THE THEORY OF RECTANGULAR HYBRID AUTOMATA

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Peter William Kopke, Jr.

August 1996

THE THEORY OF

RECTANGULAR HYBRID AUTOMATA

Peter William Kopke, Jr., Ph.D.

Cornell University 1996

A *hybrid automaton* consists of a finite automaton interacting with a dynamical system. Hybrid automata are used to model embedded controllers and other systems that consist of interacting discrete and continuous components. A hybrid automaton is *rectangular* if each of its continuous variables x satisfies a nondeterministic differential equation of the form $a \leq \frac{dx}{dt} \leq b$, where a and b are rational constants. Rectangular hybrid automata are particularly useful for the analysis of communication protocols in which local clocks have bounded drift, and for the conservative approximation of systems with more complex continuous behavior.

We examine several verification problems on the class of rectangular hybrid automata, including reachability, temporal logic model checking, and controller synthesis. Both dense-time and discrete-time models are considered. We identify subclasses of rectangular hybrid automata for which these problems are decidable and give complexity analyses.

An investigation of the structural properties of rectangular hybrid automata is undertaken. One method for proving the decidability of verification problems on infinite-state systems is to find finite quotient systems on which analysis can proceed. Three state-space equivalence relations with strong connections to temporal logic are bisimilarity, similarity, and language equivalence. We characterize the quotient spaces of rectangular hybrid automata with respect to these equivalence relations.

Biographical Sketch

Peter William Kopke, Jr., was born on June 8, 1969. He received the Sc.B. degree in Mathematics from Brown University in May, 1991. In his five years at Cornell University, he was awarded three degrees in Computer Science: the M.Eng. degree in May 1992, the M.S. degree in May 1995, and the Ph.D. degree in August 1996.

This dissertation is dedicated to Karen Stephanie James

Acknowledgements

I thank Professor Thomas A. Henzinger for serving as my committee chairman and mentor. He taught me much about how to write mathematics and how to conduct research. He devoted much time to me and my concerns, and he always had a fresh idea. I thank Professor Anil Nerode, and the US Army Research Office, for employing me for the last two years.[1] I also thank Professor Nerode for guiding my research on the control of hybrid systems. I thank Professor Dexter Kozen for asking me to consider the case of discrete time, in which the theory of rectangular hybrid automata is so agreeable. I thank Dr. Howard Wong-Toi for many discussions on control and hybrid automata, and for his willingness to read my work carefully.

[1]This research was partially supported by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University, Contract DAAH04-94-G-0217.

Table of Contents

1 Introduction **1**

1.1 Historical Background 2

1.1.1 Model Checking 2

1.1.2 Hybrid Automata 5

1.1.3 Synthesis and Control 7

1.1.4 Simulation 10

1.1.5 Other Approaches to Hybrid Verification 12

1.2 This Dissertation 15

1.2.1 Summary 15

1.2.2 Attribution 18

1.2.3 Notation and Mathematical Conventions 18

2 Rectangular Hybrid Automata **20**

2.1 Transition Systems 20

2.1.1 Reachability and Effectiveness 20

2.1.2 Languages 24

2.1.3 Simulation and Bisimulation 24

2.1.4 Temporal Logic 28

2.1.5 Control 32

2.2 Rectangular Hybrid Automata 35

2.2.1 Syntax 35

2.2.2 Semantics 36

2.2.3 Graphical Display Language 39

2.2.4 Triangular Automata 40

2.2.5 Basic Definitions 41

3 Reachability: Decidability **43**

3.1 The Time-Abstract Transition System 43

3.2 Skewed-Clock Translation: Compact Case 47

3.3 Skewed-Clock Translation: General Case 54

3.4 Initialized Rectangular Automata 75

4 Reachability: Undecidability **77**
4.1 Uninitialized Automata 80
4.2 Non-Rectangular Automata 87

5 Linear Temporal Logic **94**
5.1 The Divergent Language 95
5.2 Compact Nondeterminism 98
5.3 Bounded Nondeterminism 102
5.4 LTL Model Checking 109

6 Simulation Equivalence **112**
6.1 2D 113
6.1.1 Bisimilarity 113
6.1.2 Similarity 114
6.2 3D and Beyond 123
6.2.1 Synchronous Analysis 123
6.2.2 Asynchronous Analysis 130

7 Language Equivalence **138**
7.1 Definitions 139
7.2 One-Sided Timed Automata 140
7.2.1 Upper-Bounded Timed Automata 143
7.2.2 Lower-Bounded Timed Automata 145
7.2.3 One-Sided Timed Automata 146
7.3 Sufficient Conditions 155
7.4 Necessary Conditions 157
7.5 Activity Cubes 163

8 Discrete Time **164**
8.1 The Discrete-Time Transition System 165
8.2 Reachability 168
8.2.1 Decidability 169
8.2.2 Undecidability 170
8.3 CTL* Model Checking 172
8.4 Controller Synthesis 173

9 Future Work **179**

Bibliography **181**

List of Figures

2.1 The transition system S_1 22
2.2 The simulation condition 25
2.3 Transition system S_2; Similarity quotient; Bisimilarity quotient . . 26
2.4 CTL* semantics 30
2.5 The initialized rectangular automaton $\hat{A}$ 40

3.1 Envelope of the activity $act(v)(a) = [k, k']$ 49
3.2 A rectangular automaton A 50
3.3 The multirate automaton with attractors N_A 51
3.4 The timed automaton with attractors T_{M_A} 51
3.5 $J \cap I \neq \emptyset$ iff $\phi_\ell \wedge \phi_u$ 64

4.1 Wrapping lemma: the skewed clock c retains its entry value upon exit 82
4.2 Proof of the Wrapping Lemma for $W = 4$, $s_1 = 1$ 82
4.3 Equality lemma: testing $c = d$ 83
4.4 Assignment lemma: performing the assignment $d := \frac{s_2}{s_1}c$ 83
4.5 Counter decrement: multiplying c by $\frac{s_1}{s_2}$ using the two-slope variable z 85
4.6 Counter increment: multiplying c by $\frac{s_2}{s_1}$ using the two-slope variable z 86
4.7 Doubling c using variable z taking slopes $0, s_1$ 87
4.8 Multiplying c by $\frac{s_1}{s_2}$ when $s_2 < 0 < s_1$ 88
4.9 Multiplying c by $\frac{s_2}{s_1}$ when $s_2 < 0 < s_1$ 88
4.10 Doubling c when $s_2 = -s_1$ 89
4.11 Halving c when $s_2 = -s_1$ 89
4.12 Multiplying by $s > 0$ with assignment updates and a skewed clock of slope s 91
4.13 Multiplying by $|s|$ with assignment updates and a skewed clock of slope $s < 0$ 92
4.14 Doubling with assignment updates and a memory cell 92
4.15 Doubling with triangular invariants and a skewed clock 93

4.16 Doubling with the triangular activity $1 \leq \dot{x} \leq \dot{y} \leq \dot{z} \leq 2$ 93

5.1 $L^{div}(C) \subsetneq L^{div}(M_C)$ 97
5.2 The need for time divergence 103

6.1 The only bisimulation is equality 114
6.2 Region equivalence on the plane 115
6.3 Skewed region equivalence $\equiv_{\mathrm{s}}^{reg}$, where $s_2 = 2s_1$ 116
6.4 $cone(\mathbf{y})$ splits the unit square into four pieces 117

7.1 Vectors $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ are one-sided region equivalent but not region equivalent 154

8.1 Incrementing x_i 171
8.2 Testing $x_i = 0$ 171
8.3 Decrementing x_i 172
8.4 Using a triangle to test $x_i = 0$ 172
8.5 Decrementing x_i using only nonnegative slopes 173

Chapter 1

Introduction

In toasters and airplanes, cameras and automobiles, digital computers are being used to control continuous processes.[1] Concomitant with the proliferation of computer technology is the need for formal verification techniques that can be used to find errors in computer-controlled devices. Computer-assisted techniques are the most useful: not only can they be used to check existing devices, but they can be integrated into computer-aided design systems. When verification tools are used during the design process, errors can be corrected before faulty physical devices are built.

We study *hybrid systems*—systems with both digital and analog components—and more specifically, the *rectangular hybrid automaton*, a mathematical model of hybrid systems. Our main focus is on the complexity of verification problems. We first consider the reachability problem:

[1] Indeed, the *ubiquitous computing* initiative aims at putting computer chips into most everyday devices.

Given a rectangular hybrid automaton A and a set of unsafe system states, determine whether A can ever enter an unsafe state.

Next we address the more general problem of verifying properties of rectangular hybrid automata expressed in temporal logic:

Given a rectangular hybrid automaton A and a formula ϕ of temporal logic, determine whether A satisfies ϕ.

To a lesser extent we consider the control problem:

Given a rectangular hybrid automaton A and a set of unsafe system states, find a feedback controller for A such that, in the closed-loop system, no unsafe state can be reached.

One method for proving the decidability of verification problems on infinite-state systems is to find finite quotient systems on which analysis can proceed. Three state-space equivalence relations with strong connections to temporal logic are bisimilarity, similarity, and language equivalence. We characterize the quotient spaces of rectangular hybrid automata with respect to these equivalence relations.

1.1 Historical Background

1.1.1 Model Checking

The classical work of Hoare on proving the correctness of sequential computer programs uses an essentially syntactic method [Hoa69]. The basic unit is the *Hoare triple*, which consists of a precondition, a program instruction, and a postcondition.

A triple is valid if whenever the program state satisfies the precondition, execution of the program instruction leads to a program state satisfying the postcondition. By assigning a valid triple to each program instruction, a *proof outline* is built, from which properties true during execution and at program termination can be inferred.

Model checking is a semantic approach to program verification. Rather than applying proof rules to program text, the denotation (or model) of a program is analyzed. Generally, the denotation of a program is a labeled graph, which may be infinite. When the model of a program is a finite graph, many verification problems are decidable. Thus the verification of such systems can proceed algorithmically and automatically, rather than by theorem-proving. The beauty of model checking is that many algorithms are essentially finite-state.

Hoare's assertional technique for proving safety properties was extended to parallel programs by Owicki and Gries [OG76]. To prove *liveness* properties (such as "every request is eventually satisfied") of concurrent systems, Pnueli introduced *temporal logic*, a modal logic that takes into account the time-dependence of events [Pnu77,Pnu81]. Temporal logic can express many common program requirements, such as termination, deadlock-freedom, invariance, and liveness. A more powerful system, the *μ-calculus*, which adds a least fixpoint operator to propositional logic, was studied by Pratt [Pra81] and Kozen [Koz83].

This early work concentrated on proof techniques for proving system correctness. Other investigations focused on the relative expressiveness of different temporal logics [GPSS80,Lam80,BMP83,Wol83,EH86] and the complexity of the satisfiability problem [ES84,EH85,LPZ85,SE89]. The most important idea was to consider the verification of *finite-state* systems, rather than infinite-state systems, such as the systems defined by traditional Turing Machines. It was quickly shown that the model-checking problems for various temporal logics were decidable on finite-state systems [CES86,SC85]. Moreover, for Computation Tree Logic (CTL), the model-checking problem was shown to be tractable (in fact, solvable in time linear in the size of the system and the size of the formula to be verified). The decision procedure was implemented, and systems with several hundred states could be verified in seconds [CES86]. This was the first automatic model checker for temporal logic.

The number of states of a concurrent system grows exponentially with the number of parallel system components. Therefore algorithms whose running times are linear in the number of states are too slow to be used on practical problems. In particular, one must avoid explicitly constructing the model of a program to be verified. Instead, sets of program states can be represented intensionally, as the characteristic sets of logical formulas. Verification algorithms can manipulate formulas to simulate operations on state sets. This technique is known as *symbolic model checking*. Symbolic model checking can be applied to the entire μ-calculus [BCM+92], and consequently to temporal logics as well. Using the *binary decision diagram* data structure of Bryant [Bry86], McMillan developed the symbolic model checker SMV for the μ-calculus which has verified (and found errors in) enormous systems [McM93,CGH+93] (in 1992 [BCM+92], systems with 10^{20} states were advertized; by now, the exponent of 20 has been left far behind).

1.1.2 Hybrid Automata

Arguably the most successful model of real-time systems is the timed automaton of Alur and Dill [AD94]. A *timed automaton* is a finite automaton equipped with a finite number of precise clock variables (variables x satisfying the differential equation $\frac{dx}{dt} = 1$). Timed automata make very good models for communication protocols in which local clocks are assumed not to drift [WZ92]. As we shall see, timed automata can also be used in the verification of systems with drifting clocks [OSY94,HW95,HKPV95]. The model checking of temporal logic is computable on timed automata [ACD90], though intractable (PSPACE-complete for CTL and linear temporal logic). It was soon realized that the symbolic approach could be of great value for model checking infinite-state systems. Henzinger et al. developed a symbolic model checking technique for timed automata [HNSY94]. Symbolic model checkers for timed automata include HyTech [Ho95,HHW95], Kronos [DOTY96], Veriti [Won94], UppAal [BLL+96], HSIS [Bal94], and Timed Cospan [AK96].

A natural generalization of the timed automaton is the *hybrid automaton*, in which the behavior of the continuous variables can satisfy arbitrary differential equations [ACHH93,NOSY93]. Since the behavior of hybrid automata can be arbitrarily complex, much research has been devoted to the identification of verification problems and subclasses of hybrid automata for which these problems are decidable [Moo90,NOSY93,BES93,KPSY93,MP93a,ACHH93,ACH93,AHH96,BER94a, BER94b,MV94,PV94,HK94,AFH94,BR95,LPY95,HH95,Hen95,HKPV95,HHK95, AMP95a]. This is the focus of Chapters 3-5 and 8 of this dissertation.

Symbolic model checking techniques have been extended to the class of *linear hybrid automata*—in which the first time derivatives of the continuous variables satisfy linear relationships—by Halbwachs et al. and Alur et al. [HRP94a,ACH+95, AHH96]. These algorithms are implemented in the model checker HyTech of Henzinger et al. [Ho95,HHW95] and the model checker Polka of Halbwachs et al. [HRP94b]. However, for linear hybrid automata, simple model checking problems are undecidable: even the reachability problem for linear hybrid automata is equivalent to the halting problem for Turing Machines. Thus the model checking "algorithms" are actually semidecision procedures at best. Nevertheless, the algorithms terminate on a wide variety of interesting examples.

A *rectangular hybrid automaton (RHA)* is a finite automaton equipped with a finite set of *drifting* clocks. Thus RHA are a natural generalization of timed automata, and a specialization of linear hybrid automata. The subclass of *initialized* RHA (in which, essentially, the bounds on the drift of each clock do not change) is of especial interest, because the reachability problem for initialized RHA is decidable. This was proven by Puri and Varaiya [PV94]. They show that when all variable tests are non-strict (i.e., of the form $x_i \leq c$ rather than $x_i < c$), that the dense-time reachability problem is reducible to the discrete-time reachability problem. In this dissertation, we extend this result from the finitary condition of reachability to the infinitary condition of divergent language emptiness, for initialized RHA with *bounded nondeterminism*. Bounded nondeterminism is essentially the restriction that all discontinuous jumps in continuous variable values be bounded a priori. As a corollary, we show that on this class of automata, the model checking problem for linear temporal logic with the fairness condition of time divergence can be solved in polynomial space.[2] We also study RHA in the context of discrete time. A rectangular hybrid automaton is *positive* if all slopes are nonnegative and variables are tested only against nonnegative constants. In discrete time, the reachability problem is decidable for positive RHA. Moreover, the model checking problem for a richer temporal logic (CTL*, which subsumes both linear temporal logic and CTL) is proven to be decidable in polynomial space.

1.1.3 Synthesis and Control

More difficult than model checking is the *synthesis problem*:

> *Given a partially specified system A and a requirement ϕ, complete the definition of A in such a way that ϕ is satisfied, or determine that no such completion exists.*

Notice that model checking is the special case in which the given system A is already completely specified. This problem has been studied for several decades by researchers in both computer science and control theory.

Church's Problem

In the context of computing, the basic question was posed by Church [Chu63]. Let Σ_0 and Σ_1 be finite alphabets, and let L be a regular subset of $(\Sigma_0 \times \Sigma_1)^\omega$. Consider the infinite game G in which two players, Players 0 and 1, take turns,

[2]Since the model checking problem for linear temporal logic on finite systems is already PSPACE-hard, it is PSPACE-complete for initialized RHA with bounded nondeterminism.

Player i picking a letter from Σ_i. If the resulting infinite sequence is an element of L, then Player 0 wins. Because the set L is regular, one of the players has a winning strategy. A *regular* strategy for G is a strategy based upon a finite automaton with output (Player i feeds the move of Player $1-i$ into the finite automaton M, and plays the letter that is output by M as his move). Church's Problem is to determine whether either player has a regular winning strategy, and if so, to synthesize a finite automaton realizing the strategy. It was proven by Büchi and Landweber that the winner of G always has a regular winning strategy, and that this strategy can be computed [BL69]. A simpler proof using tree automata was given by Rabin [Rab72].

Suppose we wish to determine whether there exists a program P which interacts with an "environment" E in such a way that the combined system satisfies a given requirement ϕ expressed in linear temporal logic (LTL). The interaction of the program with the environment can be considered to be a two-player game in which the object of the program is the satisfaction of ϕ. Call a program *admissible* if it implements a winning strategy for this game. Suppose further that the behavior of the environment can be specified by a finite automaton M. Let L be the set of pairs of infinite sequences (α, β) such that β is accepted by M and (α, β) satisfies ϕ. Because every property expressible in LTL is regular, the language L is regular. Therefore the solution to Church's problem can be used to to determine whether any admissible program exists. Moreover, if there exists an admissible program, then there exists a finite-state admissible program. This is the work of Rosner and Pnueli, who applied these ideas to synchronous, asynchronous, and distributed environments [PR89a,PR89b,PR90]. Further work along these lines, dealing with realizable specifications in the context of fairness, appears in [ALW89, AM94,Var95]. Wong-Toi and Dill used this approach to automatically synthesize processes and schedulers when the environment and admissible processes are timed automata (and therefore infinite-state) [WD90]. A different approach to synthesis based upon satisfiability testing appears in [CE81,Wol82].

Supervisory Control of Discrete Event Systems

Control theorists have studied finite (and infinite) automata under the rubric of *discrete event systems*. In the framework of Ramadge and Wonham [RW87b, WR87,RW87a,RW89], a *supervisor* for a finite automaton M consists of a finite automaton with output, where the output specifies a subset of events (letters, transition labels) that may be taken by M. Certain events, deemed *uncontrollable*, may not be prevented from occurring. The objective is to find a supervisor such that the closed-loop system is deadlock-free, and the finitary language of the closed-loop system is (1) a subset of a given language L, but (2) as large as possible. When M is deterministic and L is given by a deterministic finite automaton, this problem may be solved in polynomial time. Cieslak et al. considered this problem in the context of incomplete information [CDFV88].

Extensions of the Ramadge-Wonham approach to the control of the infinitary behavior of discrete event systems were made by Thistle and Wonham [TW94a, TW94b,Thi95], and Ramadge [Ram89]. The introduction of discrete time was made in [GR88,BH88,OW89,BW94]. Dense time was introduced by Wong-Toi and

Hoffmann in [WH91,HW92a], who synthesized supervisors for timed automata.

Symbolic methods for the computation of supervisors were developed by Hoffman and Wong-Toi [HW92b]. Maler et al. gave simple procedures for the symbolic computation of supervisors for the control of the infinitary behavior of discrete and timed systems, for specifications given as automaton acceptance conditions [MPS95,AMP95b]. In this formulation, a supervisor is a strategy for winning an infinite game played on the graph of system states. McNaughton shows how to calculate the winning strategy for any game defined by a Muller acceptance condition on a finite automaton [McN93]. His proof uses the later appearance record that Gurevich and Harrington used to complement tree automata [GH82].

In Chapter 8 of this dissertation, we solve a controller synthesis problem for rectangular automata in the discrete time domain.

1.1.4 Simulation

Two of the basic tools that we use to establish our results are the concepts of *simulation* [Mil71] and *bisimulation* [Par81], which are fundamental to the study of process algebra. A *simulation* is a relation $\preceq$ between the state spaces of two systems, such that whenever $q \preceq r$ and q can perform an action σ leading to a state q', r can perform action σ leading to a state r' such that $q' \preceq r'$. A *bisimulation* requires both $\preceq$ and $\preceq^{-1}$ to be simulations.

Simulation is a basic notion in refinement calculi [AL88,GSSL94,Sie95]. A "concrete" system C is considered to *implement* an "abstract" system A if the abstract system simulates the concrete system. A *behavior* is a sequence of system actions. Trivially, if C implements A, then every behavior of C is a behavior of A. More deep are theorems that play the role of the converse [AL88]. Here is one that can be simply stated: if every behavior of C is a behavior of A, then there exists a system B such that B simulates C and A backward-simulates B (roughly speaking, A backward-simulates B if A simulates B when the transition relations of the two systems are reversed) [LV95].

Simulation and bisimulation have deep connections to temporal logic. Consider the set of bisimulations between a given system A and itself. The union of all such bisimulations is again a bisimulation, which we denote by $\equiv_A^{bis}$. Browne et al. showed that for any two states q and r of A, $q \equiv_A^{bis} r$ iff q and r satisfy the same formulas of CTL (or CTL*). The union of all simulations on A is again a simulation, which we denote $\preceq_A$. Let $\equiv_S^{sim}$ be defined by $q \equiv_S^{sim} r$ iff $q \preceq_A r$ and $r \preceq_A q$. The equivalence relation $\equiv_A^{sim}$ is known as *similarity*. For similarity, the theorem analogous to the above holds for the universal fragment of CTL (or CTL*) [BBLS92,DGG93].

It follows that CTL model checking (resp., universal CTL model checking) can be performed on the quotient system $A/{\equiv_A^{bis}}$ (resp. $A/{\equiv_A^{sim}}$). These quotient systems can be computed symbolically for any transition system in which boolean operations and the predecessor operation for each action are computable [BFH90, DHW92,HHK95]. If the bisimilarity quotient system $A/{\equiv_A^{bis}}$ is finite, then supervisor synthesis can be computed symbolically with a guarantee of termination [Hen95]. We show that the same holds if the similarity quotient system $A/{\equiv_A^{sim}}$ is finite, provided that a simple technical condition holds. We show that every two-dimensional rectangular hybrid automata with uniform dynamics has finite similarity quotient, and that consequently controller synthesis in dense time is computable for this class of automata.

1.1.5 Other Approaches to Hybrid Verification

Real-Time Temporal Logics

Various extensions to temporal logic have been proposed for the specification of real-time requirements. One natural approach is based upon the use of a flexible clock variable T, giving the global system time, and first-order quantification for rigid variables over the time domain [PdR82]. An approach advocated by Koymans, Vytopil, and de Roever (and many others) is the introduction of new temporal operators which refer to time explicitly [KVdR83,KdR85,Koy90]. One of the more useful logics of this sort is MITL, for which the model checking problem is decidable on timed automata [AFH96]. Another variant of this approach is the age operator Γ of Manna and Pnueli [MP93b]. The interpretation of $\Gamma(\phi)$ is the length of time that ϕ has remained continuously true. A fragment of a monadic second-order logic that is expressively equivalent to timed automata was discovered by Wilke [Wil94]. This fragment subsumes both MITL and propositional linear temporal logic augmented with the age operator, and can be used to prove the decidability of satisfiability for these logics. Another approach to real-time specification uses multiple flexible real-time specification variables in a propositional context [ACD93,HNSY94]. For purposes of automatic verification, this is the most useful approach: the KRONOS system verifies the logic TCTL (an extension of CTL with specification clock variables) on timed automata.

