340, 2008.

[93] S. Mirzazad-Barijough and Ji-Woong Lee. Stability and transient performance of discrete-time piecewise affine systems. *IEEE Trans. Autom. Control*, 57(4):936–949, April 2012.

[94] I.M. Mitchell. Comparing forward and backward reachability as tools for safety analysis. In A. Bemporad, A. Bicchi, and G. Buttazzo, editors, *Hybrid Systems: Computation and Control (HSCC'07)*, volume 4416 of *Lecture Notes in Computer Science*, pages 428–443. Springer, Heidelberg, 2007.

[95] I.M. Mitchell, A.M. Bayen, and C.J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Autom. Control*, 50(7):947–957, July 2005.

[96] T. Murata. Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, April 1989.

[97] I. Necoara, B. De Schutter, T.J.J van den Boom, and H. Hellendoorn. Model predictive control for uncertain max-min-plus-scaling systems. *International Journal of Control*, 81(5):701–713, 2008.

[98] G. J. Olsder. Performance analysis of data-driven networks. In J. McCanny, J. McWhirter, and E. Swartzlander, Jr., editors, *Systolic Array Processors*, pages 33–41. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[99] G.J. Olsder. Eigenvalues of dynamic max-min systems. *Discrete Event Dynamic Systems*, 1(2):177–207, 1991.

[100] G.J. Olsder, J.A.C. Resing, R.E. De Vries, M.S. Keane, and G. Hooghiemstra. Discrete event systems with stochastic processing times. *IEEE Trans. Autom. Control*, 35(3):299–302, March 1990.

[101] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, December 2003.

[102] G. Pola and M.D. Di Benedetto. Symbolic models and control of discrete-time piecewise affine systems: An approximate simulation approach. *IEEE Trans. Autom. Control*, 59(1):175–180, January 2014.

[103] M. Prandini and J. Hu. Stochastic reachability: Theory and numerical approximation. In C.G. Cassandras and J. Lygeros, editors, *Stochastic hybrid systems*, Automation and Control Engineering Series 24, pages 107–138. Taylor & Francis Group/CRC Press, 2006.

[104] J.-M. Prou and E. Wagneur. Controllability in the max-algebra. *Kybernetika*, 35(1): 13–24, 1999.

[105] J.A.C. Resing, R.E. de Vries, G. Hooghiemstra, M.S. Keane, and G.J. Olsder. Asymptotic behavior of random discrete event systems. *Stochastic Processes and their Applications*, 36(2):195–216, December 1990.

[106] B.J.P. Roset, H. Nijmeijer, J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Event driven manufacturing systems as time domain control systems. In *Proc. 44th IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC'05)*, pages 446–451, December 2005.

[107] E. D. Sontag. Nonlinear regulation: The piecewise-linear approach. *IEEE Trans. Autom. Control*, 26(2):346–358, April 1981.

[108] Subiono and J.W. van der Woude. Power algorithms for (max,+)- and bipartite (min,max,+)-systems. *Discrete Event Dynamic Systems*, 10(4):369–389, 2000.

[109] T.J.J. van den Boom and B. De Schutter. Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice*, 14 (10):1199–1211, 2006.

[110] T.J.J. van den Boom and B. De Schutter. Modeling and control of switching max-plus-linear systems with random and deterministic switching. *Discrete Event Dynamic Systems*, 22(3):293–332, 2012.

[111] J. W. van der Woude and B. Heidergott. Asymptotic growth rate of stochastic max-plus systems that with a positive probability have a sunflower-like support. In *Proc. 8th Int. Workshop Discrete Event Systems*, pages 451–456, July 2006.

[112] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Coupling event domain and time domain models of manufacturing systems. In *Proc. 45th IEEE Conf. Decision and Control (CDC'06)*, pages 6068–6073, December 2006.

[113] L. Vandenberghe, B.L. de Moor, and J. Vandewalle. The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits. *IEEE Trans. Circuits Syst.*, 36(11):1382–1391, November 1989.

[114] I. Wegener. *Branching Programs and Binary Decision Diagrams - Theory and Applications*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[115] W.M. Wonham. *Linear multivariable control: A geometric approach*, volume 101 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1974.

[116] W.M. Wonham. *Linear multivariable control: A geometric approach*. Springer-Verlag, Berlin, 1979.

[117] B. Yordanov and C. Belta. Formal analysis of discrete-time piecewise affine systems. *IEEE Trans. Autom. Control*, 55(12):2834–2840, December 2010.

[118] B. Yordanov, J. Tůmová, I. Černá, J. Barnat, and C. Belta. Temporal logic control of discrete-time piecewise affine systems. *IEEE Trans. Autom. Control*, 57(6):1491–1504, June 2012.

[119] B. Yordanov, J. Tůmová, I. Černá, J. Barnat, and C. Belta. Formal analysis of piecewise affine systems through formula-guided refinement. *Automatica*, 49(1):261–266, 2013.

[120] K. Zimmermann. A general separation theorem in extremal algebras. *Ekonom.-Mat. Obzor*, 13(2):179–201, 1977.

Glossary

List of Symbols and Notations

Below follows a list of the most frequently used symbols and notations in this thesis.

$a, \mathbf{a}$	atomic proposition and set of atomic propositions resp.
$a, a^{(\cdot)}$	amount of data arrivals in a communication network
a	support parameter of the beta distribution
$A, \bar{A}$	state matrix of MPL systems and of augmented MPL systems resp.
$A(\cdot)$	state matrix of SMPL systems
$\mathbf{A}$	collection of matrices used in PWA systems
$\mathcal{A}, \hat{\mathcal{A}}$	concrete and abstract invariant set resp.
$A^{(\ell)}$	state matrix of switching MiPL systems in mode ℓ
Act	set of actions in transition systems
AP	set of atomic propositions in transition systems
b	burstiness parameter of an affine arrival curve
b	support parameter of the beta distribution
B	input matrix of MPL systems and backlog of a network
$\mathcal{B}(\cdot)$	collection of Borel sets
B_{max}, B'_{max}	backlog bounds
$B_1, B_2, B_3, \ldots$	blocks
$\mathbf{B}$	collection of matrices used in PWA systems
$B^{(\ell)}$	input matrix of switching MiPL systems in mode ℓ
BB	atomic proposition “the backlog is bounded by B'_{max}”
c	cyclicity of state matrix in autonomous MPL systems
c	rate parameter of a service curve and discontinuity point
$\mathcal{C}$	a Borel set
cf	generic operator yielding the canonical form of DBM
C	collection of states of transition systems
CI	atomic proposition “the input is conformant”
d	delay parameter of a service curve and virtual delay
d	time duration between consecutive events
D_{max}, D'_{max}	virtual delay bounds
DB	atomic proposition “the virtual delay is bounded by D'_{max}”