Deductive Approaches

Schneider et al. augment proof outlines with timing and other information about continuous variables [SBM92,FS94]. Hooman developed another extension of Hoare logic for the specification and derivation of correct real-time programs [Hoo91, Hoo94]. The *duration calculus* of Zhou et al. provides an interval-based temporal logic for specifying system properties and a proof calculus for proving system properties [ZHR91]. Duration properties such as "the gas burner leaks no more than five minutes out of any given hour" are easily expressible in the duration calculus. Bouajjani et al. provide a translation from a subclass of the duration calculus into linear hybrid automata [BLR95]. Abadi and Lamport [AL92] extend Lamport's Temporal Logic of Actions (TLA) [Lam91] to real time using a global clock as in [PdR82] above. TLA can also be used to reason about hybrid systems [Lam93]. In both the duration calculus and TLA, systems and properties are expressed in the same formal language. It follows that in either formalism, proving that a system satisfies its requirement reduces to proving the validity of an implication. Lynch et al. introduced the formalism of *I/O automata*, graphs in which edge labels model communication, and developed a simulation-based verification methodology [LT87, LV95]. Real time was introduced into the model in [LV92,GSSL94], and recently, more general continuous behavior has been considered [LSVW96]. To admit compositional reasoning, a notion of environment-freedom is introduced, which bars an I/O automaton from constraining its environment. Environment-freedom is defined in terms of a game between an I/O automaton and its environment. Thus games are useful for verification as well as for synthesis.

Methods from Control Theory

Isaacs inaugurated the study of *differential games*, in which two players control continuous variables to some end—often there is a pursuit motif, one player attempting to intercept the other [Isa65]. Pontryagin et al. further developed this theory, and called it *optimal control* [PBGM62]. Given a continuous system taking a continuous control and a cost function f on system trajectories, the basic problem of optimal control is to find a control function such that the trajectory of the controlled system minimizes f.

While many physical systems can be profitably modeled as linear (the control of these systems being well understood), many cannot. The solution of these inherently nonlinear problems seems to require switching, that is, adequate controllers require a discrete structure as well as a continuous structure. Such hybrid controllers (in the discrete time domain) were investigated by Sontag in [Son81].

An approach to hybrid controller design based upon relaxed variational methods has been developed by Nerode, Kohn, et al. [NK93a,NK93b,GKN94,KNRY95, KNR96]. Instead of obtaining the optimal control for a given cost function, an approximately optimal solution (within a given tolerance of the optimal) is obtained as a chattering approximation to the optimal control. The resulting controller is a finite automaton which samples the continuous system state at a fixed rate, and supplies control based upon its measurement.

1.2 This Dissertation

The objective of the author has been to provide a comprehensive account of the theory of rectangular hybrid automata as it now stands.[3] To meet this objective, two technical chapters containing previous work have been required. Both dense and discrete time domains are considered, though the bulk of the work (Chapters 3-7) assumes a dense time domain.

1.2.1 Summary

Chapter 2 contains basic material on transition systems, simulation and bisimulation, temporal logic, and control, as well as the definition of rectangular hybrid automata.

Chapter 3 contains the proof of Henzinger et al. [HPV94] of the theorem of Puri and Varaiya [PV94], which states that the reachability problem is decidable for the class of initialized RHA. The proof is based on a translation from initialized RHA into timed automata.

Chapter 4 consists of several undecidability results, which together show that the class of initialized RHA is a maximal class for which the reachability problem is decidable. When either rectangularity or initialization is relaxed, the reachability

[3] We have not treated the work of Henzinger and Ho [HH95] or Puri et al. [PBV96] on the approximation of nonlinear hybrid automata by rectangular hybrid automata.

problem becomes undecidable for automata in which only one continuous variable is not a precise clock.

Chapter 5 is concerned with the model checking problem for linear temporal logic under the fairness condition of time divergence. The condition of bounded nondeterminism is identified, and it is shown that the translation from rectangular hybrid automata into timed automata is divergent language-preserving for the class of initialized RHA with bounded nondeterminism. It follows that the model checking problem for linear temporal logic under the fairness condition of time divergence is PSPACE-complete on this class of automata.

Chapter 6 is concerned with the bisimilarity and similarity quotient systems of RHA. It is easy to show that even in two dimensions, bisimilarity degenerates to equality on many RHA. However, every positive 2D RHA has finite similarity quotient. It follows that the control problem for invariance for 2D RHA is decidable in deterministic exponential time. Our most surprising result is Theorem 6.2.1, which states that in three or more dimensions, the only synchronous[4] simulation is the equality relation. In four or more dimensions, the asynchronous similarity quotient degenerates to equality.

Chapter 7 is concerned with the finitary language equivalence problem, which asks under what conditions two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ cannot be distinguished (in terms of reachability) by any rectangular automaton with a given fixed continuous dynamics. A sufficient condition for language equivalence may be derived from the translation from RHA into timed automata, using previously known theory

[4]We distinguish between synchronous semantics, in which time is treated as a visible action, and asynchronous semantics, in which time is invisible.

of timed automata. A more precise study of the class $\mathcal{C}$ of automata that are targets of the translation uncovers a special property. Every clock variable x of an automaton $T \in \mathcal{C}$ is either (1) only bounded from above by edge guards, or (2) only bounded from below by edge guards. The language-equivalence quotient for these *one-sided* timed automata is coarser than the language-equivalence quotient for the class of all timed automata by a multiplicative exponential factor. The upshot is a superior sufficient condition for language equivalence. Interestingly, language equivalence for one-sided timed automata coincides with asynchronous similarity; bisimilarity is finer. For a second time, the analysis of similarity provides results that may not be obtained using bisimilarity. The remainder of the chapter is devoted to finding necessary conditions for language equivalence.

Chapter 8 contains a study of RHA in the discrete time domain, in which the theory is not displeasing. Positive RHA have finite bisimilarity quotients, even when uninitialized. It follows that on the class of positive RHA in discrete time, the CTL* model checking problem is PSPACE-complete, and the control problem for invariance is EXPTIME-complete. When positivity or rectangularity is relaxed, the reachability problem becomes undecidable. In addition, the reachability problem is decidable for initialized RHA. Thus the two incomparable class of positive RHA and initialized RHA are both maximal classes for which the reachability problem is decidable in discrete time.

1.2.2 Attribution

The author is responsible for at most one theorem in Chapters 2-3. Thereafter, all unattributed theorems are due to the author and his collaborators. The main results of Chapters 4 and 5 appeared first in [HKPV95], or will appear in the journal version [HKPV] of that article. The work on two-dimensional rectangular hybrid automata in Chapter 6 appeared in [HHK95]. The case of three or more dimensions, as well as the study of one-sided timed automata in Chapter 7, appeared in [HK96]. The final section of Chapter 7, as well as the whole of Chapter 8, receive their first publication in this dissertation.

1.2.3 Notation and Mathematical Conventions

The set of all subsets of a set S is denoted by $\mathcal{P}(S)$. The set of infinite sequences of elements of S is denoted by S^ω, and the set of finite sequences (strings) of elements of S is denoted by S^*. The number of elements of a finite set S is denoted by $|S|$. The set of real numbers is denoted by $\mathbb{R}$, the set of rational numbers is denoted by $\mathbb{Q}$, the set $\{0, 1, 2, \ldots\}$ of natural numbers is denoted by $\mathbb{N}$, and the set of integers is denoted by $\mathbb{Z}$. Sequences and strings are usually denoted by symbols with a short line above, as in $\bar{\sigma}$. The elements of $\bar{\sigma}$ are $\sigma_0, \sigma_1, \ldots$ Occasionally, for ease of notation, we begin with σ_1. The length of a string $\bar{\sigma}$ is denoted by $|\bar{\sigma}|$.

When defining a symbol by an equation, we use the symbol $=_{\rm def}$. For example, $\mathbb{R}_{\geq 0} =_{\rm def} \{x \in \mathbb{R} \mid x \geq 0\}$, and similarly $\mathbb{N}_{>0} =_{\rm def} \{n \in \mathbb{N} \mid n > 0\}$. Technical terms are set in italic type when defined. For example, a *vector* is an element of $\mathbb{R}^n$, for some $n \in \mathbb{N}_{>0}$. Vectors are written in boldface type. For any given $n \in \mathbb{N}_{>0}$, $\mathbf{1}_n$ is the vector $(1, 1, \ldots, 1)$, and $\mathbf{0}_n$ is the vector $(0, 0, \ldots, 0)$; usually we omit the subscript n. Components of vectors are written in italic type. Thus if $\mathbf{x} \in \mathbb{R}^n$, then $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. When subscripts are applied to boldface characters, as in $\mathbf{x}_1, \mathbf{x}_2, \ldots$, each $\mathbf{x}_i$ is a vector. We also use superscripts to denote sequences of vectors, as in $\mathbf{x}^1, \mathbf{x}^2, \ldots$ Floors and ceilings of vectors are taken componentwise, so for $\mathbf{x} \in \mathbb{R}^n$, $\lfloor \mathbf{x} \rfloor = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_n \rfloor)$ and $\lceil \mathbf{x} \rceil = (\lceil x_1 \rceil, \lceil x_2 \rceil, \ldots, \lceil x_n \rceil)$.

We have occasion to use the λ notation for function specification. For example, $\lambda x.x^2$ is the function mapping each element in its domain (specified in the surrounding text) to its square. Given two relations $R \subset A \times B$ and $S \subset B \times C$,

$$S \circ R =_{\text{def}} \{(a, c) \in A \times C \mid \exists b \in B.\, (a, b) \in R \text{ and } (b, c) \in S\}.$$

The *domain* of a relation R is the set

$$\text{dom } R =_{\text{def}} \{a \mid \exists b.\, (a, b) \in R\}.$$

Given an equivalence relation $\sim$ on a set S, the set of equivalence classes of $\sim$ is denoted by $S/{\sim}$.

Chapter 2

Rectangular Hybrid Automata

This chapter contains basic material on transition systems, simulation, temporal logic, and controller synthesis, as well as the definition of rectangular hybrid automata.

2.1 Transition Systems

We use transition systems, which are graphs with labels on both the vertices and the edges, as our basic semantic model.

2.1.1 Reachability and Effectiveness

Definition 2.1.1 A *transition system* $S = (Q_S, \Sigma_S, \rightarrow_S, Q_S^0, \Pi_S, \models_S)$ consists of a set Q_S of *states*, a set Σ_S of *events*, a multiset $\rightarrow_S \subset Q_S \times \Sigma_S \times Q_S$ called the *transition relation*, a set $Q_S^0 \subset Q_S$ of *initial states*, a finite set Π_S of *atomic propositions*, and a *satisfaction relation* $\models_S \subset Q_S \times \Pi_S$. We omit the subscripts from the components of S whenever it will not cause confusion. We write $q \xrightarrow{\sigma} q'$ instead of $(q, \sigma, q') \in \, \rightarrow$, and $q \models \pi$ instead of $(q, \pi) \in \, \models$.

The transition system S is *finite* if Q is finite. A *region* is a subset of Q. We say S is *deadlock-free* if for every state $q \in Q$, there exist a state $q' \in Q$ and a label $\sigma \in \Sigma$ such that $q \xrightarrow{\sigma} q'$.

We say that the atomic proposition π *is true in state* q, or q *satisfies* π, if $q \models \pi$. We write $\Pi(q)$ for the set $\{\pi \in \Pi \mid q \models \pi\}$ of all atomic propositions true in state q. An *atomic formula* α is a boolean combination of atomic propositions. We extend the satisfaction relation to atomic formulas by induction: $q \models \neg\alpha$ iff it is not the case that $q \models \alpha$; $q \models \alpha_1 \vee \alpha_2$ iff $q \models \alpha_1$ or $q \models \alpha_2$; $q \models \alpha_1 \wedge \alpha_2$ iff $q \models \alpha_1$ and $q \models \alpha_2$. Define $[\![\alpha]\!] =_{\text{def}} \{q \in Q \mid q \models \alpha\}$. ■

Example 2.1.2 Consider the deadlock-free transition system

$$S_1 = (\mathbb{Z}, \{a, b\}, \rightarrow, \{0\}, \{\mathit{even}, \mathit{odd}, \mathit{div3}\}, \models)$$

depicted in Figure 2.1, where $m \xrightarrow{a} n$ iff m is odd and $m + 1 = n$; $m \xrightarrow{b} n$ iff m is even and $m + 2 = n$; $m \models \mathit{even}$ iff m is even; $m \models \mathit{odd}$ iff m is odd; and $m \models \mathit{div3}$ iff m is divisible by 3. It follows that $m \models \mathit{even} \wedge \mathit{div3}$ iff m is divisible by 6. ■

Definition 2.1.3 Let S be a transition system. For each event $\sigma \in \Sigma$, the σ-*predecessor operator* $\mathit{Pre}^{\sigma} \colon \mathcal{P}(Q) \rightarrow \mathcal{P}(Q)$ is defined by

$$\mathit{Pre}^{\sigma}(R) =_{\text{def}} \{q \in Q \mid \exists r \in R.\, q \xrightarrow{\sigma} r\}.$$

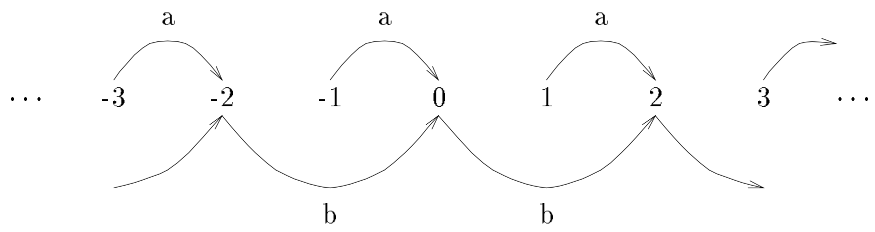

Figure 2.1: The transition system S_1

Define $Pre(R) =_{\rm def} \bigcup_{\sigma\in\Sigma} Pre^{\sigma}(R)$, and $Pre^*(R) =_{\rm def} \bigcup_{n\in\mathbb{N}} Pre^n(R)$.[1] The *$\sigma$-successor operator* $Post^{\sigma} : \mathcal{P}(Q) \to \mathcal{P}(Q)$ is defined analogously: $Post^{\sigma}(R) =_{\rm def} \{q \in Q \mid \exists r \in R.\, r \xrightarrow{\sigma} q\}$. Further define $Post(R) =_{\rm def} \bigcup_{\sigma\in\Sigma} Post^{\sigma}(R)$, and $Post^*(R) =_{\rm def} \bigcup_{n\in\mathbb{N}} Post^n(R)$. ■

Example 2.1.4 For the transition system S_1 from Example 2.1.2, $Pre^a([\![even]\!]) = [\![odd]\!]$, $Pre^b([\![even]\!]) = [\![even]\!]$, $Pre([\![even]\!]) = \mathbb{Z}$, and $Pre([\![odd]\!]) = Pre^a([\![odd]\!]) = Pre^b([\![odd]\!]) = \emptyset$. ■

Definition 2.1.5 Let S be a transition system. A region $R \subset Q$ is *reachable* if $Post^*(Q^0) \cap R \neq \emptyset$, or, equivalently, if $Pre^*(R) \cap Q^0 \neq \emptyset$. A state $q \in Q$ is *reachable* if the singleton $\{q\}$ is reachable. ■

Example 2.1.6 For the transition system S_1 from Example 2.1.2, the set $[\![odd]\!]$ is not reachable, while every nonnegative even integer is reachable. ■

Definition 2.1.7 The *reachability problem* for a class $\mathcal{C}$ of transition systems is stated in the following way:

[1]Here Pre^n denotes the n-fold composition of Pre, rather than the n-predecessor operator (treating n as an event). Numbers are indeed events for rectangular automata, but we trust that this dual usage of the superscript will not cause confusion.

Given a transition system $S \in \mathcal{C}$ and an atomic proposition $\pi \in \Pi$, determine whether $[\![\pi]\!]$ is reachable. ■

Definition 2.1.8 A *theory* $\mathcal{T}$ for a transition system S consists of a set $\mathcal{F} \supset \Pi$ of elements called *formulas*, a formula $\phi_0 \in \mathcal{F}$, and a map $[\![\cdot]\!]_{\mathcal{T}} \colon \mathcal{F} \to \mathcal{P}(Q)$ such that

1. for every atomic proposition $\pi \in \Pi$, $[\![\pi]\!]_{\mathcal{T}} = [\![\pi]\!]$,
2. for all formulas $\phi_1, \phi_2 \in \mathcal{F}$, the three expressions $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$ are formulas, with $[\![\phi_1 \wedge \phi_2]\!]_{\mathcal{T}} = [\![\phi_1]\!]_{\mathcal{T}} \cap [\![\phi_2]\!]_{\mathcal{T}}$, $[\![\phi_1 \vee \phi_2]\!]_{\mathcal{T}} = [\![\phi_1]\!]_{\mathcal{T}} \cup [\![\phi_2]\!]_{\mathcal{T}}$, and $[\![\neg\phi_1]\!]_{\mathcal{T}} = Q \setminus [\![\phi_1]\!]_{\mathcal{T}}$,
3. the set $\{\phi \in \mathcal{F} \mid [\![\phi]\!]_{\mathcal{T}} = \emptyset\}$ is recursive,
4. $[\![\phi_0]\!]_{\mathcal{T}} = Q^0$,
5. for each event $\sigma \in \Sigma$, there exists a computable map $\mathit{Pre}^\sigma : \mathcal{F} \to \mathcal{F}$ such that $[\![\mathit{Pre}^\sigma(\phi)]\!]_{\mathcal{T}} = \mathit{Pre}^\sigma([\![\phi]\!]_{\mathcal{T}})$ for all $\phi \in \mathcal{F}$.

By (1) and (2), the map $[\![\cdot]\!]_{\mathcal{T}}$ is an extension of $[\![\cdot]\!]$ from the set of atomic formulas to $\mathcal{F}$, and so we omit the subscript from $[\![\cdot]\!]_{\mathcal{T}}$.

A transition system S is *effective* if there exists a theory for S. ■

Proposition 2.1.9 *The reachability problem for the class of effective transition systems is recursively enumerable.*

Proof. Put $\psi_0 = \pi$, and $\psi_{n+1} = \bigvee_{\sigma \in \Sigma} \mathit{Pre}^\sigma(\psi_n)$. Then $[\![\psi_n]\!] = \mathit{Pre}^n([\![\pi]\!])$ for each $n \in \mathbb{N}$. By definition, $[\![\pi]\!]$ is reachable iff $[\![\phi_0 \wedge \psi_n]\!] \neq \emptyset$ for some $n \in \mathbb{N}$. By (3) above, the test $[\![\phi_0 \wedge \psi_n]\!] \neq \emptyset$ is computable. ■

2.1.2 Languages

We define the language of a transition system in a way that takes both state labels and action labels into account. This facilitates the correspondence between temporal logics, simulation, and language equivalence.

Definition 2.1.10 The *ω-language* of R in S, denoted $L^\omega(R,S)$, is the set of pairs of infinite words $(\bar{P},\bar{\sigma}) \in \mathcal{P}(\Pi_S)^\omega \times \Sigma_S^\omega$ such that there exists an infinite sequence $\bar{q}$ of states with (1) $\Pi(q_i) = P_i$ for all $i \in \mathbb{N}$, (2) $q_i \xrightarrow{\sigma_i} q_{i+1}$ for all $i \in \mathbb{N}$, and (3) $q_0 \in R$. The *ω-language* of S, denoted $L^\omega(S)$, is $L^\omega(Q^0,S)$. Two states q and r are *ω-language equivalent* if $L^\omega(\{q\},S) = L^\omega(\{r\},S)$. The *language* of R in S, denoted $L(R,S)$, is the set of pairs of finite words $(\bar{P},\bar{\sigma}) \in \mathcal{P}(\Pi_S)^* \times \Sigma_S^*$ with $|\bar{P}| = |\bar{\sigma}|+1$ and such that there exists an finite sequence $\bar{q}$ of states with (1) $\Pi(q_i) = P_i$ for all $0 \le i < |\bar{P}|$, (2) $q_i \xrightarrow{\sigma_i} q_{i+1}$ for all $0 \le i < |\bar{P}|-1$, and (3) $q_0 \in R$. The *language* of S, denoted $L(S)$, is $L(Q^0,S)$. ■

We use no acceptance condition. Later, for transition systems with time events, we will define the divergent language, which requires the sum of the durations of the time steps in an infinite word to diverge.

2.1.3 Simulation and Bisimulation

Definition 2.1.11 Let S be a transition system. We say that a binary relation $\sim$ on Q *respects satisfaction* if $q \sim r$ implies that for every atomic proposition π, $q \models \pi$ iff $r \models \pi$. Let $\sim$ be such an equivalence relation that respects satisfaction. A *$\sim$-block* is a union of $\sim$-equivalence classes. For regions $R, R' \subset Q$, and an

$$\begin{tikzcd} q \arrow[d, "\sigma"'] & \preceq & r \arrow[d, "\sigma"] \\ q' & \preceq & \exists r' \end{tikzcd}$$

Figure 2.2: The simulation condition

event $\sigma \in \Sigma$, define $R \xrightarrow{\sigma}_{\exists} R'$ iff $R \cap Pre^{\sigma}(R') \neq \emptyset$. The *quotient system* $S/_{\sim}$ is the transition system

$$(Q/_{\sim}, \Sigma, \rightarrow_{\exists}, Q^0/_{\sim}, \Pi, \overline{\models}),$$

where $\overline{\models}$ is defined by $R \overline{\models} \pi$ iff $q \models \pi$ for all (equivalently, for some) $q \in R$. ■

Definition 2.1.12 Let S be a transition system. A *simulation* on S is a binary relation $\preceq$ on Q such that (1) $\preceq$ respects satisfaction, and (2) whenever $q \preceq r$ and $q \xrightarrow{\sigma} q'$, there exists a state $r' \in Q$ such that $q' \preceq r'$ and $r \xrightarrow{\sigma} r'$. ■

Condition (2) is called the *simulation condition*. It is depicted in Figure 2.2.

Definition 2.1.13 Let S be a transition system. The union of all simulations on S, denoted $\preceq_S$, is a reflexive, transitive simulation on S. We say that r *simulates* q if $q \preceq_S r$. Define the binary relation $\equiv_S^{sim}$ on Q by $q \equiv_S^{sim} r$ iff $q \preceq_S r$ and $r \preceq_S q$. Since $\preceq_S$ is reflexive and transitive, the binary relation $\equiv_S^{sim}$ is an equivalence relation, called *similarity*. We say that q and r are *similar* if $q \equiv_S^{sim} r$. The quotient system $S/_{\equiv_S^{sim}}$ is the *similarity quotient* of S. ■

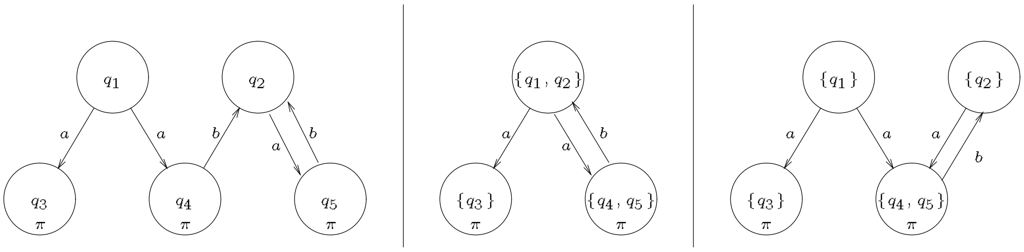

Figure 2.3: Transition system S_2; Similarity quotient; Bisimilarity quotient

Example 2.1.14 Consider the transition system S_2 depicted in the left frame of Figure 2.3. There is one atomic proposition, π. States satisfying π are labeled with π. The similarity quotient system $S_2/_{\equiv^{sim}_{S_2}}$ is given in the middle frame of the figure. ■

The next proposition is easy to prove.