E	max-plus eigenspace and the bound of approximation error
f	abstraction function
$\mathbf{f} = (f_1, \ldots, f_n)$	tuple characterizing a block
$\mathbf{g} = (g_1, \ldots, g_n)$	tuple characterizing a region of PWA systems
g^c, g^d	continuous and discrete parts of a density function resp.
$\mathcal{G}$	precedence (or communication) graph
h	content of the leaky bucket and Lipschitz constant
H	a constant
I, I_f	initial states of concrete and abstract transition systems resp.
$\mathbb{I}$	indicator function
$Im, \overline{Im}$	image operator w.r.t. autonomous and nonautonomous models resp.
J	jump distance of a density function
k	discrete-event counter and index of partition sets
k_0	length of the transient part
$k_\emptyset(\mathbf{x})$	there is no point in $\mathbb{R}^n$ that can reach $\mathbf{x}$ in $k_\emptyset$ steps or more
L, L_f	labeling function of concrete and abstract transition systems resp.
m	dimension of the input space
m	cardinality of the partition of invariant set
M	upper bound of a density function
n	dimension of the state space
n_m	number of modes in switching MiPL systems
N	event horizon
$\mathbb{N}$	set of natural numbers, i.e. $\{1, 2, 3, \ldots\}$
O	big O notation
$proj$	orthogonal projection operator
$Paths$	set of all paths in transition systems
$Post$	operator yielding the direct successors in transition systems
Pr, $P_{\mathbf{z}_0}$, $\hat{P}_{\hat{s}_0}$	probability measure
Pre	operator yielding the direct predecessors in transition systems
$q_A, q_{\bar{A}}$	number of regions in PWA system generated by A and $\hat{A}$ resp.
q_k, q_{-k}	number of DBM in $\mathcal{X}_k$ and $\mathcal{X}_{-k}$ for $k \in \mathbb{N} \cup \{0\}$ resp.
r	rate parameter of an affine arrival curve
R	region of PWA systems and cumulative function of input flow
R^*	cumulative function of output flow
$\mathbf{R}$	collection of regions used in PWA systems

$\mathcal{R}$	binary relation
$\mathcal{R}_\Pi$	equivalence relation induced by Π
$\mathbb{R}, \mathbb{R}_\varepsilon, \mathbb{R}_\top$	set of real numbers, $\mathbb{R} \cup \{-\infty\}$, and $\mathbb{R} \cup \{+\infty\}$ resp.
$s, \hat{s}$	concrete and abstract states of transition systems resp.
$\mathbf{s} = [s_1, \ldots, s_n]^T$	regular schedule
$S, \hat{S}$	state space of concrete and abstract transition systems resp.
t	time and density function
T	distribution function
$\hat{T}$	transition probability matrix of MC abstraction
$Traces$	set of traces of transition systems
TS, TS_f	concrete and abstract transition systems resp.
$\mathbf{u} = [u_1, \ldots, u_m]^T$	input vector of MPL systems
U	input space of MPL systems
$\mathcal{U}, \mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_{-1}, \ldots$	set of possible inputs
$V, \hat{V}$	value functions
W	matrix used to define the tuple **f**
$\mathbf{x} = [x_1, \ldots, x_n]^T$	state vector of MPL systems
$\bar{\mathbf{x}} = [x_1, \ldots, u_m]^T$	state vector of augmented MPL systems
X	state space of MPL systems
$\mathcal{X}$	set of states in MPL systems and safe set
$\mathcal{X}_0$	set of initial and final conditions
$\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \ldots$	(forward) reach sets
$\mathcal{X}_{-1}, \mathcal{X}_{-2}, \mathcal{X}_{-3}, \ldots$	backward reach sets
$\mathbf{z} = [z_1, \ldots, z_n]^T$	delay w.r.t. regular schedule
α	scalar, arrival curve, and shape parameter of beta distribution
β	scalar, service curve, and shape parameter of beta distribution
γ	any arbitrary action variable
δ	grid size parameter
ε	zero (or neutral) element of the max-algebraic addition
θ	the unit step function
λ	max-plus eigenvalue
μ	mean of an exponential distribution
Π	partition of the state space
ς	shift parameter of an exponential distribution
τ	irrelevant action variable
φ	LTL formula
$\oplus, \oplus'$	max-algebraic and min-algebraic additions resp.
$\otimes, \otimes'$	max-algebraic and min-algebraic multiplication resp.
$\top$	zero (or neutral) element of min-algebraic addition

$\bowtie$	$\{>,<,\geq,\leq\}$
$\longrightarrow$	transition relation
$2^{\cdot}$	power-set operator
$\lvert\cdot\rvert$	number of elements of a set and absolute-value operator
$\neg,\wedge,\vee$	Boolean operators: not, and, or
$\bigcirc,\mathsf{U},\Box,\Diamond$	temporal operators: next, until, always, eventually
$\times$	cross product operator
$\circ$	function composition operator
$\lfloor x\rfloor$	smallest integer greater than or equal to x
ℓ	the mode of switching MiPL systems