Proposition 2.1.15 *Let S be a transition system and let q and r be states of S. If the two states q and r are similar then they are ω-language equivalent.* ■

Definition 2.1.16 Let S be a transition system. A region $R \subset Q$ is *upward-closed under simulation* if $q \in R$ and $q \preceq_S r$ implies $r \in R$. ■

Proposition 2.1.17 *The reachability problem is decidable for the class of effective transition systems with finite similarity quotient.*

Proof. Let S be a transition system and let π be an atomic proposition of S. Put $\xi_0 = \pi$, and $\xi_{n+1} = \xi_n \vee \bigvee_{\sigma \in \Sigma} Pre^{\sigma}(\xi_n)$. Then $[\![\pi]\!]$ is reachable iff $[\![\phi_0 \wedge \xi_n]\!] \neq \emptyset$ for some $n \in \mathbb{N}$. We show by induction that each $[\![\xi_n]\!]$ is upward-closed under simulation. For the base case, $[\![\xi_0]\!] = [\![\pi]\!]$ is upward-closed because $\preceq_S$ respects satisfaction. Inductively, if ξ_n is upward-closed, $q' \in [\![\xi_n]\!]$, $q \xrightarrow{\sigma} q'$, and $q \preceq_S r$, then there exists a state r' such that $r \xrightarrow{\sigma} r'$ and $q' \preceq_S r'$. Since $[\![\xi_n]\!]$ is upward-closed, it follows that $r' \in [\![\xi_n]\!]$. Since every upward-closed set is a $\equiv_S^{sim}$-block, it follows that $([\![\xi_n]\!])_{n \in \mathbb{N}}$ is an increasing sequence of $\equiv_S^{sim}$-blocks. If there are only finitely many $\equiv_S^{sim}$-blocks, then $[\![\xi_k]\!] = [\![\xi_{k+1}]\!]$ for some $k \in \mathbb{N}$, and $[\![\pi]\!]$ is reachable iff $[\![\phi_0 \cap \xi_k]\!] \neq \emptyset$. The test $[\![\xi_k]\!] = [\![\xi_{k+1}]\!]$ is computable by conditions (2) and (3) in the definition of theory. ■

Definition 2.1.18 Let S be a transition system. A *bisimulation* on S is a symmetric simulation on S. The union of all bisimulations on S, denoted $\equiv_S^{bis}$, is a bisimulation on S. The bisimulation $\equiv_S^{bis}$ is called *bisimilarity*. We say that q and r are *bisimilar* if $q \equiv_S^{bis} r$. The quotient system $S/_{\equiv_S^{bis}}$ is the *bisimilarity quotient* of S. ■

Example 2.1.19 The bisimilarity quotient system $S_2/_{\equiv_{S_2}^{sim}}$ is given in the right frame of Figure 2.3. ■

It is sometimes necessary to restrict the state space of a transition system to the set of reachable states in order for the (bi)similarity quotient to be finite.

Definition 2.1.20 The *reachable restriction* of a transition system S is the transition system $(Post^*(Q_S^0), \Sigma_S, \rightarrow_S, Q_S^0, \Pi_S, \models_S)$, where $\rightarrow_S$ and $\models_S$ are suitably restricted to $Post^*(Q_S^0)$. ■

Most theorems about systems with finite (bi)similarity quotient carry over to systems whose reachable restriction has finite (bi)similarity quotient, since the unreachable states rarely play any role.

2.1.4 Temporal Logic

System requirements are often expressed in temporal logic. In this section, we describe several logics used for this purpose, define their model checking problems, and give the basic complexity results that we use in later chapters.

Definition 2.1.21 Let $\mathcal{C}$ be a class of transition systems and let $\mathcal{L}$ be a logic or a fragment of a logic. The *model checking problem* for $\mathcal{L}$ on $\mathcal{C}$ is stated in the following way:

> *Given a transition system $S \in \mathcal{C}$ and a formula ϕ of $\mathcal{L}$, determine whether all initial states of S satisfy ϕ.* ■

CTL*

CTL* is a branching-time temporal logic that subsumes both linear temporal logic and computation tree logic.

Definition 2.1.22 Let S be a transition system and let q be a state of S. A *computation* from q is an infinite sequence $\kappa = q_0\sigma_0q_1\sigma_1\ldots \in (Q \cup \Sigma)^\omega$ such that $q_0 = q$ and $q_i \xrightarrow{\sigma_i} q_{i+1}$ for every $i \in \mathbb{N}$. Let $\kappa = q_0\sigma_0q_1\sigma_1\ldots$ be a computation. For each $i \in \mathbb{N}$, κ^i is the suffix of κ beginning with q_i. Thus $\kappa^i = q_i\sigma_iq_{i+1}\sigma_{i+1}\ldots$ is a computation from q_i. ■

Definition 2.1.23 Let S be a deadlock-free transition system. The formulas of *CTL** on S are defined inductively as follows:

Atomic Formulas. The atomic formulas of CTL* are the atomic formulas of S.

State Formulas. Every atomic formula is a state formula. Given two state formulas χ_1 and χ_2, and a path formula ξ, the following five expressions are state formulas: $\neg\chi_1$, $\chi_1 \vee \chi_2$, $\chi_1 \wedge \chi_2$, $\exists\xi$, $\forall\xi$.

Path Formulas. Every state formula is a path formula. Given two path formulas ξ_1 and ξ_2, and a label $\sigma \in \Sigma$, the following six expressions are path formulas: $\neg\xi_1$, $\xi_1 \vee \xi_2$, $\xi_1 \wedge \xi_2$, $\textcircled{\sigma}\xi_1$, $\xi_1\mathcal{U}\xi_2$, $\xi_1\mathcal{W}\xi_2$. ■

The two sorts of formulas require two satisfaction relations.

Definition 2.1.24 We define a satisfaction relation $\models_s$ between states of S and state formulas, and a satisfaction relation $\models_p$ between computations of S and path formulas. See Figure 2.1.4. (The boolean connectives are handled in the usual way, e.g., $q \models_s \chi_1 \wedge \chi_2$ iff $q \models_s \chi_1$ and $q \models_s \chi_2$, so we omit them from the figure.) We say that state q *satisfies* the state formula χ iff $q \models_s \chi$, and computation κ *satisfies* the path formula ξ iff $\kappa \models_p \xi$. ■

Definition 2.1.25 A CTL* formula χ is in *normal form* if in χ negations are applied only to atomic formulas. If χ is in normal form and does not contain the $\exists$ operator (resp. the $\forall$ operator) then χ is *universal* (resp. existential). ■

Theorem 2.1.26 [BCG88,BBLS92,DGG93] *Let S be a transition system and let q and r be states of S.*

1. *If $q \equiv_S^{bis} r$, then q and r satisfy the same state formulas of CTL*.*
2. *If $q \equiv_S^{sim} r$, then q and r satisfy the same universal state formulas of CTL*.*

$q \models_s \alpha$	iff	$q \models \alpha$, for an atomic formula α
$q \models_s \exists\xi$	iff	there exists a computation κ from q with $\kappa \models_p \xi$
$q \models_s \forall\xi$	iff	for every computation κ from q, $\kappa \models_p \xi$
$\kappa \models_p \chi$	iff	$q_0 \models_s \chi$, for a state formula χ, where $\kappa = q_0\sigma_0 q_1\sigma_1 \ldots$
$\kappa \models_p \sigma\xi$	iff	$\kappa^1 \models \xi$ and $\sigma_0 = \sigma$, where $\kappa = q_0\sigma_0 q_1\sigma_1 \ldots$,
$\kappa \models_p \xi_1 \mathcal{U} \xi_2$	iff	$\exists n \in \mathbb{N}.\, \kappa^n \models_p \xi_2$ and $\forall m < n.\, \kappa^m \models_p \xi_1$
$\kappa \models_p \xi_1 \mathcal{W} \xi_2$	iff	$\kappa \models_p \xi_1 \mathcal{U} \xi_2$ or $\forall n \in \mathbb{N}.\, \kappa^n \models_p \xi_1$

Figure 2.4: CTL* semantics

3. *If the bisimilarity quotient of S is finite, then $q \equiv_S^{bis} r$ iff q and r satisfy the same state formulas of CTL*.*

4. *If the similarity quotient of S is finite, then $q \equiv_S^{sim} r$ iff q and r satisfy the same universal state formulas of CTL*.* ■

Theorem 2.1.27 *1. The model checking problem for CTL* on finite transition systems is PSPACE-complete [CES86].*

2. *The model checking problem for CTL* on finite transition systems can be solved in space $O(m(m + \log n)^2)$, where m is the length of the formula and n is the size of the transition system. [Kup95].*

We will apply the second part of Theorem 2.1.27 to show that the model checking problem for the universal fragment of CTL* on positive two-dimensional rectangular hybrid automata can be solved in polynomial space.

Using the translation from CTL* into the μ-calculus [EL86], the following Corollary can be proven.

Corollary 2.1.28 *1. The model checking problem for CTL* is decidable on the class of effective transition systems with finite bisimilarity quotient.*

2. The model checking problems for the universal and existential fragments of CTL are decidable on the class of effective transition systems with finite similarity quotient.*

Definition 2.1.29 Let S be a transition system A *fairness condition* is a set $\mathcal{F}$ of computations of S. The semantics of CTL* with fairness condition $\mathcal{F}$ is defined as in Figure 2.1.4, except that the $\exists$ and $\forall$ operators quantify over the computations in $\mathcal{F}$. ∎

When the fairness condition $\mathcal{F}$ is defined in one of the usual ways (as an unconditional, weak fairness (justice), or strong fairness (compassion) condition [MP92]), then the complexity of the model checking problem for CTL* does not change.

Linear Temporal Logic

Linear temporal logic (LTL) is a sublogic of CTL* which only takes into account the set of computations of a state, and not the whole tree of computation.

Definition 2.1.30 The formulas of LTL are those state formulas of CTL* of the form $\forall\xi$, where the path formula ξ does not contain the $\exists$ or $\forall$ path quantifiers. ∎

We have the following simple proposition.

Proposition 2.1.31 *Let S be a deadlock-free transition system, and let q and r be states of S. If q and r are ω-language equivalent, then they satisfy the same formulas of LTL.* ■

Theorem 2.1.32 1. *The model checking problem for LTL on finite transition systems is PSPACE-complete [SC85].*

2. *The model checking problem for LTL on finite transition systems can be solved in space $O((m + \log n)^2)$, where m is the length of the formula and n is the size of the transition system. [VW86].* ■

We will apply the second part of Theorem 2.1.32 to show that the model checking problem for LTL under the fairness condition of time divergence on initialized rectangular hybrid automata with bounded nondeterminism can be solved in polynomial space.

2.1.5 Control

Definition 2.1.33 Let S be a transition system. A *control map* is a function $\kappa : Q \to \Sigma$ such that for all $q \in Q$, there exists an $r \in Q$ such that $q \stackrel{\kappa(q)}{\to} r$. The *closed-loop system* is the transition system

$$\kappa(S) =_{\text{def}} (Q, \Sigma, \Rightarrow, Q^0, \Pi, \models),$$

where $q \stackrel{\sigma}{\Rightarrow} q'$ iff $q \stackrel{\sigma}{\to} q'$ and $\kappa(q) = \sigma$. ■

If S is not deadlock-free, then this definition of control map is rather onerous (no control map exists), and requires slight modification. The present definition is sufficient for our purposes.

Definition 2.1.34 The *control decision problem* for a class $\mathcal{C}$ of transition systems is stated in the following way:

> *Given a transition system $S \in \mathcal{C}$ and an atomic proposition $\pi \in \Pi$, determine whether there exists a control map κ such that $[\![\pi]\!]$ is not reachable in the closed-loop system $\kappa(S)$.*

If such a control map κ exists, we say that $[\![\pi]\!]$ is *avoidable*. *Controller synthesis* further requires the construction of the control map κ when $[\![\pi]\!]$ is avoidable. ■

Proposition 2.1.35 [Hen95,AMP95b] *The control decision problem is co-r.e. for the class of effective transition systems.*

Proof. Let S be an effective transition system, let $(\mathcal{F}, \phi_0, [\![\cdot]\!])$ be a theory for S, and let $\pi \in \Pi$ be an atomic proposition. Define a map $\eta \colon \mathcal{F} \to \mathcal{F}$ by

$$\eta(\phi) = \bigwedge_{\sigma \in \Sigma} (Pre^{\sigma}(\phi) \vee \neg Pre^{\sigma}(true)).$$

The region $[\![\eta(\phi)]\!]$ is the set of all states that no control map can keep out of $[\![\phi]\!]$ at the next transition. Put $\psi_0 = \pi$, and $\psi_{n+1} = \psi_n \vee \eta(\psi_n)$. It follows that the control decision problem is answered in the affirmative iff $[\![\phi_0 \wedge \psi_n]\!] = \emptyset$ for all $n \in \mathbb{N}$. ■

Proposition 2.1.36 [Hen95] *The control decision problem is decidable for the class $\mathcal{C}$ of effective transition systems with finite bisimilarity quotient. Moreover, controller synthesis for $\mathcal{C}$ is computable.*

Proof. In the notation of the proof of Proposition 2.1.35, $([\![\psi_n]\!])_{n \in \mathbb{N}}$ is an increasing sequence of $\equiv_S^{bis}$-blocks. The reachability algorithm terminates because there are only finitely many $\equiv_S^{bis}$-blocks, and so $\psi_{k+1} = \psi_k$ for some k. Controller synthesis is accomplished by first computing the bisimilarity quotient [BFH90,Hen95], and then choosing for each bisimulation equivalence class R an event $\sigma \in \Sigma$ such that $Pre^\sigma(\llbracket\psi_k\rrbracket) \cap R = \emptyset$. ■

This theorem may be generalized to systems with finite similarity quotients, if the domain of each $\xrightarrow{\sigma}$ is the set of states satisfying an atomic formula.

Definition 2.1.37 A transition system S is *standard* if for each $\sigma \in \Sigma$, there exists an atomic formula α such that $Pre^\sigma(Q) = \llbracket\alpha\rrbracket$. ■

Theorem 2.1.38 *The control decision problem is decidable for the class $\mathcal{C}$ of standard, effective transition systems with finite similarity quotient. Moreover, controller synthesis for $\mathcal{C}$ is computable.*

Proof. Suppose S is a standard, effective transition system with finite similarity quotient. In the notation of the proof of Proposition 2.1.35, we show by induction that each $\llbracket\psi_n\rrbracket$ is upward-closed under simulation, and therefore a $\equiv_S^{sim}$-block. Thereafter the proof proceeds as for Proposition 2.1.36. First, $\llbracket\psi_0\rrbracket = \llbracket\pi\rrbracket$ is trivially upward-closed. Next, suppose $\llbracket\psi_n\rrbracket$ is upward-closed, $q \in \llbracket\psi_{n+1}\rrbracket$, and $q \preceq_S r$. If $q \in \llbracket\psi_n\rrbracket$ then $r \in \llbracket\psi_n\rrbracket$ by the inductive hypothesis. Otherwise, let $\sigma \in \Sigma$ be an event. Either $q \notin Pre^\sigma(Q)$ or $q \in Pre^\sigma(\llbracket\psi_n\rrbracket)$. If $q \notin Pre^\sigma(Q)$, then $r \notin Pre^\sigma(Q)$ because $q \preceq_S r$, $\preceq_S$ respects satisfaction, and S is standard. If $q \in Pre^\sigma(\llbracket\psi_n\rrbracket)$, then there exists a state $q' \in \llbracket\psi_n\rrbracket$ such that $q \xrightarrow{\sigma} q'$. Since $q \preceq_S r$, there exists a state r' such that $r \xrightarrow{\sigma} r'$ and $q' \preceq_S r'$. Since $\llbracket\psi_n\rrbracket$ is upward-closed, it follows that $r' \in \llbracket\psi_n\rrbracket$. Thus $r \in \llbracket\psi_{n+1}\rrbracket$. ■

2.2 Rectangular Hybrid Automata

2.2.1 Syntax

Definition 2.2.1 An n-dimensional *rectangle* is a subset B of $\mathbb{R}^n$ that is a cartesian product $\prod_{i=1}^{n} B_i$ of (possibly unbounded) intervals, all of whose finite endpoints are rational numbers. The set of all n-dimensional rectangles is denoted by $\mathcal{B}_n$. ■

Definition 2.2.2 An *n-dimensional rectangular automaton*[2] A consists of the following components:

Control graph. A finite directed multigraph (V_A, E_A), representing the discrete component of the system.

Invariants. A function $inv : V_A \to \mathcal{B}_n$ mapping each vertex to its *invariant*, an n-dimensional rectangle. The invariant of vertex v specifies the allowable continuous states while control is in v.

Activity rectangles. A function $act_A : V_A \to \mathcal{B}_n$ mapping each vertex to its *activity rectangle*. The activity rectangle $act_A(v)$ constrains the continuous behavior of A while control is in vertex v.

Initial conditions. A function $init_A : V_A \to \mathcal{B}_n$ mapping each control mode to its *initial condition*.

[2]For the sake of brevity, we use the term "rectangular automaton" rather than "rectangular hybrid automaton" in the main text.

Discrete actions. Functions $preguard_A : E_A \to \mathcal{B}_n$, $update_A : E_A \to 2^{\{1,\ldots,n\}}$, and $postguard_A : E_A \to \mathcal{B}_n$. These functions specify the conditions under which edges in the control graph may be traversed, and the effects of these transitions.

Events. A finite alphabet Σ_A of *events*, and a function $event_A \colon E_A \to \Sigma_A$ assigning an event to each edge.

We usually suppress the subscript on the components of A. Thus a rectangular automaton A is a tuple

$$(V, E, inv, act, init, preguard, update, postguard, \Sigma, event).$$

An *n-dimensional rectangular automaton with silent moves* differs in that the function $event_A$ maps E_A into Σ_A^τ, where $\Sigma_A^\tau = \Sigma_A \cup \{\tau\}$ augments Σ_A with the *silent event* $\tau \notin \Sigma_A$. ■

2.2.2 Semantics

A state of a rectangular automaton consists of a discrete part and a continuous part. The initial function *init* specifies a set of initial automaton states: when the discrete state begins at v, the continuous state must begin in $init(v)$. The functions *preguard*, *postguard*, and *update* constrain the behavior of the automaton state during edge steps. The edge $e = (v, w)$ may be traversed only if the discrete state resides at v and the continuous state lies in $preguard(e)$. For each $i \notin update(e)$, the ith coordinate of the continuous state is not changed and must lie in $postguard(e)_i$ (recall that $postguard(e)_i$ is the projection of the rectangle $postguard(e)$ on its ith coordinate). For each $i \in \mathit{update}(e)$, the ith coordinate of the continuous state is nondeterministically assigned a new value in $\mathit{postguard}(e)_i$. The invariant function *inv* and the activity function *act* constrain the behavior of the automaton state during time steps. While the discrete state resides at vertex v, the continuous state nondeterministically follows a differentiable trajectory within $\mathit{inv}(v)$, and its first derivative remains within $\mathit{act}(v)$. A rectangular automaton with silent moves may traverse τ-edges during time steps.

Definition 2.2.3 Let A be an n-dimensional rectangular automaton, with or without silent moves. A *state* $(v, \mathbf{x})$ of A consists of a discrete part $v \in V$ and a continuous part $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} \in \mathit{inv}(v)$. The *state space* Q_A of A is the set $\{(v, \mathbf{x}) \in V \times \mathbb{R}^n \mid \mathbf{x} \in \mathit{inv}(v)\}$ of all states of A. A subset of Q_A is called a *zone*. Each zone $Z \subset Q_A$ can be uniquely decomposed into a collection $\bigcup_{v \in V}\{v\} \times [Z]_v$ of sets $[Z]_v \subset \mathbb{R}^n$, one for each vertex v. The zone Z is *closed* (resp. *bounded*, *compact*), if each $[Z]_v$ is closed (resp. bounded, compact). The zone Z is *rectangular* if each $[Z]_v$ is a rectangle. The zone Z is *multirectangular* if it is a finite union of rectangular zones. The state $(v, \mathbf{x})$ of A is *initial* if $\mathbf{x} \in \mathit{init}(v)$. The *initial zone* $Q_A^0 \subset Q_A$ is the set of all initial states of A. ■

Both the state space Q_A and the initial zone Q_A^0 are rectangular zones.

Definition 2.2.4 Let A be a rectangular automaton, with or without silent moves. For each edge $e = (v, w) \in E$, we define the binary *edge-step relation* $\xrightarrow{e}$ on Q_A by $(v', \mathbf{x}) \xrightarrow{e} (w', \mathbf{y})$ iff (1) $v' = v$ and $w' = w$, (2) $\mathbf{x} \in \mathit{preguard}(e)$ and $\mathbf{y} \in \mathit{postguard}(e)$, and (3) for each $i \notin \mathit{update}(e)$, $x_i = y_i$. Hence $\mathbf{x}$ and $\mathbf{y}$ differ only at coordinates in the update set $update(e)$. For each event $\sigma \in \Sigma^\tau$, we define the binary *jump relation* $\xrightarrow{\sigma}$ on Q_A by $(v,\mathbf{x}) \xrightarrow{\sigma} (w,\mathbf{y})$ iff $(v,\mathbf{x}) \xrightarrow{e} (w,\mathbf{y})$ for some edge $e \in E$ with $event(e) = \sigma$.

For each $t \in \mathbb{R}_{\geq 0}$, we define the relation $\stackrel{t}{\Rightarrow}$ on Q_A by $(v,\mathbf{x}) \stackrel{t}{\Rightarrow} (w,\mathbf{y})$ iff (1) $v = w$ and (2) either $t = 0$ and $\mathbf{x} = \mathbf{y}$, or $t > 0$ and $\frac{\mathbf{y}-\mathbf{x}}{t} \in act(v)$. Notice that due to the convexity of rectangles, $(v,\mathbf{x}) \stackrel{t}{\Rightarrow} (w,\mathbf{y})$ iff there is a smooth function $f : [0,t] \to inv(v)$ with $f(0) = \mathbf{x}$, $f(t) = \mathbf{y}$, and for all $s \in (0,t)$, $\dot{f}(s) \in act(v)$. Hence the continuous state may evolve from $\mathbf{x}$ to $\mathbf{y}$ via any smooth trajectory satisfying the constraints imposed by $inv(v)$ and $act(v)$. If A does not have silent moves, then we define the *time-step relation* $\xrightarrow{t}$ to be $\stackrel{t}{\Rightarrow}$. If A has silent moves, then the time-step relation $\xrightarrow{t}$ is defined by $q \xrightarrow{t} q'$ iff there exists an integer $m \geq 1$, nonnegative reals $t_1, \ldots, t_m$, and states $q_1, \ldots, q_{2m-2}$ such that $q \stackrel{t_1}{\Rightarrow} q_1 \xrightarrow{\tau} q_2 \stackrel{t_2}{\Rightarrow} q_3 \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_{2m-2} \stackrel{t_m}{\Rightarrow} q'$ and $t = \sum_{i=1}^{m} t_i$. ■

A rectangular automaton A defines several transition systems over the state space Q_A and subsets of the transition relation $\rightarrow$. In Chapter 8, we study discrete-time semantics, in which time steps are only allowed to have duration 1. There we use the *discrete-time* transition system of A, for which the set of events is $\Sigma \cup \{1\}$. In Chapter 5, we study divergent-language emptiness in the context of dense time. There we use the *timed* transition system, for which the set of events is $\Sigma \cup \mathbb{R}_{\geq 0}$. In Chapter 7, we study finitary language equivalence. There we use the *time-abstract* transition system, for which the set of events is $\Sigma \cup \{time\}$. The *time* event subsumes all real-valued time durations.

2.2.3 Graphical Display Language

It is often convenient to give the continuous variables names $a, b, c, \ldots$ instead of referring to them by their coordinates. If the variable a corresponds to the ith coordinate, we write $act(v)(a)$ instead $act(v)_i$, etc. When using this conventions, the set $update(e)$ is a set of variable names, rather than a set of coordinates. In figures, we annotate each vertex with its activity function, and sometimes with its invariant. For example, if $act(v)(a) = [3, 5]$, $act(v)(b) = [4, 4]$, $inv(v)(a) = (1, 7]$, and $inv(v)(b) = (-\infty, 0]$, we write "$\dot{a} \in [3, 5]$", "$\dot{b} = 4$", "$1 < a \leq 7$", and "$b \leq 0$" inside of v. Often however, we give the invariant in the text and omit it from the figure. Edges are annotated with event labels and guarded commands. A guarded command ψ defines regions $preguard(\psi)$ and $postguard(\psi)$, and an update set $update(\psi)$, in a natural manner. For example, if ψ is the guarded command

$$a \leq 5 \ \wedge\ b = 4 \ \rightarrow\ b := 7;\ c :\in [0, 5],$$

then $preguard(\psi)(a) = (-\infty, 5]$, $preguard(\psi)(b) = [4, 4]$, and $preguard(\psi)(c) = \mathbb{R}$; $update(\psi) = \{b, c\}$; $postguard(\psi)(a) = \mathbb{R}$, $postguard(\psi)(b) = [7, 7]$, and finally $postguard(\psi)(c) = [0, 5]$.

Example 2.2.5 Consider, for instance, the 2D rectangular automaton $\hat{A}$ of Figure 2.5. The set of events of $\hat{A}$ is $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. Suppose c refers to the first coordinate, and d to the second. Then $act_{\hat{A}}(v_1) = [1, 3] \times [-3, 2]$ and $act_{\hat{A}}(v_4) = [1, 3] \times [1, 2]$. If edge e_i is labeled with event σ_i, then $preguard(e_1) = (-\infty, 5] \times (-\infty, -3]$, $update(e_1) = \{1\}$, and $postguard(e_1) = \{4\} \times \mathbb{R}$. Similarly, $preguard(e_3) = [-3, \infty) \times (-\infty, -2]$, $update(e_3) = \{1\}$, and $postguard(e_3) = [-2, -1] \times \mathbb{R}$. ■

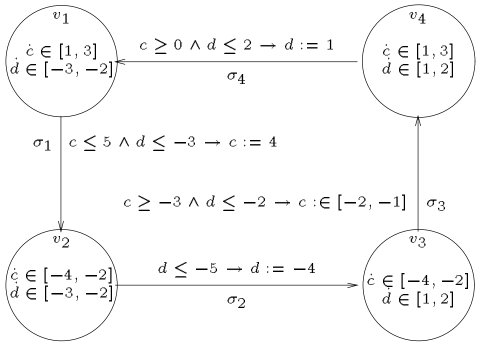

Figure 2.5: The initialized rectangular automaton $\hat{A}$

2.2.4 Triangular Automata

Rectangular automata only allow comparisons between continuous variables and rational constants. This can be generalized to allow comparisons between the values of the continuous variables. The result is the class of triangular automata. Generally speaking, all interesting questions about triangular automata are undecidable. We use triangular automata in several places to show that rectangular automata are the most general class for which a given question is decidable.

Definition 2.2.6 An n-dimensional *triangle* is a subset Δ of $\mathbb{R}^n$ that is the intersection of a rectangle B with a finite number (zero or more) half spaces of the form $\{\mathbf{x} \in \mathbb{R}^n \mid x_i \leq x_j\}$ or $\{\mathbf{x} \in \mathbb{R}^n \mid x_i < x_j\}$, where i and j can vary through $\{1, \ldots, n\}$. ■

Definition 2.2.7 A *triangular automaton* has the same components as a rectangular automaton, except that for each vertex $v \in V$, and each edge $e \in E$, the sets *init*(v), *inv*(v), *act*(v), *preguard*(e), and *postguard*(e) need only be triangles, and not necessarily rectangles. ■

Every rectangle is a triangle, and so every rectangular automaton is a triangular automaton.

2.2.5 Basic Definitions

In this section we collect a few definitions which are used throughout this dissertation.

Definition 2.2.8 The *size* of a rectangular automaton A is its length as a string when all rational constants are written in binary notation. A rectangle $B \subset \mathbb{R}^n$ is written as a list of the endpoints of its projections:

$$(\inf B_1, \sup B_1, \ldots, \inf B_n, \sup B_n),$$

with special encodings for ∞ and $-\infty$, and a list of bits specifying, for each i, which of the endpoints of the B_i belong to B_i. ■

Definition 2.2.9 A rectangle B is *integral* if all of its finite endpoints are integers. A rectangular automaton A, with or without silent moves, is *integral* if for every $v \in V$ and every $e \in E$, the rectangles $init(v)$, $inv(v)$, $act(v)$, $preguard(e)$, and $postguard(e)$ are integral. ■

Definition 2.2.10 A triangular automaton A is *positive* if for every vertex $v \in V$, and every edge e, the triangles $inv(v)$, $act(v)$, $preguard(e)$, and $postguard(e)$ are contained in the positive orthant $\mathbb{R}^n_{\geq 0}$. ■

Definition 2.2.11 A rectangular automaton A is *closed* (resp. *bounded*, *compact*) if all $init(v)$, $inv(v)$, $act(v)$, $preguard(e)$, and $postguard(e)$ are closed (resp. bounded, compact). ■

Definition 2.2.12 Let h be a natural number. A triangle $B \subset \mathbb{R}^n_{\geq 0}$ is *h-definable* if for each $1 \leq i \leq n$, either $B_i \subset [0, h]$ or $(h, \infty) \subset B_i$. A triangular automaton A is *h-definable* if for all $v \in V$ and all $e \in E$, *init*(v), *inv*(v), *act*(v), *preguard*(e), and *postguard*(e) are h-definable triangles. ■

Definition 2.2.13 Let B be an n-dimensional rectangle, and let $J \subset \{1, \ldots, n\}$ be a set of coordinates. Say $J = \{j_1, \ldots, j_k\}$, where $j_1 < j_2 < \cdots < j_k$. The *projection* of B onto J, denoted $B\lceil_J$, is the $|J|$-dimensional rectangle $\prod_{i=1}^k B_{j_i}$. For a vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}\lceil_J$ is the vector $(x_{j_1}, x_{j_2}, \ldots, x_{j_k})$. ■

Definition 2.2.14 Let A be an n-dimensional rectangular automaton, and let $J \subset \{1, \ldots, n\}$ be a nonempty set of coordinates. The *projection* of A onto J is the $|J|$-dimensional rectangular automaton $A\lceil_J$ defined from A by replacing each *inv*(v), *init*(v), *act*(v), *preguard*(e), and *postguard*(e) by its projection onto J, and replacing each *update*(e) with *update*$(e) \cap J$. ■

Chapter 3

Reachability: Decidability

The reachability problem, which asks whether a system can ever enter a given set of states, is the most basic problem in program verification. In this chapter we prove the theorem of Puri and Varaiya [PV94], which states that the reachability problem is decidable for the class of initialized rectangular automata.

3.1 The Time-Abstract Transition System

Definition 3.1.1 Let A be a triangular automaton, with or without silent moves. Let $\Pi_A = \{\pi_v \mid v \in V_A\}$, and define $(w, \mathbf{x}) \models_A \pi_v$ iff $v = w$. Define the binary relation $\stackrel{time}{\rightarrow}$ on Q_A by $(v, \mathbf{x}) \stackrel{time}{\rightarrow} (w, \mathbf{y})$ iff $(v, \mathbf{x}) \stackrel{t}{\rightarrow} (w, \mathbf{y})$ for some $t \in \mathbb{R}_{\geq 0}$.

The triangular automaton A defines the *time-abstract* transition system

$$S_A^{abs} =_{\text{def}} (Q_A, \Sigma_A \cup \{time\}, \rightarrow, Q_A^0, \Pi_A, \models_A). \blacksquare$$

Definition 3.1.2 Let A be a rectangular automaton. A state $(v, \mathbf{x}) \in Q_A$ or zone $Z \subset Q_A$ is said to be *reachable* if it is reachable in the time-abstract transition system S_A^{abs}. ■

Definition 3.1.3 The *dense-time reachability problem* for a class $\mathcal{C}$ of triangular automata is stated in the following way:

> *Given a triangular automaton $A \in \mathcal{C}$ and a* rectangular *zone $Z \subset Q_A$, determine whether Z is reachable in the time-abstract transition system S_A^{abs}.* ■

Timed Automata

The most important subclass of rectangular automata is the class of timed automata [Alu91,ACD93,AD94], in which every continuous variable is a precise clock. The time-abstract transition systems of timed automata have finite bisimilarity quotients. It follows that the reachability and control problems for timed automata are decidable. Moreover, the divergent language emptiness problem for timed automata is also decidable.

Definition 3.1.4 A rectangular automaton A has *deterministic updates* if (1) the set of initial states Q_A^0 is finite, and (2) for every edge $e \in E$, and every i. if $i \in \mathit{update}(e)$ then $\mathit{postguard}(e)_i$ is a singleton. ■

Condition (2) requires that assignments to the continuous variables be deterministic.

Definition 3.1.5 A *multirate automaton* is a rectangular automaton M with deterministic updates such that there exists a vector $\mathbf{s} \in \mathbb{R}^n$ with $act(v) = \{\mathbf{s}\}$ for every vertex $v \in V$. If $\mathbf{s} = \mathbf{1}$, then M is a *timed automaton*. ■

Thus each continuous variable x of a timed automaton satisfies the differential equation $\frac{dx}{dt} = 1$.

Theorem 3.1.6 [AD94] *For every timed automaton T, with or without silent moves, the reachable restriction of the time-abstract transition system S_T^{abs} has finite bisimilarity quotient.*

Proof Sketch. Without loss of generality, assume that T is integral. Since the set of initial states of T is finite, assignments are deterministic, and each continuous variable has positive slope, there exists a $k \in \mathbb{Z}$ such that for every reachable state $(v, \mathbf{x})$, $x_i \geq k$ for all i. It suffice to assume $k = 0$, so that T is positive. Let T be h-definable. Define $(v, \mathbf{x}) \equiv (w, \mathbf{y})$ iff $v = w$, $\mathbf{x} \sim_n^{dis_h} \mathbf{y}$, and for all i and j with $x_i, x_j \leq h$, $\lfloor x_i - x_j \rfloor = \lfloor y_i - y_j \rfloor$. Then $\equiv$ is a bisimulation on the reachable restriction of S_T^{abs}. The number of equivalence classes of $\equiv$ is $O(|V|(2m+2)^n n!^2)$. ■

Corollary 3.1.7 [ACHH93,NOSY93] *For every multirate automaton M, with or without silent moves, the reachable restriction of the time-abstract transition system S_M^{abs} has finite bisimilarity quotient.*

Proof Sketch. Given a multirate automaton M, we define a timed automaton T_M for which the time-abstract transition systems S_M^{abs} and $S_{T_M}^{abs}$ are isomorphic. Let $act_M(v) = \{\mathbf{s}\}$ for each vertex, and assume for simplicity that each s_i is nonzero.

For a rectangle B, define $\alpha(B) = \{\mathbf{y} \in \mathbb{R}^n \mid \exists \mathbf{x} \in B. \forall i. y_i = \frac{x_i}{s_i}\}$. Let T_M have the same control graph as M, put $act_{T_M}(v) = \{\mathbf{1}\}$ for all $v \in V$, and replace every other rectangle B appearing in the definition of M by $\alpha(B)$. The timed automaton T_M has the desired properties. ■

Theorem 3.1.8 [AD94] *The reachability problem is PSPACE-complete for the class of multirate automata with silent moves.*

Proof. Let M be a multirate automaton. By the proofs of Theorem 3.1.6 and Corollary 3.1.7, the number of bisimulation equivalence classes of the reachable restriction of the time-abstract transition system S_M^{abs} is singly exponential in the size of M. (This also depends upon encoding constants in binary (Definition 2.2.8) rather than in unary.) Reachability can be solved in space logarithmic in the number of equivalence classes. ■

Definition 3.1.9 For a nonempty closed interval $I \subset \mathbb{R}$, and a number $x \in \mathbb{R}$, define $closest(x, I)$ to be the closest number to x in I. Thus if $x \in I$, then $closest(x, I) = x$; if $x < \min I$, then $closest(x, I) = \min I$; and if $x > \max I$, then $closest(x, I) = \max I$. ■

Definition 3.1.10 A *multirate automaton with attractors* M has the same components as a multirate automaton (with one restriction), but the transition relation is defined in a different way. First, for every edge e, and every i, it is required that if $i \notin update(e)$ then $postguard(e)_i$ be closed. The edge-step relation $\xrightarrow{e} \subset Q_M^2$ is modified in the following way: for an edge $e = (v, w) \in E$, define $(v', \mathbf{x}) \xrightarrow{e} (w', \mathbf{y})$

iff (1) $v' = v$ and $w' = w$, (2) $\mathbf{x} \in \mathit{preguard}(e)$, (3) $\mathbf{y} \in \mathit{postguard}(e)$, and (4) for each $i \notin \mathit{update}(e)$, $y_i = \mathit{closest}(x_i, \mathit{postguard}(e)_i)$. ■

Corollary 3.1.11 *The reachability problem is PSPACE-complete for the class of multirate automata with attractors and silent moves.*

Proof. Each edge e of a n-dimensional multirate automaton M with attractors can be replaced by no more than $3^{n-\mathit{update}(e)}$ edges in such a way that time-abstract transition system of the resulting multirate automaton (without attractors) is isomorphic to the time-abstract transition system of M. For example, in one dimension, if $1 \notin \mathit{update}(e)$, and I is bounded, then the attractor edge e is replaced by three conventional edges, e_1, e_2, and e_3, where (1) $\mathit{preguard}(e_1) = \mathit{preguard}(e) \cap (-\infty, \min I)$, $\mathit{update}(e_1) = \{1\}$, and $\mathit{postguard}(e_1) = \{\min I\}$; (2) $\mathit{preguard}(e_2) = \mathit{postguard}(e_2) = \mathit{preguard}(e) \cap I$ and $\mathit{update}(e_2) = \emptyset$; and (3) $\mathit{preguard}(e_3) = \mathit{preguard}(e) \cap (\max I, \infty)$, $\mathit{update}(e_3) = \{1\}$, and $\mathit{postguard}(e_3) = \{\max I\}$. The size of the bisimilarity quotient remains singly exponential. ■

3.2 Skewed-Clock Translation: Compact Case

Definition 3.2.1 We say a rectangular automaton A has *uniform dynamics* if for all vertices $v, w \in V$, $\mathit{act}(v) = \mathit{act}(w)$. In this case we also say that A has *uniform activity rectangle* B, where $B = \mathit{act}(v)$. ■

We reduce the reachability problem for rectangular automata with uniform dynamics to the reachability problem for multirate automata with attractors.[1]

We translate a given n-dimensional rectangular automaton A with uniform dynamics into a $2n$-dimensional multirate automaton N_A with attractors, such that N_A contains all reachability information about A. In this section we make the translation for compact A, which contains all of the main ideas. In the next, we give the general construction, which includes much additional bookkeeping.

Let A be an n-dimensional compact rectangular automaton with uniform dynamics. The idea is to replace each continuous variable a of A with two multirate variables (skewed clocks), a_ℓ and a_u, such that if $act_A(v)(a) = [k, k']$ then $act_{N_A}(v)(a_\ell) = \{k\}$ and $act_{N_A}(v)(a_u) = \{k'\}$. Consider Figure 3.1. With each time step, the activity of a creates an envelope, whose boundaries are tracked by a_ℓ and a_u. With each edge step, the values of a_ℓ and a_u are updated so that the interval $[a_\ell, a_u]$ is precisely the range of possible values of a. In Figure 3.1, at time t a transition is taken along an edge e with $preguard_A(e)(a) = [m, \infty)$. Since the value of a_ℓ is below m at time t, a_ℓ is updated to the new value m. This update is accomplished by using an attractor. In the following formal definition of the multirate automaton N_A, the variables $y_{\ell(i)}$ and $y_{u(i)}$ correspond respectively to the lower and upper bound multirate variables for variable x_i of A. For concreteness, put $\ell(i) = 2i - 1$ and $u(i) = 2i$.

[1]The original reduction [HPV94] did not use attractors, and therefore required the on-the-fly techniques that are still required in the general case. Attractors were introduced in [HK96] to simplify the construction.

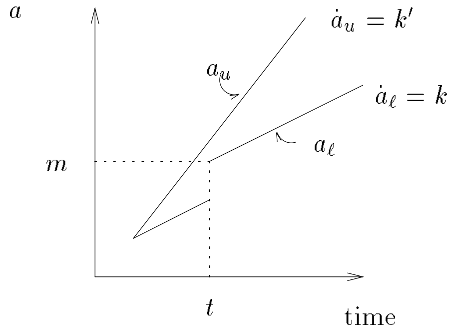

Figure 3.1: Envelope of the activity $act(v)(a) = [k, k']$

Definition 3.2.2 [HPV94] Let A be a compact, n-dimensional rectangular automaton with uniform dynamics. We assume that for each edge $e = (v, w)$ of A, $preguard(e) \subset inv(v)$, $postguard(e) \subset inv(w)$, and $preguard_A(e)_i = postguard_A(e)_i$ for each $i \notin update_A(e)$. If this is not the case, then we replace each guard with its intersection with the appropriate invariant, and then replace each $preguard_A(e)_i$ and $postguard_A(e)_i$ with $i \notin update_A(e)$ by their intersection $preguard_A(e)_i \cap postguard_A(e)_i$.

We define a $2n$-dimensional multirate automaton N_A with attractors. It has the same observation alphabet as A, and the same control graph. The initial function $init_{N_A}$ is given by

$$init_{N_A}(v)_{\ell(i)} = \{\min init_A(v)_i\} \text{ and } init_{N_A}(v)_{u(i)} = \{\max init_A(v)_i\}.$$

The invariant function inv_{N_A} is given by

$$inv_{N_A}(v)_{\ell(i)} = (-\infty, \max inv_A(v)_i] \text{ and } inv_{N_A}(v)_{u(i)} = [\min inv_A(v)_i, \infty).$$

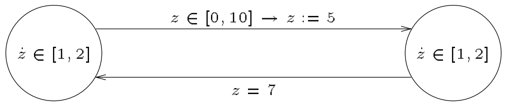

Figure 3.2: A rectangular automaton A

The activity function act_{N_A} is defined as outlined above,

$$act_{N_A}(v)_{\ell(i)} = \{\min act_A(v)_i\} \text{ and } act_{N_A}(v)_{u(i)} = \{\max act_A(v)_i\}.$$

The guards and attractors are defined in the following way. Let $e \in E$. Define

$$preguard_{N_A}(e)_{\ell(i)} = (-\infty, \max preguard_A(e)_i],$$
$$preguard_{N_A}(e)_{u(i)} = [\min preguard_A(e)_i, \infty).$$

The update sets are the same: $update_{N_A}(e) = update_A(e)$. For $i \in update(e)$,

$$postguard_{N_A}(e)_{\ell(i)} = \{\min preguard_A(e)_i\},$$
$$postguard_{N_A}(e)_{u(i)} = \{\max preguard_A(e)_i\}.$$

For $i \notin update(e)$, $postguard_{N_A}(e)_{\ell(i)} = postguard_{N_A}(e)_{u(i)} = postguard_A(e)$. This completes the definition of N_A. ∎

Example 3.2.3 Figure 3.2 shows a typical rectangular automaton A with uniform dynamics. Figure 3.3 gives the multirate automaton with attractors N_A. Figure 3.4 shows the timed automaton with attractors T_{N_A}. ∎

Definition 3.2.4 Let A be a rectangular automaton, with or without silent moves. For an edge $e \in E$, and a zone $Z \subset Q_A$, define

$$Post^e(Z) =_{\text{def}} \{(w, \mathbf{y}) \mid \exists (v, \mathbf{x}) \in Z. (v, \mathbf{x}) \xrightarrow{e} (w, \mathbf{y})\}.$$

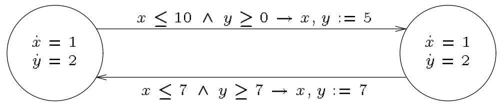

Figure 3.3: The multirate automaton with attractors N_A

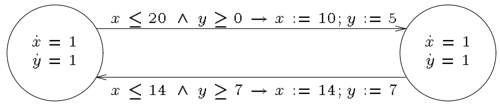

Figure 3.4: The timed automaton with attractors T_{M_A}

If Z is a singleton $\{(v, \mathbf{x})\}$, then we write $Post^e(v, \mathbf{x})$ instead of $Post^e(\{v, \mathbf{x}\})$. In the analogous way define the operator Pre^e. ■

Lemma 3.2.5 *Let A be a compact rectangular automaton with uniform dynamics. For every reachable state $(v, \mathbf{y})$ of N_A, $y_{\ell(i)} \leq y_{u(i)}$.* ■

Definition 3.2.6 Let A be a compact rectangular automaton with uniform dynamics. Define the map $\xi_A : Q_{N_A} \to \mathcal{P}(Q_A)$ by

$$\xi_A(v, \mathbf{y}) =_{\text{def}} \{v\} \times (inv_A(v) \cap \prod_{i=1}^{n} [y_{\ell(i)}, y_{u(i)}]).$$

Extend ξ_A to $\mathcal{P}(Q_{N_A})$ by $\xi_A(Z) =_{\text{def}} \bigcup_{(v,\mathbf{y}) \in Z} \xi_A(v, \mathbf{y})$. ■

Our first lemma shows that the correspondence between N_A and A is properly initialized.

Lemma 3.2.7 *For every compact rectangular automaton A with uniform dynamics, $\xi_A(Q^0_{N_A}) = Q^0_A$.* ■

The next lemma shows that the correspondence is maintained by transitions.

Lemma 3.2.8 *Let A be a compact rectangular automaton with uniform dynamics. For every reachable state $(v, \mathbf{y})$ of N_A, and every $\pi \in E \cup \mathbb{R}_{\geq 0}$,*

$$\xi_A(Post^{\pi}_{N_A}(v, \mathbf{y})) = Post^{\pi}_{A}(\xi_A(v, \mathbf{y})).$$

Proof. Suppose $\pi = \delta \in \mathbb{R}_{\geq 0}$. To show that $\xi_A(Post^{\delta}_{N_A}(v, \mathbf{y})) = Post^{\delta}_{A}(\xi_A(v, \mathbf{y}))$, it suffices to look at one coordinate at a time, or, equivalently, to assume that A is one-dimensional. We do the latter, so N_A is two-dimensional. If $Post^{\delta}_{N_A}(v, y, y')$ is nonempty, then

$$Post^{\delta}_{N_A}(v, y, y') = \{(v, y + k\delta, y' + k'\delta)\},$$

where $act_A(v) = [k, k']$. Since $\xi_A(v, y, y') = \{v\} \times [y, y']$, it is easy to see that $Post^{\delta}_{A}(\xi_A(v, y, y')) = \{v\} \times (inv_A(v) \cap [y + k\delta, y' + k'\delta])$. It follows that, as desired, $\xi_A(Post^{\delta}_{N_A}(v, y, y')) = Post^{\delta}_{A}(\xi_A(v, y, y'))$.

Now suppose that $Post^{\delta}_{N_A}(v, y, y')$ is empty. Inspecting the invariant $inv_{N_A}(v)$, we find that either $y + k\delta > \max inv_A(v)$ or $y' + k'\delta < \min inv_A(v)$. In either case it follows that $Post^{\delta}_{A}(v, y, y') = \emptyset$. We have proven that $\xi_A(Post^{\delta}_{N_A}(v, y, y')) = Post^{\delta}_{A}(\xi_A(v, y, y'))$ when $Post^{\delta}_{N_A}(v, y, y')$ is empty.