List of Abbreviations

The following abbreviations are used in this thesis:

CTL	Computation Tree Logic
DBM	Difference Bound Matrices
LTL	Linear Temporal Logic
MiPL	Min Plus Linear
MPL	Max Plus Linear
PCTL	Probabilistic Computation Tree Logic
PWA	Piecewise Affine
PWSA	Piecewise Switched Affine
SMPL	Stochastic Max Plus Linear

Samenvatting

Abstracties van Max-Plus-Lineaire Systemen

Max-plus-lineaire (MPL) systemen vormen een klasse van discrete-gebeurtenissystemen met een continue toestandsruimte die de tijdstippen van de onderliggende opeenvolgende discrete gebeurtenissen karakteriseert. Dit soort systemen zijn geschikt voor het beschrijven van de synchronisatie van de tijdstippen van parallelle processen. MPL systemen worden gebruikt voor de analyse en planning van infrastructuurnetwerken zoals communicatie- en spoorwegsystemen, productie- en fabricageprocessen en biologische systemen. Stochastische max-plus-lineaire (SMPL) systemen vormen een uitbreiding van MPL systemen waarin de tijdsverschillen tussen opeenvolgende gebeurtenissen gekarakteriseerd worden door probabilistische grootheden. In vergelijking met MPL systemen zijn SMPL systemen realistischer voor praktische toepassingen zoals b.v. het modelleren van een spoorwegsysteem waarin de rijtijd van een trein afhankelijk is van het gedrag van de bestuurder, van weersomstandigheden of van het aantal passagiers op de stations.

Verificatie wordt gebruikt om vast te stellen of een gegeven systeem bepaalde eigenschappen bezit die in formules zijn uitgedrukt. Een voorbeeld hiervan is bereikbaarheidsanalyse (in het Engels: *reachability analysis*), wat een fundamenteel probleem is in het domein van formele methoden, systeemtheorie en prestatie- en betrouwbaarheidsanalyse. Bereikbaarheidsanalyse omhelst het bepalen of een bepaalde systeemtoestand haalbaar is vanuit bepaalde initiële systeemtoestanden.

Verificatietechnieken en -methoden voor systemen met een eindig aantal toestanden hebben in de afgelopen decennia brede aandacht gekregen en zijn sterk ontwikkeld. Indien een systeem echter een groot aantal of zelfs oneindig veel toestanden heeft, kunnen we dergelijke technieken in het algemeen niet direct toepassen. In dat geval is het nodig om abstractietechnieken te gebruiken om een specifiek model formeel om te zetten in een eindige abstractie daarvan. Deze abstractie kan vervolgens automatisch geverifieerd worden met behulp van resultaten uit de literatuur.

In dit proefschrift ontwikkelen we nieuwe abstractietechnieken voor MPL systemen en passen we deze toe in communicatienetwerken. Daarnaast behandelen we de bereikbaarheid van MPL systemen en abstractietechnieken voor SMPL systemen. In het onderstaande bespreken we kort de technieken die in dit proefschrift worden voorgesteld alsmede de toepassingen daarvan in communicatienetwerken.

- **Eindige abstracties van MPL systemen.** We beschouwen het volgende probleem: gegeven een MPL systeem en een specificatie, bepaal of het MPL systeem aan de specificatie voldoet. De specificatie wordt uitgedrukt als een formule in lineaire tijdslogica (LTL). We stellen enkele algoritmen voor om abstracties te genereren. Deze algoritmen maken gebruik van differentie-begrensde matrices (*difference-bound matrices* - DBM) voor de representatie van gebieden en van stuksgewijs affiene (*piecewise affine* - PWA) modellen voor de representatie van de MPL dynamica. Een DBM is een doorsnede van een eindig aantal halfruimtes die gekarakteriseerd worden door het verschil tussen twee variabelen. Deze aanpak maakt de studie mogelijk van algemene eigenschappen van het originele MPL systeem door – via model *checking* – equivalente logische specificaties van de abstractie te verifiëren. Meer specifiek tonen we aan dat indien de abstractie voldoet aan de specificatie, het MPL systeem ook aan deze specificatie voldoet.

- **Bereikbaarheid van MPL systemen.** We breiden de voor- en achterwaartse bereikbaarheidsberekeningen voor MPL systemen beschreven in de literatuur uit door een willekeurige verzameling van respectievelijk begin- en eindcondities te beschouwen. In beide gevallen zijn de systeemmatrices niet noodzakelijk max-plus-inverteerbaar. We gebruiken geoptimaliseerde datastructuren, zoals de bovengenoemde DBM, die rekenkundig gemakkelijk gemanipuleerd kunnen worden. We lichten de toepassing van bereikbaarheidsberekeningen toe bij de analyse van de veiligheid en het transitiegedrag van MPL systemen. Ten slotte zetten we de voorwaartse bereikbaarheidsberekeningen met succes af tegen een alternatieve aanpak die gebaseerd is op de vaak gebruikte *Multi Parametric Toolbox* (MPT) versie 2.

- **Automatische verificatie van netwerkeigenschappen.** We passen onze abstractietechnieken toe om de grenzen te verifiëren voor de *backlog* en voor de virtuele vertraging in een communicatienetwerk. Alhoewel zulke eigenschappen reeds geanalyseerd kunnen worden met behulp van netwerkanalyse (*network calculus*), ligt de kracht van onze aanpak in zowel zijn totaal geautomatiseerde aard als in het openen van de weg naar automatische verificatie van bepaalde communicatietopologiën. Voorbeelden hiervan zijn geaggregeerde stromen, waarmee netwerkanalyse niet gemakkelijk kan omgaan. Daarnaast maakt het gebruik van abstractiemethoden, zoals die worden voorgesteld voor de automatische synthese van regelsoftware, de gelijktijdige verificatie van regel- en communicatiesoftware mogelijk.

- **Eindige abstracties van SMPL systemen.** We onderzoeken het gebruik van eindige abstracties om probabilistische invariantieproblemen met een eindige horizon voor SMPL systemen op te lossen. Het probabilistische invariantieprobleem komt neer op het bepalen van de kans dat aan de invariantie-eigenschap voor elke toegestane beginconditie voldaan is. Invariantie-eigenschappen bestaan uit een voorwaarde op de toestanden en vereisen dat deze voorwaarde in probabilistische zin van kracht is op alle bereikbare toestanden. Omdat een analytische oplossing van dit probleem in het algemeen niet afgeleid kan worden, maken we gebruik van formele abstractietechnieken uit de literatuur om een (kwantificeerbare) benaderende oplossing te bepalen voor het probleem.

De in dit proefschrift ontwikkelde abstractie- en bereikbaarheidsalgoritmen voor MPL systemen zijn geïmplementeerd als MATLAB *software*, “*Verification via biSimulations of*

MPL models” (VeriSiMPL, zoals in “*very simple*”), die vrij beschikbaar is om te *downloaden* van http://www.sourceforge.net/projects/verisimpl/.

Dieky Adzkiya

Summary

Abstractions of Max-Plus-Linear Systems

Max-Plus-Linear (MPL) systems are a class of discrete-event systems with a continuous state space characterizing the timing of the underlying sequential discrete events. These systems are predisposed to describe the timing synchronization between interleaved processes. MPL systems are employed in the analysis and scheduling of infrastructure networks, such as communication and railway systems, production and manufacturing lines, or biological systems. As a natural extension, Stochastic Max-Plus-Linear (SMPL) systems are MPL systems where the delays between successive events are characterized by random quantities. In practical applications SMPL systems are more realistic than simple MPL ones: for instance in a model for a railway network, train running times depend on driver behavior, on weather conditions, and on passenger numbers at stations.