Now suppose $\pi = e = (v, w) \in E$. For each i, $[y_{\ell(i)}, y_{u(i)}] \cap inv(v)_i \cap preguard(e)_i$ is nonempty iff $y_{\ell(i)} \leq \max preguard_A(e)_i$ and $y_{u(i)} \geq \min preguard_A(e)_i$ (because $preguard(e) \subset inv(v)$ by assumption). Thus $\xi_A(Post^{e}_{N_A}(v, \mathbf{y}))$ is nonempty iff $Post^{e}_{A}(\xi_A(v, \mathbf{y}))$ is nonempty. So suppose $(v, \mathbf{y}) \xrightarrow{e} (w, \mathbf{z})$ in N_A, For each $i \in update(e)$, $[z_{\ell(i)}, z_{u(i)}] = postguard(e)_i$. For each $i \notin update(e)$,

$$\begin{aligned}
[z_{\ell(i)}, z_{u(i)}] &= [closest(y_{\ell(i)}, postguard_A(e)_i), closest(y_{u(i)}, postguard_A(e)_i)] \\
&= [y_{\ell(i)}, y_{u(i)}] \cap postguard_A(e)_i.
\end{aligned}$$

It follows that $\xi_A(Post^e_{N_A}(v, \mathbf{y})) = Post^e_A(\xi_A(v, \mathbf{y}))$. ■

Theorem 3.2.9 [HPV94] *For every compact rectangular automaton A with uniform dynamics, $L(S^{abs}_{N_A}) = L(S^{abs}_A)$.*

Proof. From Lemmas 3.2.7 and 3.2.8. ■

Theorem 3.2.10 [PV94] *The dense-time reachability problem for the class of compact rectangular automata with uniform dynamics is PSPACE-complete.*

Proof. It suffices to consider the case of *vertex reachability*, which singles out a vertex $\hat{v} \in V$, and asks if any state of the form $(\hat{v}, \mathbf{x})$ is reachable. Given A and a rectangular zone $Z \subset Q_A$, we define another rectangular automaton with uniform dynamics $\hat{A}$ which has a vertex $\hat{v}$, such that Z is reachable in S^{abs}_A iff $\{\hat{v}\} \times inv_{\hat{A}}(\hat{v})$ is reachable in $S^{abs}_{\hat{A}}$. The automaton $\hat{A}$ differs from A only in its control graph: $V_{\hat{A}} = V_A \cup \{\hat{v}\}$, where $\hat{v} \notin V_A$, and $E_{\hat{A}} = E_A \cup \{(w, \hat{v}) \mid w \in V_A\}$. The new edges are used to transfer control to $\hat{v}$ when entry to Z is detected. For each vertex $w \in V_A$, $preguard_{\hat{A}}(w, \hat{v}) = postguard_{\hat{A}}(w, \hat{v}) = [Z]_w$. Thus whenever Z is entered, an edge step may be taken to $\hat{v}$.

Therefore we assume that Z is of the form $\{v\} \times inv_A(v)$. By Lemmas 3.2.7 and 3.2.8, the zone Z is reachable in S^{abs}_A iff the zone $v \times inv_{N_A}(v)$ is reachable in $S^{abs}_{N_A}$. The latter can be solved in polynomial space, by Corollary 3.1.11. PSPACE-hardness follows from Theorem 3.1.8. ■

3.3 Skewed-Clock Translation: General Case

All of the main ideas are already present in the construction for compact automata: the extension to the general case (first reported in [HPV94]) is mostly a matter of bookkeeping. In particular, for each lower or upper bound multirate variable, one bit is used to distinguish a strict from a non-strict bound, and another bit is used to distinguish a finite from an infinite bound. The reader who has mastered the previous section will, with some care and effort, be able to carry the translation over to the general case. It is included here for completeness.

We encounter the following difficulties when the rectangular automaton A has non-compact components. (1) For each variable x, the lower and upper bounds represented by y_ℓ and y_u may be strict or weak (non-strict). To solve this problem, we introduce a bit called the *weak/strict bit* for each multirate variable. (2) Upper and lower bounds may be finite or infinite. For this, we introduce a bit called the *finite/infinite bit* for each multirate variable. The weak/strict and finite/infinite bitvectors are encoded in the discrete state. (3) Unbounded activity rectangles cause discontinuous jumps in upper and lower bounds. For example suppose the variable x is assigned the value 3 by traversal of edge $e = (v, w)$, where $act(w)(x) = [1, \infty)$. Then in M_A, both y_ℓ and y_u are assigned the value 3 by traversal of edge e. But after any positive amount of time passes, the upper bound on x should be ∞. We introduce a τ-edge called a *jump edge* which is taken before any positive time step. In this example, the jump edge sets the finite/infinite bit for y_u to inf. Since the result of the jump edge presupposes that some positive amount of time has passed, all edge transitions inherited from A are disabled until time passes. Implementing this restriction requires a new clock z and a bit called the *time passage bit*. (4) Open activity rectangles can cause a weak bound to change to strict after any positive amount of time passes. For example, suppose in the above case that $act(w)(x) = [1, 5)$. Then after edge e is traversed, the upper bound on x is a non-strict bound of 3. But after $t > 0$ time units pass, the upper bound is a strict bound of $3 + 5t$. Once again, we use the jump edge to solve this problem. When the jump edge is traversed, the weak/strict bit for y_u is set to *str*.

Let A be an n-dimensional rectangular automaton with uniform dynamics. We define a multirate automaton M_A with attractors and silent moves. Since the entire definition is lengthy, it is convenient to prove lemmas as soon as a sufficient portion of the definition is complete. The definition of M_A will be given in the text, and will not be numbered.

We first define the continuous variables of M_A, then the vertex set, and then an analogue to ξ_A. Next the activity and invariant are defined. Then come jump edges and edges inherited from A. Last, we define the initial zone.

For each variable x_i of A, the multirate automaton M_A has multirate variables $y_{\ell(i)}$ and $y_{u(i)}$ corresponding respectively to the lower bound and upper bound on x_i. We add a zeroth coordinate to M_A, so that the synchronization clock z is given by y_0 in M_A. Thus M_A is $(2n+1)$-dimensional. The event alphabet of M_A is that of A. The vertex set V_{M_A} is $V_A \times (\{0,1\}^{2n})^2 \times \{0,1\}$. A state of M_A is a tuple of the form $(v, \vec{\lambda}, \vec{\nu}, tp, \mathbf{y})$, where $\vec{\lambda}$ is the vector of finite/infinite bits (*fin* $= 0$, *inf* $= 1$), $\vec{\nu}$ is the vector of weak/strict bits bits (*wk* $= 0$, *str* $= 1$), *tp* is the time passage bit, and $\mathbf{y} \in \mathbb{R}^{2n+1}$ is the continuous state.

Relating the state spaces of M_A and A

We define a map $\eta_A : Q_{M_A} \rightarrow 2^{V_A \times \mathbb{R}^n}$, which specifies how the variables of M_A give the range of possible values for the variables of A. Let $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in Q_{M_A}$. The set $\eta_A(q)$ is a rectangle, so each component $\eta_A(q)_i$ is an interval, and hence is completely specified by its infimum, supremum, and which, if any, of the latter two it contains.

- If the finite/infinite bit $\lambda_{\ell(i)} = \mathit{inf}$, then $\inf \eta_A(q)_i = -\infty$.
- If the finite/infinite bit $\lambda_{\ell(i)} = \mathit{fin}$, then $\inf \eta_A(q)_i = y_{\ell(i)}$.
- The weak/strict bit $\nu_{\ell(i)}$ determines the strictness of the lower bound: if $\lambda_{\ell(i)} = \mathit{fin}$, then $\inf \eta_A(q) \in \eta_A(q)$ iff $\nu_{\ell(i)} = \mathit{wk}$.

We make the corresponding definitions for the upper bound.

- If the finite/infinite bit $\lambda_{u(i)} = \mathit{inf}$, then $\sup \eta_A(q)_i = \infty$.
- If the finite/infinite bit $\lambda_{u(i)} = \mathit{fin}$, then $\sup \eta_A(q)_i = y_{u(i)}$.
- The weak/strict bit $\nu_{u(i)}$ determines the strictness of the lower bound: if $\lambda_{u(i)} = \mathit{fin}$, then $\sup \eta_A(q) \in \eta_A(q)$ iff $\nu_{u(i)} = \mathit{wk}$.

Since the lower (resp. upper) bound multirate variables do not respect the lower (resp. upper) bounds of the invariant of A, $\eta_A(q) \not\subset Q_A$ for some states q of M_A. We remedy this deficiency by introducing a map $\beta_A : Q_{M_A} \rightarrow 2^{Q_A}$ defined by

$\beta_A(q) =_{\rm def} \eta_A(q) \cap Q_A$, except if $y_0 = 0$ and $tp = 1$, when $\beta_A(q) =_{\rm def} \emptyset$. We make the latter adjustment because the states in which $y_0 = 0$ and $tp = 1$ are only reached transiently after a jump edge before a positive time step—no edge inherited from A can be traversed from such a state. The *upper half-space* U_{M_A} of M_A is the zone of all states $q \in Q_{M_A}$ such that $\beta_A(q) \neq \emptyset$. We extend η_A and β_A to functions from $2^{Q_{M_A}}$ to 2^{Q_A} by $\eta_A(Z) = \bigcup_{q \in Z} \eta_A(q)$, and similarly for β_A. The truncation of η_A to β_A is justified by the following lemma, which follows from the assumption that the preguard of every edge is contained in the invariant of the source vertex.

Lemma 3.3.1 *For every $e \in E_A$, and every $q \in U_{M_A}$, $\eta_A(q) \cap preguard_A(e) \neq \emptyset$ iff $\beta_A(q) \cap preguard_A(e) \neq \emptyset$.* ■

For the rest of this section, i ranges over $\{1, \ldots, n\}$, j ranges over $\{1, \ldots, 2n\}$, v ranges over V_A, $\vec{\lambda}$ and $\vec{\nu}$ range over $\{0,1\}^{2n}$, and tp ranges over $\{0,1\}$. So when we quantify these variables, as in "for all $v, \vec{\lambda}, \vec{\nu}, tp, i, j$", the quantification is over the domain just specified for each variable.

Invariants

Let I be an interval. Define the *lower strictness* of I by

$$strict{\downarrow} I = \begin{cases} wk, & \text{if } \inf I \in I, \\ str, & \text{if } \inf I \notin I. \end{cases}$$

Define the *upper strictness* of I by

$$strict{\uparrow} I = \begin{cases} wk, & \text{if } \sup I \in I, \\ str, & \text{if } \sup I \notin I. \end{cases}$$

We now define the invariant function inv_{M_A}. Let $v, \vec{\lambda}, \vec{\nu}, tp, i$ be given. Then

$$inv_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_{\ell(i)} = \begin{cases} (-\infty, \infty), & \text{if } \lambda_{\ell(i)} = inf, \\ (-\infty, \sup inv_A(v)_i], & \text{if } \lambda_{\ell(i)} = fin \text{ and} \\ & \qquad \nu_{\ell(i)} = strict\!\uparrow inv_A(v)_i = wk, \\ (-\infty, \sup inv_A(v)_i), & \text{otherwise.} \end{cases}$$

If the finite/infinite bit $\lambda_{\ell(i)}$ is infinite, then the value of lower bound multirate variable $y_{\ell(i)}$ is irrelevant, so we do not constrain it. If the finite/infinite bit is finite, then the upper bound on this interval is strict unless both the strictness $\nu_{\ell(i)}$ of the lower bound multirate variable $y_{\ell(i)}$ and the upper strictness of the invariant are both weak. The motivation for this is that if I and J are intervals, and $\inf I = \sup J$, then $I \cap J \neq \emptyset$ iff $strict \downarrow I = strict \uparrow J = wk$. Here I is meant to represent the range of allowable values for x_i in A, as determined by the state of M_A, and J is meant to represent an invariant $inv_A(v)_i$. We have the corresponding definitions for the upper bounds. Define

$$inv_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_{u(i)} = \begin{cases} (-\infty, \infty), & \text{if } \lambda_{u(i)} = inf, \\ [\inf inv_A(v)_i, \infty), & \text{if } \lambda_{u(i)} = fin \text{ and} \\ & \qquad \nu_{u(i)} = strict\!\downarrow inv_A(v)_i = wk, \\ (\inf inv_A(v)_i, \infty), & \text{otherwise.} \end{cases}$$

The invariant of the synchronization clock is defined so that time may not pass if the time passage bit tp is 0: $inv_{M_A}(v, \vec{\lambda}, \vec{\nu}, 0)_0 = \{0\}$ and $inv_{M_A}(v, \vec{\lambda}, \vec{\nu}, 1)_0 = [0, \infty)$.

Activity rectangles

We now define the activity function act_{M_A}. For all $v, \vec{\lambda}, \vec{\nu}, tp, i$,

$$act_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_{\ell(i)} = \begin{cases} \{\inf act_A(v)_i\}, & \text{if } \inf act_A(v)_i > -\infty, \\ \{0\}, & \text{if } \inf act_A(v)_i = -\infty. \end{cases}$$

$$act_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_{u(i)} = \begin{cases} \{\sup act_A(v)_i\}, & \text{if } \sup act_A(v)_i < \infty, \\ \{0\}, & \text{if } \sup act_A(v)_i = \infty. \end{cases}$$

The slope of $y_{\ell(i)}$ in M_A is the infimum of the allowable slopes for x_i in A, unless it is infinite. The synchronization variable is a clock: $act_{M_A}(v, \vec{\lambda}, \vec{\nu}, tp)_0 = \{1\}$.

It remains to define the edges of M_A. The multirate automaton M_A has a family of attractor edges for each edge of A. We say that these are *inherited* from A. In addition, the multirate automaton M_A has a set of τ-edges called *jump edges*. The jump edges provide changes to bound value and strictness that are caused by the passage of any positive amount of time. For example, an unbounded activity always causes a discontinuous jump in the bound value. Such a jump can only be simulated by an edge transition.

Jump edges

For all $v, \vec{\lambda}, \vec{\nu}$, there is an edge from $(v, \vec{\lambda}, \vec{\nu}, 0)$ to $(v, \vec{\lambda}', \vec{\nu}', 1)$ with silent event τ. The target vertex components $\vec{\lambda}', \vec{\nu}'$ will be defined presently. The preguard and postguard are both $\{0\} \times \mathbb{R}^{2n}$, and so this edge can only be traversed when $y_0 = 0$. The update set is empty. Recall that jump edges provide changes that become necessary if any positive amount of time passes. These edges are taken proactively, before any time passes: i.e., only if the synchronization clock y_0 has value 0. To prevent spurious edges from being taken due to the changes made by these edges, no edge inherited from A may be traversed until a positive amount of time has passed, i.e., until $y_0 > 0$.

The finite/infinite bitvector must be changed to account for finite bounds that become infinite due to an unbounded activity.

$$\lambda'_{\ell(i)} = \begin{cases} inf, & \text{if } \inf act_A(v)_i = -\infty, \\ \lambda_{\ell(i)}, & \text{otherwise.} \end{cases}$$

$$\lambda'_{u(i)} = \begin{cases} inf, & \text{if } \sup act_A(v)_i = \infty, \\ \lambda_{u(i)}, & \text{otherwise.} \end{cases}$$

The weak/strict bitvector must be changed to account for weak bounds that become strict due to a strict activity.

$$\nu'_{\ell(i)} = \begin{cases} str, & \text{if } strict{\downarrow}\, act_A(v)_i = str, \\ \nu_{\ell(i)}, & \text{otherwise.} \end{cases}$$

$$\nu'_{u(i)} = \begin{cases} str, & \text{if } strict{\uparrow}\, act_A(v)_i = str, \\ \nu_{u(i)}, & \text{otherwise.} \end{cases}$$

The jump edges, just defined, play the following role in M_A. Suppose an edge inherited from A is taken. Then $tp = 0$. If no edge inherited from A is traversed, then before any time may pass (since no time may pass when $tp = 0$), a jump edge is traversed, setting tp to 1, and performing whatever bookkeeping is required for the weak/strict and finite/infinite bitvectors. The changes made by the jump edge reflect the situation after some positive amount of time has passed only. Therefore no edge inherited from A is allowed to be traversed before some time passes: if $tp = 1$ and $y_0 = 0$, then no inherited transitions are enabled.

Inherited edges

We now define the edges of M_A inherited from A. For each edge $e = (v, w) \in E_A$, every $\vec{\lambda}, \vec{\nu} \in \{0,1\}^{2n}$, and every $tp \in \{0,1\}$, there is a set $\Psi(e, \vec{\lambda}, \vec{\nu}, tp)$ of edges in M_A. Each edge $e' \in \Psi(e, \vec{\lambda}, \vec{\nu}, tp)$ has source vertex of the form $(v, \vec{\lambda}, \vec{\nu}, tp)$, and target vertex of the form $(w, \vec{\lambda}', \vec{\nu}', 0)$. Notice that the time passage bit is always cleared by e'. The event of e' is that of e: $event_{M_A}(e') = event_A(e)$. The update set is defined by $update_{M_A}(e') = \{0\} \cup \{\ell(i) \mid i \in update_A(e)\} \cup \{u(i) \mid i \in update_A(e)\}$. The postguard of e' is defined as in the compact case: Suppose $i \in update_A(e)$. Then $postguard_{M_A}(e')_{\ell(i)} = \{\inf postguard_A(e)_i\}$, when the infimum is finite, and $postguard_{M_A}(e')_{\ell(i)} = \{0\}$ if $\inf postguard_A(e)_i = -\infty$ (the definition in this case is arbitrary). Similarly, $postguard_{M_A}(e')_{u(i)} = \{\sup postguard_A(e)_i\}$, when the supremum is finite, and $postguard_{M_A}(e')_{u(i)} = \{0\}$ if $\sup postguard_A(e)_i = \infty$. If $i \notin update(e)$, then $postguard_{M_A}(e')_{\ell(i)} = postguard_{M_A}(e')_{u(i)} = cl(postguard_A(e)_i)$, where $cl(I)$ denotes the closure of I.

The preguards are more difficult to define, because of the requirement of updating strictness. Indeed, the proper propagation of strictness information is what requires there to be more than one edge in $\Psi(e, \vec{\lambda}, \vec{\nu}, tp)$. It is convenient to define the preguards, as well as the bitvector components of the source and target vertices of each edge $e' \in \Psi(e, \vec{\lambda}, \vec{\nu}, tp)$S, by means of a logical formula $\psi(e)$ in conjunctive normal form. When $\psi(e)$ is converted into disjunctive normal form, each disjunct specifies an edge e' by specifying the target bitvectors and a predicate on $\mathbb{R}^{2n+1}$ whose characteristic set is a rectangle. This set becomes the preguard of e'.

The formula $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ uses the following free variables: the symbols $y_{\ell(i)}$ and $y_{u(i)}$ to refer to the continuous variables of M_A; the variables $\vec{\lambda}'$ and $\vec{\nu}'$ refer to the values of the finite/infinite bitvector and the weak/strict bitvector after the edge is traversed. The formula $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ defines the set $[\![\psi(e, \vec{\lambda}, \vec{\nu}, tp)]\!] \subset ((\{0,1\}^{2n})^2 \times \mathbb{R}^{2n+1}$ of all pairs $((\vec{\lambda}, \vec{\nu}), \mathbf{y})$ satisfying $\psi(e)$. From $[\![\psi(e)]\!]$ we extract the preguard and the target values of the finite/infinite and weak/strict bitvectors.

The formula $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ is a conjunction of CNF formulas $\theta_{tp} \wedge \bigwedge_{i=1}^{n} \psi_i(e, \vec{\lambda}, \vec{\nu})$. The formula θ_0 is $y_0 = 0$, and formula θ_1 is $y_0 > 0 \wedge y_0' = 0$. Hence an edge derived from $\psi(e, \vec{\lambda}, \vec{\nu}, 0)$ can be traversed only if the synchronization clock has value 0, and an edge derived from $\psi(e, \vec{\lambda}, \vec{\nu}, 1)$ can be traversed only if the synchronization clock has value greater than 0. In the latter case the synchronization clock is reset to 0.

We wish an edge of M_A derived from the edge family $\Psi(e, \vec{\lambda}, \vec{\nu}, tp)$ to be enabled for traversal in state $q \in Q_{M_A}$ iff the range of possible values for each x_i intersects $\mathit{preguard}(e)_i$, i.e., iff $\eta_A(q) \cap \mathit{preguard}(e) \neq \emptyset$. It follows from the inclusion of edge preguards in source vertex invariants that $\eta_A(q) \cap \mathit{preguard}(e) \neq \emptyset$ iff $\beta_A(q) \cap \mathit{preguard}(e) \neq \emptyset$. If all values are finite and all bounds are weak, then the intersection is nonempty iff $y_{\ell(i)} \leq \max \mathit{preguard}(e)_i$ and $y_{u(i)} \geq \min \mathit{preguard}(e)_i$. This was the requirement given in the construction of N_A for compact A. Taking strictness and infinite bounds into account, we obtain the more complicated conditions $\ell guard(\vec{\lambda}, \vec{\nu}, i, preguard(e)_i)$ and $uguard(\vec{\lambda}, \vec{\nu}, i, preguard(e)_i)$. For an interval I, and $1 \leq i \leq n$, define

$$\ell guard(\vec{\lambda}, \vec{\nu}, i, I) = \begin{cases} true, & \text{if } \lambda_{\ell(i)} = inf \\ y_{\ell(i)} \leq \sup I, & \text{if } \lambda_{\ell(i)} = fin \text{ and } \nu_{\ell(i)} = strict{\uparrow} I = wk \\ y_{\ell(i)} < \sup I, & \text{otherwise} \end{cases}$$

$$uguard(\vec{\lambda}, \vec{\nu}, i, I) = \begin{cases} true, & \text{if } \lambda_{u(i)} = inf \\ y_{u(i)} \geq \inf I, & \text{if } \lambda_{u(i)} = fin \text{ and } \nu_{u(i)} = strict{\downarrow} I = wk \\ y_{u(i)} > \inf I, & \text{otherwise} \end{cases}$$

To understand this definition, consider the conditions under which an interval J intersects an interval I.

Lemma 3.3.2 *Let I and J be nonempty intervals, and let ϕ_ℓ and ϕ_u be defined as in Figure 3.3. Then $I \cap J \neq \emptyset$ iff ϕ_ℓ and ϕ_u are true.*

The formula $\ell guard(\vec{\lambda}, \vec{\nu}, i, I)$ corresponds to the predicate ϕ_ℓ and the formula $uguard(\vec{\lambda}, \vec{\nu}, i, I)$ corresponds to the predicate ϕ_u. Notice ϕ_ℓ is always satisfied if $\inf J = -\infty$. Thus $\ell guard(\vec{\lambda}, \vec{\nu}, i, I)$ is satisfied if $\lambda_{\ell(i)} = inf$. If $strict{\downarrow} J = strict{\uparrow} I = wk$, then ϕ_ℓ is $\inf J \leq \sup I$. Hence the second line of the definition of $\ell guard(\vec{\lambda}, \vec{\nu}, i, I)$. Finally, if either $strict{\downarrow} J = str$ or $strict{\uparrow} I = str$, then ϕ_ℓ is $\inf J < \sup I$. Hence the third line of the definition of $\ell guard(\vec{\lambda}, \vec{\nu}, i, I)$. Symmetrical remarks apply to $uguard(\vec{\lambda}, \vec{\nu}, i, I)$ and ϕ_u.

Lemma 3.3.3 *Let e be an edge of A, let $\vec{\lambda}, \vec{\nu} \in \{0,1\}^{2n}$, and for $tp = 0, 1$, let ψ_{tp} be the CNF expression*

$$\theta_{tp} \wedge \bigwedge_{i=1}^{n} (\ell guard(\vec{\lambda}, \vec{\nu}, i, preguard(e)_i) \wedge uguard(\vec{\lambda}, \vec{\nu}, i, preguard(e)_i)).$$

strict$\downarrow J$	*strict*$\uparrow I$	ϕ_ℓ
wk	*wk*	$\inf J \leq \sup I$
wk	*str*	$\inf J < \sup I$
str	*wk*	$\inf J < \sup I$
str	*str*	$\inf J < \sup I$

strict$\uparrow J$	*strict*$\downarrow I$	ϕ_u
wk	*wk*	$\sup J \geq \inf I$
wk	*str*	$\sup J > \inf I$
str	*wk*	$\sup J > \inf I$
str	*str*	$\sup J < \inf I$

Figure 3.5: $J \cap I \neq \emptyset$ iff $\phi_\ell \wedge \phi_u$

Then for every state $((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, $Post^e_A(\beta_A(q)) \neq \emptyset$ *iff* $\mathbf{y}$ *satisfies* ψ_{tp}.