Verification is used to establish whether the system under consideration possesses certain properties expressed as formulae. As an example, reachability analysis is a fundamental problem in the area of formal methods, systems theory, and performance and dependability analysis. It is concerned with assessing whether a certain state of a system is attainable from given initial states of the system.

Verification techniques and tools for finite-state systems have been widely investigated and developed in the past decades. However, if the system has a large number of states or even infinitely many states, in general we cannot apply such techniques directly. In this case we need to employ abstraction techniques to formally relate a concrete model to a finite abstraction of it, which is then amenable to be automatically verified by the relevant results in the literature.

In this PhD thesis we develop novel abstraction techniques for MPL systems, and use them in an application to communication networks. Additionally we discuss reachability of MPL systems and abstraction techniques for SMPL systems. Next we provide a summary of the techniques proposed in this PhD thesis and the applications to communication networks:

- **Finite abstractions of MPL systems.** We consider the following problem: given an MPL system and a specification, we determine whether the MPL system satisfies the specification. The specification is expressed as a formula in Linear Temporal Logic (LTL). We propose some algorithms to generate abstractions. The algorithms utilize Difference-Bound Matrices (DBM) for the representation of regions and Piece-Wise Affine (PWA) models for the representation of the MPL dynamics. A DBM is an intersection of finitely many half-space representations that are characterized by the difference of two variables. This approach enables the study of general properties of the original MPL system by verifying (via model checking) equivalent logical specifications over the abstraction. More precisely we show that if the abstraction satisfies the specification, the MPL system also satisfies the specification.

- **Reachability of MPL systems.** We extend the forward and backward reachability computations of MPL systems in the literature by considering an arbitrary set of initial and final conditions, respectively. Furthermore in both cases, the system matrices do not necessarily have to be max-plus invertible. We employ optimized data structures, such as the DBM used in the abstraction procedure above, that are easy to manipulate computationally. We illustrate the application of reachability computations over safety and transient analysis of MPL systems. Finally we successfully benchmark the forward reachability computations against an alternative approach based on the well-developed Multi Parametric Toolbox (MPT) version 2.

- **Automatic verification of network properties.** We apply our abstraction techniques to verify bounds for backlog and virtual delay in a communication network. Although such properties can already be analyzed using network calculus tools, the virtue of our approach lies in its completely automated nature, and in opening the door to the automatic verification of certain communication topologies, e.g. flow aggregates, which network calculus cannot easily cope with. Furthermore, the use of abstraction approaches similar to those proposed for the automatic synthesis of control software, enables the simultaneous verification of control and communication software.

- **Finite abstractions of SMPL systems.** We investigate the use of finite abstractions to study finite-horizon probabilistic invariance problem over SMPL systems. The probabilistic invariance problem amounts to determining the probability of satisfying the invariance property for each allowable initial condition. Invariance properties are given by a condition on the states and require that the condition holds (in probability) over all the reachable states. In general an analytical solution of this problem cannot be derived, thus we leverage formal abstraction techniques in the literature to determine a (quantifiably) approximate solution of the problem.

The abstraction and reachability algorithms for MPL systems developed in this thesis have been implemented as a MATLAB software tool, “Verification via biSimulations of MPL models” (VeriSiMPL, as in “very simple”), which is freely available for download at http://www.sourceforge.net/projects/verisimpl/.

Dieky Adzkiya

Curriculum Vitae

Dieky Adzkiya was born on 17 May 1983 in Lamongan, Indonesia. He received the B.Sc. degree in September 2005 and the M.Sc. degree in October 2008, both in mathematics from the Sepuluh Nopember Institute of Technology, Surabaya, Indonesia. In June 2010, he started his PhD study at the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His PhD project was focused on providing abstraction techniques for max-plus-linear systems. In 2013, he spent 2 months as a visiting scholar at the Honeywell Prague Laboratory, Prague, Czech Republic. Since June 2014, he is a postdoc researcher at the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His research interests are in the analysis and verification of discrete-event systems and in their applications.