Proof. Put $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y})$. In the discussion following Lemma 3.3.2, we proved that if q satisfies θ_{tp}, and $\eta_A(q) = \{v\} \times I$, then $I \cap preguard_A(e) \neq \emptyset$ iff q satisfies ψ_{tp}. So we may restrict attention to those states q for which $\beta_A(q) \neq \eta_A(q)$. There are two classes of such states. First, if $tp = 1$ and $y_0 = 0$, then $\beta_A(q) = \emptyset$. In this case, q does not satisfy θ_1, and so q does not satisfy ψ_1. Second, there is the class of states in which $\eta_A(q) \not\subset Q_A$, when, e.g., a lower bound multirate variable has dropped below the infimum of the invariant. In this case, the result follows from Lemma 3.3.1. ■

We continue the construction for $i \in update_A(e)$. In this case, the lower and upper bounds on x_i are assigned to the infimum and supremum of $postguard_A(e)_i$, with corresponding assignments made to the finite/infinite, off/on, and weak/strict bitvectors. For an interval I, and $1 \leq i \leq n$, define

$$\ell assign(i, I) = \begin{cases} \lambda'_{\ell(i)} = fin \;\wedge\; \nu'_{\ell(i)} = strict{\downarrow}\, postguard_A(e)_i & \text{if } \inf I \neq -\infty, \\ \lambda'_{\ell(i)} = inf \;\wedge\; \nu'_{\ell(i)} = str & \text{if } \inf I = -\infty. \end{cases}$$

$$uassign(i,I) = \begin{cases} \lambda'_{u(i)} = \mathit{fin} \ \wedge \ \nu'_{u(i)} = \mathit{strict}{\uparrow}\, postguard_A(e)_i & \text{if } \sup I \neq \infty, \\ \lambda'_{u(i)} = \mathit{inf} \ \wedge \ \nu'_{u(i)} = \mathit{str} & \text{if } \sup I = \infty. \end{cases}$$

For $i \in update_A(e)$, the formula $\psi_i(e, \vec{\lambda}, \vec{\nu})$ is

$$\begin{array}{lll} \ell guard(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i) & \wedge & uguard(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i) \quad \wedge \\ \ell assign(i, postguard_A(e)_i) & \wedge & uassign(i, postguard_A(e)_i). \end{array}$$

The remainder of the definition of $\psi(e, \vec{\lambda}, \vec{\nu}, tp)$ makes no mention of any $i \in update(e)$. Define $Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q) = \bigcup_{e' \in \Psi(e,\vec{\lambda},\vec{\nu},tp)} Post_{M_A}^{e'}(q)$, Recall that for a rectangular zone Z, we have the canonical decomposition $Z = \bigcup_{v \in V} \{v\} \times [Z]_v$, where $[Z]_v \subset \mathbb{R}$. We have the following lemma.

Lemma 3.3.4 *For every edge $e = (v, w)$ of A, every state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, and every $i \in update(e)$, $([Post_A^e(\beta_A(q))]_w)_i = ([\beta_A(Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q))]_w)_i$.* ■

So far, there has been no disjunction. Disjunction is used for $i \notin update(e)$. This case is complicated because the strictness of a bound may change, as well as its value. Our definitions follow once again from Lemma 3.3.2. The CNF formulas $\ell adjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i)$ and $uadjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i)$ give the conditions on $y_{\ell(i)}$ and $y_{u(i)}$ that necessitate the various changes in strictness. Let I be a nonempty interval. Define the formula $\ell adjust(\vec{\lambda}, \vec{\nu}, i, I)$ to be $\lambda'_{\ell(i)} =$

$\lambda_{\ell(i)} \wedge \nu'_{\ell(i)} = \nu_{\ell(i)}$,if $\inf I = -\infty$. Otherwise $\ell adjust(\vec{\lambda}, \vec{\nu}, i, I)$ is the formula

$$
\begin{aligned}
&(\lambda_{\ell(i)} = \mathit{inf} \wedge \lambda'_{\ell(i)} = \mathit{fin} \wedge \nu'_{\ell(i)} = \mathit{strict}{\downarrow} I) \\
&\vee (\lambda_{\ell(i)} = \mathit{fin} \wedge y_{\ell(i)} < \inf I \wedge \lambda'_{\ell(i)} = \mathit{fin} \wedge \nu'_{\ell(i)} = \mathit{strict}{\downarrow} I) \\
&\vee (\lambda_{\ell(i)} = \mathit{fin} \wedge y_{\ell(i)} = \inf I \wedge \nu_{\ell(i)} = \mathit{wk} \wedge \lambda'_{\ell(i)} = \mathit{fin} \wedge \nu'_{\ell(i)} = \mathit{strict}{\downarrow} I) \\
&\vee (\lambda_{\ell(i)} = \mathit{fin} \wedge y_{\ell(i)} = \inf I \wedge \nu_{\ell(i)} = \mathit{str} \wedge \lambda'_{\ell(i)} = \mathit{fin} \wedge \nu'_{\ell(i)} = \nu_{\ell(i)}) \\
&\vee (\lambda_{\ell(i)} = \mathit{fin} \wedge y_{\ell(i)} > \inf I \wedge \lambda'_{\ell(i)} = \mathit{fin} \wedge \nu'_{\ell(i)} = \nu_{\ell(i)}).
\end{aligned}
$$

Similarly, define the formula $uadjust(\vec{\lambda}, \vec{\nu}, i, I)$ to be $\lambda'_{u(i)} = \lambda_{u(i)} \wedge \nu'_{u(i)} = \nu_{u(i)}$ if $\sup I = \infty$. Otherwise $uadjust(\vec{\lambda}, \vec{\nu}, i, I)$ is the formula

$$
\begin{aligned}
&(\lambda_{u(i)} = \mathit{inf} \wedge \lambda'_{u(i)} = \mathit{fin} \wedge \nu'_{u(i)} = \mathit{strict}{\downarrow} I) \\
&\vee (\lambda_{u(i)} = \mathit{fin} \wedge y_{u(i)} > \sup I \wedge \lambda'_{u(i)} = \mathit{fin} \wedge \nu'_{u(i)} = \mathit{strict}{\uparrow} I) \\
&\vee (\lambda_{u(i)} = \mathit{fin} \wedge y_{u(i)} = \sup I \wedge \nu_{u(i)} = \mathit{wk} \wedge \lambda'_{u(i)} = \mathit{fin} \wedge \nu'_{u(i)} = \mathit{strict}{\uparrow} I) \\
&\vee (\lambda_{u(i)} = \mathit{fin} \wedge y_{u(i)} = \sup I \wedge \nu_{u(i)} = \mathit{str} \wedge \lambda'_{u(i)} = \mathit{fin} \wedge \nu'_{u(i)} = \nu_{u(i)}) \\
&\vee (\lambda_{u(i)} = \mathit{fin} \wedge y_{u(i)} < \sup I \wedge \lambda'_{u(i)} = \mathit{fin} \wedge \nu'_{u(i)} = \nu_{u(i)}).
\end{aligned}
$$

Explanation of these definitions is deferred to the next paragraph. For $i \notin update_A(e)_i$, the formula $\psi_i(e, \vec{\lambda}, \vec{\nu}, tp)$ is

$$
\begin{aligned}
&\ell guard(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i) \quad \wedge \quad \ell adjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i) \quad \wedge \\
&uguard(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i) \quad \wedge \quad uadjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i).
\end{aligned}
$$

The $\ell guard$ and $uguard$ conjuncts ensure that the edge can be taken iff the interval defined by the lower and upper bounds and the finite/infinite and weak/strict bits intersects the preguard interval $preguard_A(e)_i$. The $\ell adjust$ and $uadjust$ conjuncts reset the lower and upper bound values and their finite/infinite and weak/strict bits based upon the new information learned about the value of x_i if the edge e is traversed in A.

We now examine the definition of $\ell adjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i)$. If edge e is traversed in A, then new information about the value of x_i is obtained, namely that it lies within $preguard_A(e)_i$. Put $k = \inf preguard_A(e)_i$. If $k = -\infty$, then there is no new information, and so neither $\vec{\lambda}$ nor $\vec{\nu}$ changes. Hence the first line of the definition. If $k > -\infty$, then we have several cases. If the finite/infinite bit $\lambda_{\ell(i)} = inf$, then the present lower bound is infinite, and so $\lambda_{\ell(i)}$ must be set to *fin*, $y_{\ell(i)}$ must be reassigned to k, and the weak/strict bit $\nu_{\ell(i)}$ must be assigned to the lower strictness of $preguard_A(e)_i$ (line two). Now suppose the finite/infinite bit $\lambda_{\ell(i)} = fin$. If $y_{\ell(i)} < k$, then again $y_{\ell(i)}$ and $\nu_{\ell(i)}$ must be reset to k and its strictness (line three). If $y_{\ell(i)} = k$ and the strictness bit $\nu_{\ell(i)} = wk$, then information is gained if the lower strictness of $preguard_A(e)_i$ is str. So in this case (line four) we require $\nu'_{\ell(i)} = strict{\downarrow} preguard_A(e)_i$. But if $y_\ell = k$ and the strictness bit $\nu_{\ell(i)} = str$, then no information is gained; and so $\nu_{\ell(i)}$ is not changed. Finally, if $y_{\ell(i)} > k$, then there is no new information, and so there are no changes (line 6). We have proven the following lemma.

Lemma 3.3.5 *For every edge $e = (v, w)$ of A, every state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, and every $i \notin update(e)$,*

$$([Post_A^e(\beta_A(q))]_w)_i = ([\beta_A(Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q))]_w)_i. \blacksquare$$

Putting together Lemmas 3.3.3, 3.3.4, and 3.3.5, we have the following lemma.

Lemma 3.3.6 *Let A be a rectangular automaton with uniform dynamics. For every state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in U_{M_A}$, and every edge e of A,*

$$Post_A^e(\beta_A(q)) = \beta_A(Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q)).$$

Moreover, $|Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q)| \leq 1$*. That is,* $Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q)$ *is either empty or a singleton.*

Proof. The first statement follows from Lemmas 3.3.3, 3.3.4, and 3.3.5, and from the fact that $Post_{M_A}^{\Psi(e,\vec{\lambda},\vec{\nu},tp)}(q) \neq \emptyset$ iff q satisfies the predicate ψ_{tp} from Lemma 3.3.3. For the second claim, notice that the disjuncts of $\ell adjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i)$ are mutually exclusive, as are the disjuncts of $uadjust(\vec{\lambda}, \vec{\nu}, i, preguard_A(e)_i)$. Each state q can satisfy at most one of the disjuncts of each of these formulas. ■

Initial zone

It remains to define the initial zone $Q^0_{M_A}$ in such a way that $\beta_A(Q^0_{M_A}) = Q^0_A$. This is done by setting $y_{\ell(i)}$ and $y_{u(i)}$ at vertex v to be the infimum and supremum of the region of $Q^0_{M_A}$ associated with v. Let $Z \subset Q_A$ be a rectangular zone. Each $\{v\} \times [Z]_v$ contributes a singleton zone $\{(v, \vec{\lambda}([Z]_v), \vec{\nu}([Z]_v), 0, \mathbf{y}([Z]_v))\}$ to $lowhigh\, Z$. If $\inf([Z]_v)_i = -\infty$, then $\lambda([Z]_v)_{\ell(i)} = inf$, $\nu([Z]_v)_{\ell(i)} = str$, and $y([Z]_v)_{\ell(i)} = 0$. If $\inf([Z]_v)_i \neq -\infty$, then $\lambda([Z]_v)_{\ell(i)} = fin$, $\nu([Z]_v)_{\ell(i)} = strict\!\downarrow([Z]_v)_i$, and $y([Z]_v)_{\ell(i)} = (\inf[Z]_v)_i$. Similarly, if $\sup([Z]_v)_i = \infty$, then $\lambda([Z]_v)_{u(i)} = inf$, $\nu([Z]_v)_{u(i)} = str$, and $y([Z]_v)_{u(i)} = 0$. If $\sup([Z]_v)_i \neq \infty$, then $\lambda([Z]_v)_{u(i)} = fin$, $\nu([Z]_v)_{u(i)} = strict\!\uparrow([Z]_v)_i$, and $y([Z]_v)_{u(i)} = \sup([Z]_v)_i$. Finally, $y_0 = 0$. Now define $Q^0_{M_A}$ to be $lowhigh\, Q^0_A$.

Lemma 3.3.7 *Let A be a rectangular automaton with uniform dynamics. For every rectangular zone $Z \subset Q_A$, $\beta_A(lowhigh\, Z) = Z$. In particular, $\beta_A(Q^0_{M_A}) = Q^0_A$.*

This completes the definition of M_A. Notice that M_A has deterministic updates, and so it is a multirate automaton with attractors. The automaton M_A has exponentially many more vertices and edges than A. However, this exponential blowup does not adversely affect the complexity of the reachability problem, because the size of the bisimilarity quotient remains singly exponential in the size of A.

Analysis of time steps

For the remainder of this section, it will be convenient to refer to the components of a state q of M_A generically as v, $\vec{\lambda}$, $\vec{\nu}$, tp, and $\mathbf{y}$. We say "$\nu_{\ell(i)}$ in q" or "$\nu_{\ell(i)}$ in q'" to distinguish components of different states. Lemma 3.3.6 gives the basic correspondence between edge steps in A and edge steps in M_A. We must now develop a correspondence for time steps. The next lemma simply says that every reachable state of M_A that is the target of a time-step has its finite/infinite and weak/strict bits set correctly.

Lemma 3.3.8 *For every reachable state $q = ((v, \vec{\lambda}, \vec{\nu}, tp), \mathbf{y}) \in Reach_{M_A} \cap U_{M_A}$ with $y_0 > 0$ and $tp = 1$, and for every i,*

1. *if* $\inf act_A(v)_i = -\infty$ *then* $\lambda_{\ell(i)} = inf$*; if* $\sup act_A(v)_i = \infty$ *then* $\lambda_{u(i)} = inf$.
2. *if* $strict{\downarrow}\, act_A(v) = str$, *then* $\nu_{\ell(i)} = str$*; if* $strict{\uparrow}\, act_A(v) = str$, *then* $\nu_{u(i)} = str$. ■

Recall that a multirectangular zone is a finite union of rectangular zones.

Proposition 3.3.9 *For every rectangular automaton A, every multirectangular zone $Z \subset Q_A$, and every label $\pi \in \mathbb{R}_{\geq 0} \cup \Sigma^\tau \cup E$, $Post_A^\pi(Z)$ is a multirectangular zone.*

Proof. Since each relation $\xrightarrow{\sigma}$ with $\sigma \in \Sigma^\tau$ is a finite union of relations $\xrightarrow{e}$ with $e \in E$, it suffices to prove the proposition for $\pi \in \mathbb{R}_{\geq 0} \cup E$. Call a zone *elementary* if it is of the form $\{v\} \times R$, where R is a rectangular region. Then a zone is multirectangular iff it is a finite union of elementary zones. We show that for any elementary zone $Z' = \{v\} \times R$, $Post_A^\pi(Z')$ is elementary. If $\pi = (v, w) \in E$, then $Post_A^\pi(Z') = \{w\} \times R'$, where

$$R'_i = \begin{cases} postguard(\pi)_i, & \text{if } i \in update(\pi)_i, \\ R_i, & \text{if } i \notin update(\pi)_i. \end{cases}$$

If $\pi \in \mathbb{R}_{\geq 0}$, then $Post_A^\pi(Z') = \{v\} \times R'$, where R' is a rectangular region satisfying

$$\inf R'_i = \max\{\inf inv(v)_i, \inf R_i + \pi \cdot \inf act(v)\}$$

and

$$\sup R'_i = \min\{\sup inv(v)_i, \sup R_i + \pi \cdot \sup act(v)\},$$

where we have used the convention that $0 \cdot \infty = 0 \cdot (-\infty) = 0$. ∎

The following lemma gives the time-step correspondence between A and M_A.

Lemma 3.3.10 *Let A be a rectangular automaton with uniform dynamics. Then for every reachable state $q \in Reach_{M_A} \cap U_{M_A}$, and every duration $t \in \mathbb{R}_{\geq 0}$,*

$$Post_A^t(\beta_A(q)) = \beta_A(Post_{M_A}^t(q)).$$

Moreover, $|Post^t_{M_A}(q) \cap U_{M_A}| \leq 1$*. That is,* $Post^t_{M_A}(q) \cap U_{M_A}$ *is either empty or a singleton.*

Proof. The second claim is the determinism of the time-step relation. It follows from the fact that M_A is initialized and its continuous variables are multirate variables: they take on only one rate between initializations. Let $q \in U_{M_A}$ be a reachable state, and let $\beta_A(q) = \{v\} \times B$.

Case 1: $t = 0$**.** In this case $Post^t_A(\beta_A(q)) = \beta_A(q)$. Every $q' \in Post^t_{M_A}(q)$ is either q itself, when obviously $\beta_A(q') = \beta_A(q)$, or the target of a jump edge, when $\beta_A(q') = \emptyset$. Hence $Post^0_A(\beta_A(q)) = \beta_A(q) = \beta_A(Post^0_{M_A}(q))$.

Case 2: $t > 0$ **and** $Post^t_A(\beta_A(q)) \neq \emptyset$**.** Recall from Proposition 3.3.9 that each $([Post^t_A(\beta_A(q))]_v)_i$ is a nonempty interval with

$$\inf Post^t_A(\beta_A(q))_i = \max\{\inf inv_A(v)_i, \inf B_i + t \cdot \inf act_A(v)_i\}, \tag{3.1}$$

and

$$\sup Post^t_A(\beta_A(q))_i = \min\{\sup inv_A(v)_i, \sup B_i + t \cdot \sup act_A(v)_i\}. \tag{3.2}$$

The strictness of the infimum is given as follows. Put $Inv = inv_A(v)_i$, $Act = act_A(v)_i$, and $Try = \inf B_i + t \cdot \inf act_A(v)_i$. If $\inf Inv > Try$, then $strict\downarrow Post^t_A(\beta_A(q)) = strict\downarrow Inv$. If $\inf Inv = Try$, then $strict\downarrow Post^t_A(\beta_A(q)) = wk$ iff $strict\downarrow Inv = strict\downarrow B_i = strict\downarrow Act = wk$. If $\inf Inv < Try$, then $strict\downarrow Post^t_A(\beta_A(q)) = wk$ iff $strict\downarrow B_i = strict\downarrow Act = wk$. The strictness of the supremum is given symmetrically.

There is exactly one state $q' \in Post^t_{M_A}(q)$. We prove $\beta_A(q') = Post^t_A(\beta_A(q))$.

Subcase 2a: $\beta_A(q)$ and $act_A(v)$ are bounded. The upper bound clock $y_{u(i)}$ moves at the supremum of the allowable rates for x_i in A. If this rate is positive then the upper bound reaches the upper boundary of $inv_A(v)_i$ after $\frac{\sup inv_A(v)_i - \sup B_i}{\sup act_A(v)_i}$ units of time pass, Hence

$$\sup \beta_A(q') = \min\{\sup inv_A(v)_i, \sup B_i + t \cdot \sup act_A(v)_i\},$$

which is the same as $\sup Post^t_A(\beta_A(q))_i$. Similarly,

$$\inf \beta_A(q') = \max\{\inf inv_A(v)_i, \inf B_i + t \cdot \inf act_A(v)_i\},$$

which is the same as $\inf Post^t_A(\beta_A(q))_i$. So the infimum and supremum of $\beta_A(q')_i$ coincide with the infimum and supremum of $post^t_A(\beta_A(q))_i$. The question of strictness remains. If $\sup B_i + t \cdot \sup act_A(v)_i > \sup inv_A(v)_i$, then $strict\!\uparrow \beta_A(q')_i = strict\!\uparrow inv_A(v)_i = strict\!\uparrow Post^t_A(\beta_A(q))$. The strictness is correct. Now suppose $\sup B_i + t \cdot \sup act_A(v)_i < \sup inv_A(v)_i$. Then in state q', the upper bound $y_{u(i)}$ has value $\sup B_i + t \cdot \sup act_A(v)_i$. Since q' is reachable, Lemma 3.3.8 implies that in q', $\nu_{u(i)} = wk$ iff $strict\!\uparrow \beta_A(q)_i = strict\!\uparrow act_A(v)_i = wk$. A glance at the discussion of strictness following Equation 3.2 shows that the strictness is correct: $strict\!\uparrow \beta_A(q')_i = strict\!\uparrow Post^t_A(\beta_A(q))$. Finally, suppose $\sup B_i + t \cdot \sup act_A(v)_i = \sup inv_A(v)_i$. In this case, again we have that in q', $\nu_{u(i)} = wk$ iff $strict\!\uparrow \beta_A(q)_i = strict\!\uparrow act_A(v)_i = wk$. But here the definition of β_A, which intersects the value of η_A with the invariant, comes to the fore, resulting in a strict bound if $strict\!\uparrow inv_A(v)_i = str$. Therefore $strict\!\uparrow \beta_A(q')_i = wk$ iff $strict\!\uparrow \beta_A(q)_i = strict\!\uparrow act_A(v)_i = strict\!\uparrow inv_A(v)_i = wk$. The discussion of strictness following Equation 3.2 shows that the strictness is correct: $strict\!\uparrow \beta_A(q')_i = strict\!\uparrow Post^t_A(\beta_A(q))$.

Symmetrical remarks apply to the lower bound multirate variable, and we have completed the discussion of strictness.

Subcase 2b: $\beta_A(q)$ is not bounded from above. In this case, $inv_A(v)$ is not bounded from above, either, and the finite/infinite bit $\lambda_{u(i)}$ is *inf* in state q. The jump edges do not change this bit when $inv_A(v)$ is unbounded from above, and so in state q', still $\lambda_{u(i)}$ is *inf*, and so $\sup \beta_A(q')_i = \infty = \sup Post^t_A(\beta_A(q))$. Symmetrical remarks apply to the case of $\beta_A(q)$ not bounded from below.

Subcase 2c: $act_A(v)_i$ is not bounded from above. Since $t > 0$, $\lambda_{u(i)} = inf$ in q' by Lemma 3.3.8. So $\sup \beta_A(q')_i = \sup inv_A(v)_i = \sup Post^t_A(\beta_A(q))_i$, with matching strictnesses. A symmetrical argument handles the lower bound. We conclude that $\beta_A(q')_i = Post^t_A(\beta_A(q))_i$. The case of $t > 0$ and $Post^t_A(\beta_A(q)) \neq \emptyset$ is complete.

Case 3: $Post^t_A(\beta_A(q)) = \emptyset$. This means that for each coordinate i, either the lower bound $y_{\ell(i)}$ rises above the upper boundary of $inv_A(v)_i$ within time t, or the upper bound $y_{u(i)}$ drops below the lower boundary of $inv_A(v)_i$ within time t. Put $bottom(i) = \max\{\inf \inf_A(v)_i, \inf B_i + t \cdot \inf act_A(v)_i\}$ (see Equation 3.1) and $top(i) = \min\{\sup inv_A(v)_i, \sup B_i + t \cdot \sup act_A(v)_i\}$ (see Equation 3.2). That is, $bottom(i)$ (resp. $top(i)$) would equal $\inf Post^t_A(\beta_A(q))_i$ (resp. $\sup Post^t_A(\beta_A(q))_i$), if only $Post^t_A(\beta_A(q))_i$ were nonempty. The fact that $Post^t_A(\beta_A(q))_i = \emptyset$ means that either $bottom(i) > \sup inv_A(v)_i$ or $top(i) > \inf inv_A(v)_i$, or we have equality in one of these two expressions with a strictness conflict. If $bottom(i) > \sup inv_A(v)_i$, then q cannot take a $\xrightarrow{t}$ transition, because any state q' with $q \xrightarrow{t} q'$ has the lower bound clock $y_{\ell(i)} > \sup inv_A(v)_i$, which is impossible, since for any such

q', $\sup inv_{M_A}(q')_{\ell(i)} = \sup inv_A(v)_i$ (see the definition of inv_{M_A}). So in this case $Post^t_{M_A}(q) = \emptyset$, and so $\beta_A(Post^t_{M_A}(q)) = \emptyset = Post^t_A(\beta_A(q))$ as desired. We have a similar deduction for $top(i) < \inf inv_A(v)$. Now suppose $bottom(i) = \sup inv_A(v)_i$. There are two possible strictness conflicts. (1) The invariant upper boundary is strict. (2) The lower bound multirate variable is strict. In either case, the invariant inv_{M_A} places $\sup inv_A(v)_i$ out of the reach of $y_{\ell(i)}$: the supremum of the invariant for $y_{\ell(i)}$ is $\sup inv_A(v)_i$, but the supremum is not contained in the invariant interval (the reader is encouraged to reread the second line of the definition of inv_{M_A}). The proof is complete. ■

Theorem 3.3.11 [HPV94] *For every rectangular automaton A with uniform dynamics, $L(S^{abs}_{M_A}) = L(S^{abs}_A)$.*

Theorem 3.3.12 [HPV94] *The dense-time reachability problem for the class of rectangular automata with uniform dynamics is PSPACE-complete.*

Proof. Let A be an initialized rectangular automaton. As before, reachability can be reduced to vertex reachability. By Lemmas 3.3.6 and 3.3.10, for every rectangular zone Z, $Post^*_A(\beta_A(Z)) = \beta_A(Post^*_{M_A}(Z))$. Thus we may reduce the vertex reachability problem from v to w in A to a reachability problem in M_A from a set of vertices of the form $\{v\} \times (\{0,1\}^{2n})^2 \times \{0,1\}$ to a set of vertices of the form $\{w\} \times (\{0,1\}^{2n})^2 \times \{0,1\}$. This can be solved in space

$$O(\log((2n+1)!^2 |V_{M_A}| (k+2)^{2n+1})),$$

where k is singly exponential in the size of A; this is polynomial in the size of A. The (exponentially large) automaton M_A need not be explicitly constructed. ■

3.4 Initialized Rectangular Automata

In this section we show that the class of initialized rectangular automata is essentially equivalent to the class of rectangular automata with uniform dynamics. It follows that the reachability problem for the former class is PSPACE-complete. In the next section, we show that if either rectangularity or initialization is relaxed, then even the simplest automata can encode undecidable problems. It follows that the class of initialized rectangular automata is essentially a maximal class for which the reachability problem is decidable.

Definition 3.4.1 [PV94] An n-dimensional rectangular automaton A is *initialized* if for every edge $e = (v, w) \in E$, and every $1 \leq i \leq n$, if $act(v)_i \neq act(w)_i$ then $i \in update(e)$. ■

Thus if A is initialized, then whenever the continuous dynamics of variable i change due to a change in control mode, the value of variable i is nondeterministically reinitialized. It follows that variable i can be replaced with several variables, one for each distinct $act(v)_i$.

Corollary 3.4.2 *The dense-time reachability problem for the class of initialized rectangular automata is PSPACE-complete.*

Proof. Let A be an n-dimensional initialized rectangular automaton. We define an m-dimensional rectangular automaton with uniform dynamics A', where $m \leq n|V_A|$, with the same control graph as A, and such that $\{v\} \times \mathbb{R}^n$ is reachable in A iff $\{v\} \times \mathbb{R}^m$ is reachable in A'. For each $1 \leq i \leq n$, let $F_i = \{act_A(v)_i \mid v \in V\}$, the set of all activity intervals for variable i in A. The set of variables of A' is indexed by $\{(i, I) \mid 1 \leq i \leq n \text{ and } I \in F_i\}$. The variable (i, I) plays the role of variable i in the control modes v such that $act_A(v)_i = I$. When $act_A(v)_i \neq I$, the behavior of variable (i, I) is unconstrained.

The activity function of A' is defined by $act_{A'}(v)_{(i,I)} = I$ for each (i, I). Suppose $I = act_A(v)_i$. Then the invariant condition $inv_{A'}(v)$ is defined by $inv_{A'}(v)_{(i,I)} = inv_A(v)_i$. If $I \neq act_A(v)_i$, then $inv_{A'}(v)_{(i,I)} = \mathbb{R}$.

For each edge $e = (v, v') \in E$, $preguard_{A'}(e)_{(i,I)} = preguard_A(e)_i$ if $act_A(v)_i = I$, and $preguard_{A'}(e)_{(i,I)} = \mathbb{R}$ otherwise; $postguard_{A'}(e)_{(i,I)} = postguard_A(e)_i$ if $act_A(v')_i = I$, and $postguard_{A'}(e)_{(i,I)} = \mathbb{R}$ otherwise; $update_{A'}(e)$ is the set of (i, I) such that $i \in update_A(e)$ and $act_A(v')_i = I$.

The initial condition function of A' is defined by $init_{A'}(v)_{(i,I)} = init_A(v)_i$ if $act_A(v)_i = I$, and $init_{A'}(v)_{(i,I)} = \{0\}$ otherwise (the latter being arbitrary). It follows that for every vertex $v \in V$, $\{v\} \times \mathbb{R}^n$ is reachable in A iff $\{v\} \times \mathbb{R}^m$ is reachable in A'. Thus the reachability problem for initialized rectangular automata is PSPACE-complete. ■

Chapter 4

Reachability: Undecidability

In the previous chapter, we showed that initialized rectangular automata form a decidable class of hybrid automata. In this chapter, we show that they form a maximal such class. We proceed in two steps. First, we show that without initialization, even a single two-slope variable leads to an undecidable reachability problem. Second, we show that the rectangularity of the model must remain inviolate. Any coupling between coordinates, such as comparisons between variables, brings undecidability with even a single non-clock variable. (Timed automata, which have only clock variables, remain decidable in the presence of variable comparisons [AD94].) A main consequence is the undecidability of compact automata with clocks and one stopwatch. These automata are important for the verification of duration properties. The undecidability of the reachability problem for rectangular automata with clocks and several stopwatches was previously known [ACHH93, KPSY93]. However several decidability results for restricted classes of automata with all but one clock presented some hope, now lost, that a limited number of stopwatches could be accommodated [ACH93,BES93,KPSY93,BER94b].

Definition 4.0.3 Let A be a rectangular automaton and let a be a continuous variable of A. Variable a is a *clock* if for every vertex $v \in V$, $act_A(v)(a) = 1$. Variable a is a *skewed clock* if there exists a slope $s \in \mathbb{Q} \setminus \{0, 1\}$ such that for every vertex $v \in V$, $act_A(v)(a) = s$. Variable a is a *memory cell* if for every vertex $v \in V$, $act_A(v)(a) = 0$. Variable a is a *two-slope variable* if there exist slopes $s_1, s_2 \in \mathbb{Q}$ such that for every vertex $v \in V$, either $act_A(v)(a) = s_1$ or $act_A(v)(a) = s_2$. ■

Definition 4.0.4 A rectangular automaton A is *simple* if it meets the following restrictions:

1. Exactly one variable of A is *not* a clock.
2. For every vertex v, $inv(v)$ and $act(v)$ are compact.
3. For every edge $e \in E$, and for all $1 \leq i \leq n$, if $i \in update(e)$ then $postguard(e)_i = \{0\}$, and if $i \notin update(e)$ then $postguard(e)_i = preguard(e)_i$.
4. For every edge $e \in E$, $preguard(e)$ is compact (and hence $postguard(e)$ is compact by 3).

The automaton A is *m-simple* if it meets restrictions 2–4, and exactly m variables of A are not clocks. ■

We use simple automata for our undecidability results. Restriction 3 allows only deterministic variable updates, and so the nondeterminism of jumps in the continuous state, allowed in our model of rectangular automata, does not contribute to our undecidability results. Many limited decidability results are based on the digitization of real-numbered delays [HMP92,BES93,BER94b,PV94]. Since the digitization technique requires closed guards and invariants, restrictions 2, 3, and 4 imply that the technique does not generalize beyond very special cases. Many other limited decidability results apply to automata with a single stopwatch [ACH93,BES93,KPSY93,BER94b,BR95,Hen95]. Restriction 1 implies that these results do not generalize either. We might also replace condition 2 with the trivial invariant $\lambda v \in V. \mathbb{R}^n$, when our proofs would require only minor modification.

All of our undecidability proofs are reductions from the halting problem for two-counter machines to the reachability problem for simple rectangular automata. A two-counter machine consists of a finite control and two unbounded counters. Three types of instructions are used: branching based upon whether a specific counter has value 0, incrementing a counter, and decrementing a counter (which leaves unchanged a counter value of 0). In our reductions, each counter is modeled by a clock. Counter value r (usually) corresponds to clock value $s_1(\frac{s_2}{s_1})^r$, where s_1 and s_2 are the slopes of a two-slope variable in a simple automaton, s_1 being the larger. When $\frac{s_1}{s_2} = 2$, decrementing (resp. incrementing) a counter corresponds to doubling (resp. halving) the value of the corresponding clock. Notice that since $s_1 > s_2$, it is the density of the continuous domain, rather than its infinite extent, that is used to code the potentially unbounded counter values.

4.1 Uninitialized Automata

We show that initialization is necessary for a decidable reachability problem.

Theorem 4.1.1 *For every two slopes $s_1, s_2 \in \mathbb{Q}$ with $s_1 \neq s_2$, the dense-time reachability problem is undecidable for the class of simple rectangular automata with one two-slope variable of slopes s_1 and s_2.*

We first prove three lemmas that are basic to all of our undecidability proofs. In figures of simple automata, all variables whose slopes are not listed are clocks—they have slope 1.

Definition 4.1.2 Let W be a positive rational number. A simple rectangular automaton A is *W-wrapping* if the following four conditions obtain:

1. for every clock variable a of A, and for every vertex v, $inv(v)(a) = [0, W]$,
2. if z is the non-clock variable of A, and z takes only nonnegative slopes (i.e., $act(v)(z) \subset [0, \infty)$ for each vertex v), then for each vertex v, $inv(v)(z) = [0, W \cdot \max_{w \in V} \max act(w)(z)]$,
3. if z is the non-clock variable of A, and z takes only nonpositive slopes, then for each vertex v, $inv(v)(z) = [W \cdot \min_{w \in V} \min act(w)(z), 0]$,
4. if z is the non-clock variable of A, and z takes both positive and negative slopes, then for each vertex v,

$$inv(v)(z) = [W \cdot \min_{w \in V} \min act(w)(z), W \cdot \max_{w \in V} \max act(w)(z)].$$

A W-*wrapping edge* for a variable a and a vertex v with $act(v)(a) = \{s\}$ is an edge e from v to itself with $preguard(e)(a) = \{sW\}$, $update(e) = \{a\}$, and $postguard(e)(a) = \{0\}$. ■

In particular, a W-*wrapping edge* for a clock a is an edge e from vertex v to itself such that $preguard(e)(a) = \{W\}$, $update(e) = \{a\}$, and $postguard(e)(a) = \{0\}$.

The invariant of a wrapping automaton forces wrapping edges to be taken when they are enabled. We use wrapping to simulate discrete events by continuous rounds taking W (or some multiple thereof) units of time. The wrapping edges ensure that variables take the same values at the beginning and end of a round, unless they are explicitly reassigned by a non-wrapping edge. This is the content of our first lemma. *We stress that in figures, we leave these wrapping conditions implicit*, in particular, we omit invariants from every figure after those regarding the basic lemmas, and we omit wrapping edges beginning with Figure 4.7. The wrapping technique originated in [Cer92].

Lemma 4.1.3 Wrapping. *Let W be a positive rational number. Let $s_1 \in \mathbb{Q}$, and consider the simple W-wrapping automaton fragment of Figure 4.1, Suppose that $c = \gamma$ when edge e_1 is traversed into v_1, where $0 < \gamma < s_1 W$ if $s_1 > 0$, and $s_1 W < \gamma < 0$ if $s_1 < 0$. Then the next time e_3 is traversed out of v_1, again $c = \gamma$.*

Proof. Figure 4.2 contains a time portrait illustrating the proof for $W = 4$ and $s_1 = 1$. The markings e_1, e_2, and e_3 along the time axis show at which points these edges are traversed. We give the proof for $s_1 > 0$. In order for e_3 to be taken in the future, the following series of steps must occur: 1) e_1 is traversed; 2) exactly

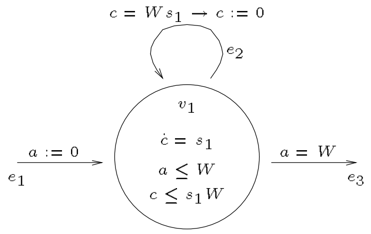

Figure 4.1: Wrapping lemma: the skewed clock c retains its entry value upon exit

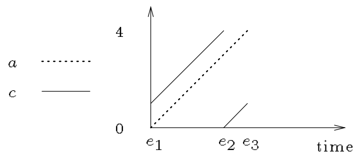

Figure 4.2: Proof of the Wrapping Lemma for $W = 4$, $s_1 = 1$

$\frac{1}{s_1}(Ws_1 - \gamma)$ units of time elapse, after which c has value Ws_1, and a has value $\frac{1}{s_1}(Ws_1 - \gamma)$; 3) the wrapping edge e_2 is traversed, after which c has value 0, and a has value $\frac{1}{s_1}(Ws_1 - \gamma)$; 4) exactly $W - \frac{1}{s_1}(Ws_1 - \gamma) = \frac{\gamma}{s_1}$ units of time elapse, after which a has value W and c has value γ. ■

By only allowing clocks c and d to wrap simultaneously, we can check if the two have the same value.

Lemma 4.1.4 Equality. *Let W be a positive rational number. Consider the simple W-wrapping automaton fragment of Figure 4.3, in which all variables are clocks. Suppose that $c = \gamma$ and $d = \delta$ when edge e_1 is traversed into v_1, where $0 < \gamma, \delta < W$. Then edge e_3 can later be traversed iff $\gamma = \delta$, and the next time e_3 is traversed, both c and d have value γ $(= \delta)$.* ■

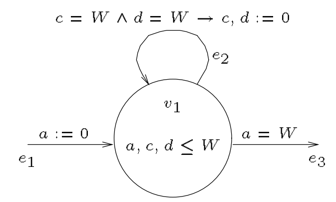

Figure 4.3: Equality lemma: testing $c = d$

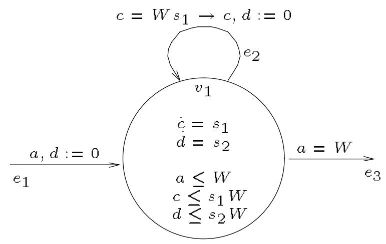

Figure 4.4: Assignment lemma: performing the assignment $d := \frac{s_2}{s_1} c$

Similarly, by assigning skewed clock d to 0 at the same time as wrapping skewed clock c to 0, we perform the assignment $d := \frac{s_2}{s_1} c$, where $\dot{c} = s_1$ and $\dot{d} = s_2$.

Lemma 4.1.5 Assignment. *Let W be a positive rational number. Let $s_1, s_2 \in \mathbb{Q}$, $s_1 \neq 0$, and consider the simple W-wrapping automaton fragment of Figure 4.4. (In Figure 4.4, it is assumed that $s_1, s_2 \geq 0$. If $s_1 < 0$, replace $c \leq s_1 W$ by $c \geq s_1 W$ in the figure. If $s_2 < 0$, replace $d \leq s_2 W$ by $d \geq s_2 W$ in the figure.) Suppose that $c = \gamma$ when edge e_1 is traversed into v_1, where $0 < \gamma < s_1 W$ if $s_1 > 0$, and $s_1 W < \gamma < 0$ if $s_1 < 0$. Then the next time e_3 is traversed, $c = \gamma$ and $d = \frac{s_2}{s_1}\gamma$.* ■

Proof of Theorem 4.1.1. We reduce the halting problem for two-counter machines to the reachability problem for simple wrapping rectangular automata with a two-slope variable taking slopes s_1 and s_2. Let M be a two-counter machine with counters C and D. Let a, b, c, and d be clocks, and let z be a two-slope variable of slopes s_1 and s_2.

Case 1: $s_1 > s_2 > 0$ **or** $s_1 < s_2 < 0$**.** Our automaton is W-wrapping, where W may be chosen to be any number larger than s_1. We encode the values of the counters C and D in the values of the clocks c and d, respectively. We encode counter value r by clock value $|s_1|(\frac{s_2}{s_1})^r$. The relationships $c = |s_1|(\frac{s_2}{s_1})^C$ and $d = |s_1|(\frac{s_2}{s_1})^D$ hold when $a = 0$ or $a = W$ (except when more than one time interval of duration W is needed to simulate one computation step). The test $C = 0$ is implemented by two edges e_1 and e_2, where $\mathit{preguard}(e_1)(c) = [s_1, s_1]$ (corresponding to $C = 0$) and $\mathit{preguard}(e_2)(c) = [0, s_2]$ (corresponding to $C \neq 0$). Decrementing a counter corresponds to first checking if it is zero as above, and if not, then multiplying the corresponding clock value by $\frac{s_1}{s_2}$. This is implemented by two assignment lemma constructions in series as in Figure 4.5. In the first, $\dot{z} = s_1$; it performs $z := s_1 c$. In the second, $\dot{z} = s_2$; it performs $c := \frac{1}{s_2} z$. The bottom portion of Figure 4.5 contains a time portrait showing the operation of the decrementation fragment with $W = 4$, $s_1 = 2$, and $s_2 = 1$. Incrementing a counter corresponds to multiplying the corresponding clock value by $\frac{s_2}{s_1}$. It is done by reversing these assignments, as in Figure 4.6. First $z := s_2 c$ is performed, and then $c := \frac{1}{s_1} z$. The bottom portion of Figure 4.6 contains a time portrait showing the operation of the incrementation fragment with $W = 4$, $s_1 = 2$, and $s_2 = 1$.

Case 2: $s_2 = 0$**.** In the remaining figures, we omit the wrapping edges required for the clock d. The construction is insensitive to the sign of s_1. The encoding of the

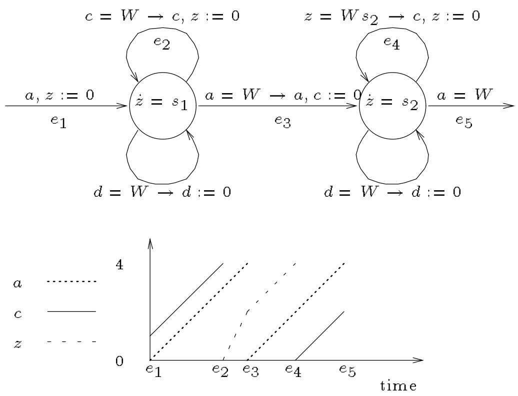

Figure 4.5: Counter decrement: multiplying c by $\frac{s_1}{s_2}$ using the two-slope variable z

two-counter machine is given by counter value r corresponding to clock value 2^{1-r}. We use wrapping constant 4. Decrementing a counter corresponds to doubling the corresponding clock. The doubling procedure is given in Figure 4.7. The idea is to perform $z := s_1 c$ using the assignment lemma, then to put $\dot{z} = 0$ until c reaches W again, and then to put $\dot{z} = s_1$ so that when a reaches W, $z = 2s_1\gamma$, where γ is the original value of c. Then perform $c := \frac{1}{s_1}z$ with the assignment lemma. The lower portion of the figure gives a time portrait illustrating the operation of the fragment for $s_1 = 2$. Halving c requires two auxiliary clocks x and y. First, a value is guessed in x. Then $y := 2x$ is performed using the above doubling procedure. Then $c = y$ is checked by the equality lemma, and if this succeeds, then $c := x$ is performed using the assignment lemma.

Case 3: $s_2 < 0 < s_1$. First suppose $|s_2| < |s_1|$. We use clock value $s_1(\frac{|s_2|}{s_1})^r$ to encode counter value r. The wrapping constant W can be any number larger than s_1. But now we use two synchronization clocks a and b. Clock c is synchro-

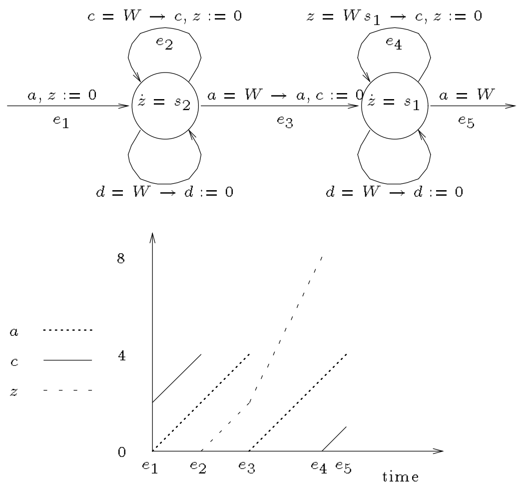

Figure 4.6: Counter increment: multiplying c by $\frac{s_2}{s_1}$ using the two-slope variable z

nized with a, and clock d is synchronized with b. The relationship $c = s_1(\frac{|s_2|}{s_1})^C$ holds when $a = 0$ or $a = W$, and the relationship $d = s_1(\frac{|s_2|}{s_1})^D$ holds when $b = 0$ or $b = W$. To multiply c by $\frac{s_1}{|s_2|}$, we first perform $z := s_1 c$ and reset c to 0. Then we put $\dot{z} = s_2$, and when z reaches 0, we reset a to 0. At this point $c = \frac{s_1}{|s_2|}\gamma$, where γ is the original value of c. See Figure 4.8. The bottom portion of the figure contains a time portrait for $W = 4$, $s_1 = 2$, and $s_2 = -1$. To multiply c by $\frac{s_2}{s_1}$, simply reverse the slopes of z. I.e., perform $z := s_2 c$, reset c to 0, then put $\dot{z} = s_1$ and when z reaches 0, reset a to 0. See Figure 4.9. The bottom portion of the figure contains a time portrait for $W = 4$, $s_1 = 2$, and $s_2 = -1$.

If $|s_2| > |s_1|$, we use clock value $|s_2|(\frac{s_1}{|s_2|})^r$ for counter value r, which simply switches the roles of multiplying by $\frac{s_1}{|s_2|}$ and multiplying by $\frac{|s_2|}{s_1}$. Finally, suppose $s_2 = -s_1$. In this case we use clock value 2^{1-r} for counter value r, and the wrapping constant is 4. Again we use separate synchronization clocks for c and d.

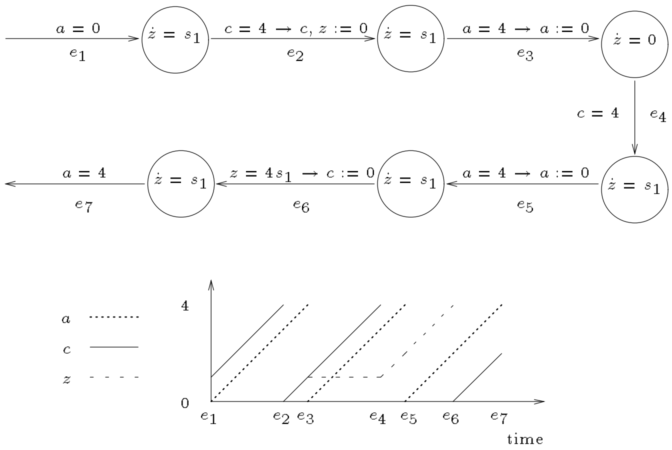

Figure 4.7: Doubling c using variable z taking slopes $0, s_1$

To double c, perform $z := s_1 c$, and then put $\dot{z} = -s_1$, resetting a when z reaches 0. See Figure 4.10, which gives the construction, and also a time portrait for $s_1 = 3$. Halving c is done by nondeterministically guessing the midpoint of the interval of time between $c = 4$ and $a = 4$. The guess is checked by starting z at value 0, giving z at slope s_1 for the first half, and slope $-s_1$ for the second half. If z returns to 0 at the same instant that a reaches 4, the guess was correct. See Figure 4.11, which gives the construction and a time portrait for $s_1 = 5$, $W = 4$. ■

4.2 Non-Rectangular Automata

A slight generalization of the invariant, activity, preguard, postguard, or update function leads to the undecidability of rectangular automata, even under the stringent restrictions of simplicity and initialization.

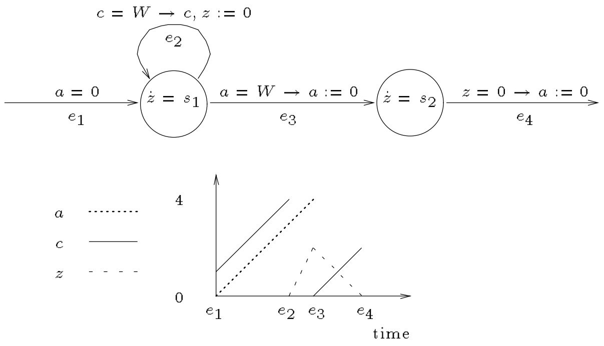

Figure 4.8: Multiplying c by $\frac{s_1}{s_2}$ when $s_2 < 0 < s_1$

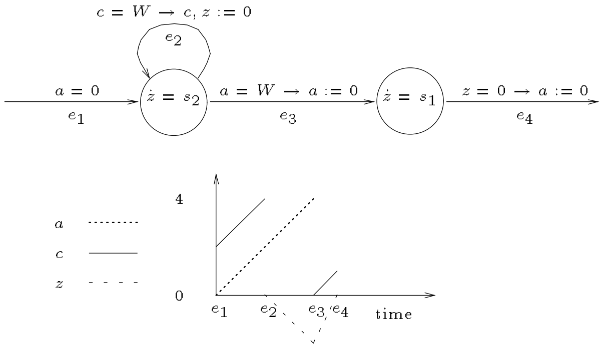

Figure 4.9: Multiplying c by $\frac{s_2}{s_1}$ when $s_2 < 0 < s_1$

Assignment updates

The update function can be generalized to allow the value of one variable to be assigned to another variable. An *assignment update* assigns to each edge e both an update set $update(e) \subset \{1, \ldots, n\}$ and an assignment function $assign(e)$: $\{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$. The edge-step relation $\xrightarrow{e}$ is then redefined as follows: $(v, \mathbf{x}) \xrightarrow{e} (w, \mathbf{y})$ iff $e = (v, w)$, $\mathbf{x} \in preguard(v)$, $\mathbf{y} \in postguard(w)$, and for all $i \notin update(e)$, $y_i = x_{assign(i)}$. Using assignment updates and one skewed clock,

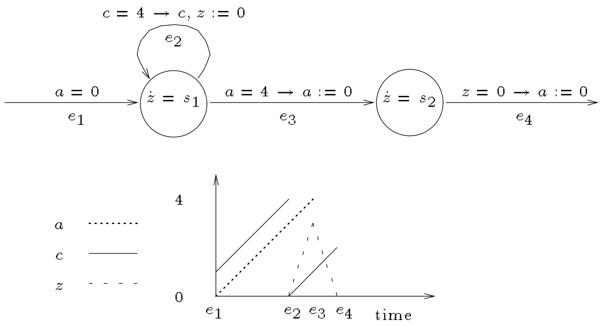

Figure 4.10: Doubling c when $s_2 = -s_1$

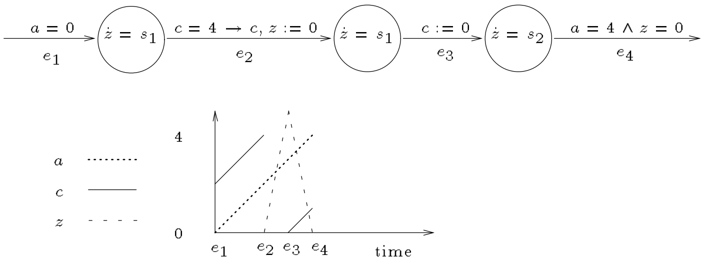

Figure 4.11: Halving c when $s_2 = -s_1$

or assignment updates and one memory cell, the proof of Theorem 4.1.1 can be duplicated. The latter gives a new proof of a result from [Cer92].

Corollary 4.2.1 *For every slope $s \in \mathbb{Q} \setminus \{0, 1\}$, the dense-time reachability problem is undecidable for the class of simple (initialized) rectangular automata with one skewed clock of slope s (resp. one memory cell) and assignment updates.*

Proof. First assume $s > 0$. With assignment updates, it is simple to multiply the value of the clock c by s when a skewed clock z of slope s is available. Simply use the assignment lemma to perform $z := sc$, and then use an assignment update to perform $c := z$. To divide c by s, do the reverse: use an assignment update to perform $z := c$, and then use the assignment lemma to perform $c := \frac{1}{s}z$. We give the construction in Figure 4.12, along with a time portrait for $s = 3$, $W = 4$.

Now assume $s < 0$ and $s \neq -1$. We use one synchronization clock a for clock c, and another synchronization clock b for clock d, as in the proof of Theorem 4.1.1 for $s < 0$. To multiply c by $|s|$, perform $z := sc$ by the assignment lemma, and then perform $a := z; c := 0$ with an assignment update. If γ was the original value of c, then after this sequence $a = s\gamma$ and $c = 0$. After $s\gamma$ time units pass, $a = 0$ and $c = s\gamma$. See Figure 4.13, which includes the construction and a time portrait for $s = -2$, $W = 4$. To divide by $|s|$, perform $z := c; c := 0$ with an assignment update, and then $\frac{\gamma}{|s|}$ time units later when z reaches 0 (and c reaches $\frac{\gamma}{|s|}$), perform $a := 0$. The constructions for $s = -1$ are similar.

When $s = 0$, we have a memory cell, which we refer to as m. We use clock value 2^{1-r} for counter value r. The doubling procedure is given in Figure 4.14. Simply assign $m := c$ when $a = 0$, then wait for c to reach 4 and then assign $c := m$. When a reaches 4, c has twice its original value. Halving is done by guessing and checking, as in Case 2 of Theorem 4.1.1. ■

Triangular preguards, postguards, and invariants

Using triangular preguards only, or triangular postguards only, or triangular invariants only, reachability is undecidable for simple rectangular automata.

Definition 4.2.2 A *rectangular automaton with triangular preguards (resp. postguards; invariants; activity rectangles)* is a triangular automaton A such that all triangles appearing in the definition of A are rectangles, except possibly for the

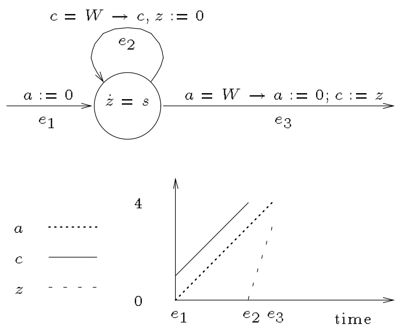

Figure 4.12: Multiplying by $s > 0$ with assignment updates and a skewed clock of slope s

preguard(e) (resp. *postguard*(e); *inv*(v); *act*(v)). ■

Corollary 4.2.3 *For every slope* $s \in \mathbb{Q} \setminus \{0, 1\}$*, the dense-time reachability problem is undecidable for the class of simple (initialized) rectangular automata with one skewed clock of slope* s *and triangular preguards (resp. postguards; invariants).*

Proof. Triangular preguards, postguards, or invariants allow comparisons between the variables of the form $x = y$. This allows an assignment update $y := x$ to be simulated by the unguarded reset $y := 0$ followed later in time by the test $y = x$. It follows that the constructions of Corollary 4.2.1 can be implemented with triangular preguards, postguards, or invariants replacing assignment updates. We give an example multiplication construction (performing $c := sc$) for triangular invariants and $s = 2$ in Figure 4.15. The "$c = z$" inside of the rightmost vertex indicates the triangular invariant. ■

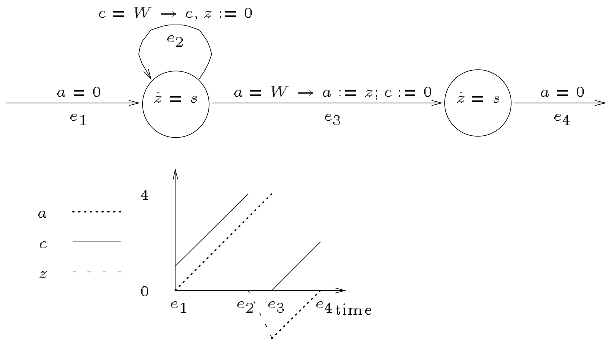

Figure 4.13: Multiplying by $|s|$ with assignment updates and a skewed clock of slope $s < 0$

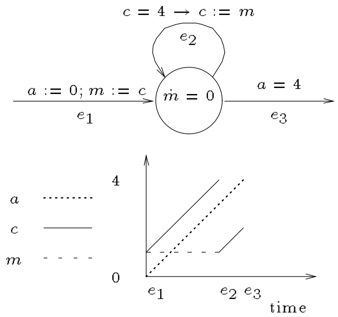

Figure 4.14: Doubling with assignment updates and a memory cell

Triangular activity rectangles

Using three variables and a uniform triangular dynamics, we can simulate the behavior of the two-slope clock from Theorem 4.1.1.

Corollary 4.2.4 *The dense-time reachability problem is undecidable for the class of 3-simple rectangular automata with triangular activity rectangles and uniform dynamics.*

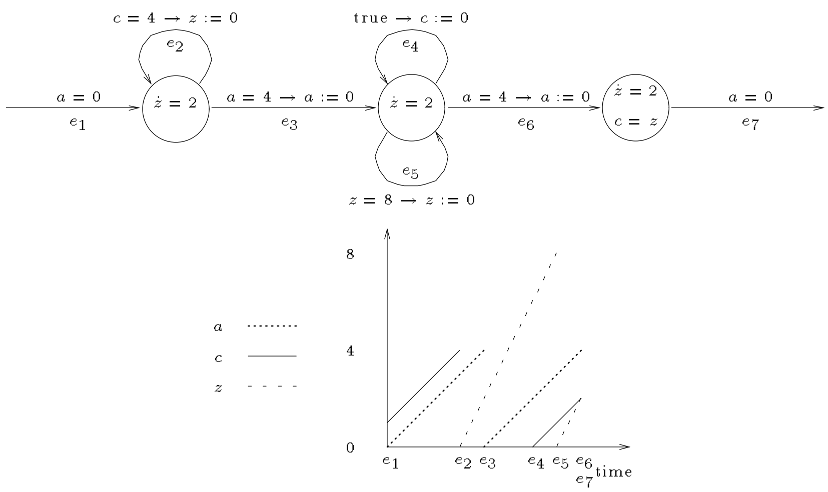

Figure 4.15: Doubling with triangular invariants and a skewed clock

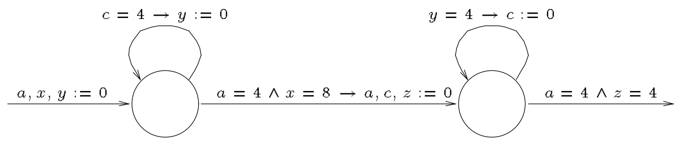

Figure 4.16: Doubling with the triangular activity $1 \leq \dot{x} \leq \dot{y} \leq \dot{z} \leq 2$

Proof. For this proof we use three variables x, y, z with global triangular activity $1 \leq \dot{x} \leq \dot{y} \leq \dot{z} \leq 2$. The doubling construction is given in Figure 4.16. The variable y will actually take only slopes 1 and 2; the former is accomplished by resetting z to 0 when a wraps to 0, and then later testing $a = 4 \wedge z = 4$; similarly the latter is accomplished by resetting x to 0 when a wraps to 0, and then later testing $a = 4 \wedge x = 8$. In this way, the two-slope variable constructions of Theorem 4.1.1 can be duplicated. ■

Chapter 5

Linear Temporal Logic

A drawback of transition system semantics for hybrid automata is the existence of infinite computations during which only a finite amount of time passes. Such computations do not correspond to any real-world behavior, and therefore must be ignored when a system is verified. We turn to the model checking problem for linear temporal logic, under the fairness condition of time divergence. On the class of multirate automata, this problem is PSPACE-complete. The translation from a rectangular automaton A with constant dynamics into the multirate automaton M_A is language-preserving, but in general does not preserve the divergent language (which is the divergent subset of the ω-language). By obtaining a condition on A under which the translation is divergent language-preserving, we a obtain class of RHA for which the LTL model checking problem under divergence is PSPACE-complete. This condition is called *bounded nondeterminism*. It essentially requires that for any bounded zone $Z \subset Q_A$, and any $\pi \in E \cup \mathbb{R}_{\geq 0}$, that

$Post^{\pi}(Z)$ be bounded.

For this chapter, we fix a rectangular automaton A with uniform dynamics, a nonempty set Π of atomic propositions of interest, and a satisfaction relation $\models \subset Q_A \times \Pi$ depending only on the vertex of a state. That is, for every $\pi \in \Pi$, and all states $(v, \mathbf{x}), (w, \mathbf{y}) \in Q_A$, if $v = w$ then $(v, \mathbf{x}) \models \pi$ iff $(w, \mathbf{y}) \models \pi$. Extend $\models$ to Q_{M_A} by defining $(v, \vec{\lambda}, \vec{\nu}, tp, \mathbf{y}) \models \pi$ iff $(v, \mathbf{x}) \models \pi$ for some $\mathbf{x}$.

5.1 The Divergent Language

Definition 5.1.1 Let B be either A or M_A. We define the *timed* transition system

$$S_B^{\mathbb{R}} =_{\text{def}} (Q_B, \Sigma_A \cup \mathbb{R}_{\geq 0}, \rightarrow, Q_B^0, \Pi, \models)$$

(whose dependence on Π and $\models$ is suppressed) and the *unlabeled timed* transition system

$$S_B^{E} =_{\text{def}} (Q_B, E_B \cup \mathbb{R}_{\geq 0}, \rightarrow, Q_B^0, \Pi, \models). \blacksquare$$

In the timed and unlabeled timed transition systems, time steps of each duration $t \in \mathbb{R}_{\geq 0}$ are distinguished.

Definition 5.1.2 A *timed word* is a pair $(\bar{P}, \bar{\pi})$ of infinite sequences where $P_i \subset \Pi$ and $\pi_i \in \Sigma_A \cup \mathbb{R}_{\geq 0}$ for all $i \in \mathbb{N}$. A *timed edge word* is a pair $(\bar{P}, \bar{\varpi})$ of infinite sequences where $P_i \subset \Pi$ and $\varpi_i \in E_A \cup \mathbb{R}_{\geq 0}$ for all $i \in \mathbb{N}$. The timed word (resp. timed edge word) $(\bar{P}, \bar{\pi})$ is *divergent* if $\sum\{\pi_i \mid \pi_i \in \mathbb{R}_{\geq 0}\} = \infty$. For a timed (edge) word $\rho = (\bar{P}, \bar{\pi})$, and a $k \in \mathbb{N}$, the *k-prefix* of ρ, denoted by $\rho\lceil_k$, is the pair of strings $(P_0 P_1 \cdots P_k, \pi_0 \pi_1 \cdots \pi_{k-1})$. $\blacksquare$

Definition 5.1.3 Let B be either A or M_A. Let $Z \subset Q_B$ be a zone. The *divergent language* of Z in B is the set

$$L^{div}(Z, B) =_{\text{def}} \{(\bar{P}, \bar{\pi}) \in L^{\omega}(Z, S_B^{\mathbb{R}}) \mid (\bar{P}, \bar{\pi}) \text{ is divergent}\}.$$

The *divergent language* of B, denoted $L^{div}(B)$, is $L^{div}(Q_B^0, B)$. The *divergent edge language* of Z in B is the set

$$EL^{div}(Z, B) =_{\text{def}} \{(\bar{P}, \bar{\varpi}) \in L^{\omega}(Z, S_A^E) \mid (\bar{P}, \bar{\varpi}) \text{ is divergent}\}.$$

The *divergent edge language* of B, denoted $EL^{div}(B)$, is $EL^{div}(Q_B^0, B)$. ■

Thus the divergent (edge) language of Z in B is the set of divergent timed (edge) words in the ω-language of $S_B^{\mathbb{R}}$ (S_B^E).

Definition 5.1.4 The *divergent language emptiness problem* for a class $\mathcal{C}$ of rectangular automata is stated in the following way:

> *Given a rectangular automaton $A \in \mathcal{C}$, determine whether $L^{div}(A)$ is empty.* ■

Theorem 5.1.5 [AD94] *The divergent language emptiness problem for the class of multirate automata with attractors and silent moves is PSPACE-complete.* ■

While Lemmas 3.3.7, 3.3.6, and 3.3.10 imply that M_A and A generate the same finite timed words (i.e., $L(S_{M_A}^{\mathbb{R}}) = L(S_A^{\mathbb{R}})$), the multirate automaton M_A may generate divergent timed words that A does not, so that $L^{div}(M_A) \neq L^{div}(A)$. For example, consider the timed automaton C in Figure 5.1, with initial zone $Z = \{v\} \times (-\infty, 0]$, The divergent timed word $((\Pi(v))^{\omega}, (1\sigma)^{\omega})$ is not an element

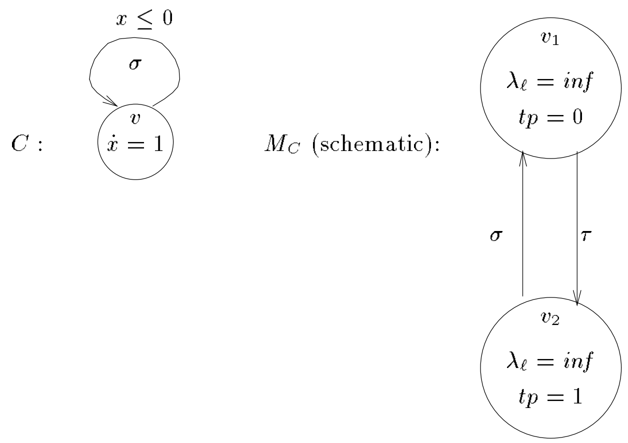

Figure 5.1: $L^{div}(C) \subsetneq L^{div}(M_C)$

of $L^{div}(C)$, because for any initial value y of the continuous variable x, at most $|y|$ time units can pass before x becomes positive and the σ-edge is disabled. However, since Z is unbounded, in M_C the lower finite/infinite bit $\lambda_{\ell(1)}$ remains at *inf* on time steps and σ steps. Therefore $((\Pi(v))^\omega, (1\sigma)^\omega)$ is an element of $L^{div}(M_C)$. Consider the schematic picture of M_C in Figure 5.1, in which continuous variables, guarded commands, the weak/strict bits, and the finite/infinite bit for the upper bound multirate variable are suppressed. The multirate automaton M_A has the divergent computation $(v_1 \xrightarrow{\tau} v_2 \xRightarrow{1} v_2 \xrightarrow{\sigma})^\omega$, where we have suppressed the continuous state. This is due to the unboundedness of the initial zone. Similar behavior is exhibited in automata with unbounded postguards. The definition of bounded nondeterminism, found below, precludes both.

Definition 5.1.6 An n-dimensional rectangular automaton A has *bounded nondeterminism* if (1) for every vertex v, $init(v)$ and $act(v)$ are bounded, and (2) for every edge e, and every $i \in \{1, \ldots, n\}$, if $i \in \mathit{update}(e)$, then $\mathit{postguard}(e)_i$ is bounded. ■

Note that bounded nondeterminism does not imply finite branching. It merely ensures that the successor of a bounded region is bounded.

We prove that if A has bounded nondeterminism, then $L^{div}(A) = L^{div}(M_A)$. The main theorem states that if A has bounded nondeterminism then the divergent language $L^{div}(A)$ is limit-closed. I.e., if every finite prefix of the timed word $(\bar{P}, \bar{\pi})$ is an element of $L(Z, S_A^{\mathbb{R}})$, then $(\bar{P}, \bar{\pi}) \in L^{div}(Z, A)$. From this it follows that $L^{div}(A) = L^{div}(M_A)$. We first give the result for A with the stronger requirement of compact nondeterminism, because the proof of the main theorem for this restricted case is a simple consequence of the fact that a decreasing sequence of nonempty compact set has nonempty intersection. Thereafter we proceed to the general case.

5.2 Compact Nondeterminism

The first proposition gives a basic property of compact zones that is inherited from Euclidean space.

Proposition 5.2.1 *Let A be a rectangular automaton, and let $(Z_i)_{i \in \mathbb{N}}$ be a decreasing sequence of nonempty compact zones of A. Then the intersection $\bigcap_{i \in \mathbb{N}} Z_i$ is nonempty.*

Proof. This follows from the corresponding statement for compact subsets of $\mathbb{R}^n$, and the fact that V_A is finite. ■

Definition 5.2.2 A rectangular automaton A has *compact nondeterminism* if it is closed and has bounded nondeterminism. ■

We show that rectangular automata with compact nondeterminism define limit-closed divergent languages. The following two technical lemmas are used to establish the compactness of all zones that are used in the proof of the main theorem.

Lemma 5.2.3 *Let A be a rectangular automaton with compact nondeterminism. For every compact multirectangular zone $Z \subset Q_A$, and every $\varpi \in \mathbb{R}_{\geq 0} \cup E_A$, the zone $Post_A^{\varpi}(Z)$ is compact and multirectangular.* ■

Lemma 5.2.4 *Let A be a rectangular automaton with compact nondeterminism. For every pair of compact multirectangular zones $Z, Z' \subset Q_A$, and every $\varpi \in \mathbb{R}_{\geq 0} \cup E_A$, the zone $Pre_A^{\varpi}(Z') \cap Z$ is compact and multirectangular.* ■

Note the asymmetry of the two lemmas. The intersection of $Pre_A^{\varpi}(Z')$ with the compact zone Z is required for compactness, because preguards of automata with compact nondeterminism are only required to be closed, not compact. The next lemma gives the meat of the limit-closure argument, showing that if all prefixes of a timed edge word may be generated from a given zone Z, then in fact there is an element of Z from which each prefix may be generated.

Lemma 5.2.5 *Let A be a rectangular automaton with compact nondeterminism, and let $Z \subset Q_A$ be a compact multirectangular zone. Suppose τ is a timed edge word such that for every $k \in \mathbb{N}$, $\tau\lceil_k \in L(Z, S_A^E)$. Then there exists a state $q \in Z$ such that for every $k \in \mathbb{N}$, $\tau\lceil_k \in L(\{q\}, S_A^E)$.*

Proof. Put $\tau = (\bar{P}, \bar{\varpi})$. For each $k \in \mathbb{N}$, define

$$J_k =_{\text{def}} \{q \in Z \mid Post_A^{\varpi_0 \varpi_1 \cdots \varpi_k}(\{q\}) \neq \emptyset\}.$$

Each J_k is nonempty because each $Post_A^{\varpi_0 \varpi_1 \cdots \varpi_k}(Z)$ is nonempty. Also, $J_k \supset J_{k+1}$ for every k. We claim each J_k is compact. If so, then the sequence $(J_k)_{k \in \mathbb{N}}$ is a decreasing sequence of nonempty compact sets. Hence the intersection $\bigcap_{k=0}^{\infty} J_k$ is nonempty. An element of the intersection is the requirement of the lemma.

We now establish the claim that each J_k is compact. By Lemma 5.2.3, for each $k \in \mathbb{N}$, the zone $Post_A^{\varpi_0 \varpi_1 \cdots \varpi_k}(Z)$ is compact and multirectangular. Hence by Lemma 5.2.4, $J_k = Z \cap Pre_A^{\varpi_k \cdots \varpi_1 \varpi_0}(Post_A^{\varpi_0 \varpi_1 \cdots \varpi_k}(Z))$ is compact as well. ■

The following main theorem establishes the limit-closure of $L^{div}(Z, A)$ for all rectangular automata A with compact nondeterminism, and all compact multirectangular zones Z.

Theorem 5.2.6 *Let A be a rectangular automaton with compact nondeterminism, and let $Z \subset Q_A$ be a compact multirectangular zone. Suppose ρ is a timed word such that for every $k \in \mathbb{N}$, $\rho\lceil_k \in L(Z, S_A^{\mathbb{R}})$. Then there exists a state $q \in Z$ such that $\rho \in L^{div}(\{q\}, A)$.*

Proof. Put $\rho = (\bar{P}, \bar{\pi})$. By König's lemma, there exists a timed edge word $\tau = (\bar{P}, \bar{\varpi})$ such that $event(\varpi_i) = \pi_i$ for all $\varpi_i \in E$, and $\tau\lceil_k \in L(Z, S_A^E)$ for every $k \in \mathbb{N}$.

Let $Z_0 = Z$. By Lemma 5.2.5, there exists a state $q_0 \in Z_0$ such that for every $k \geq 0$, $Post_A^{\varpi_0 \varpi_1 \cdots \varpi_k}(\{q_0\}) \neq \emptyset$. Let $Z_1 = Post_A^{\varpi_0}(\{q_0\})$. The zone Z_1 is compact and multirectangular, and for each $k \geq 1$, $Post_A^{\varpi_1 \varpi_2 \cdots \varpi_k}(Z_1)$ is nonempty.

So by Lemma 5.2.5, there exists a state $q_1 \in Z_1$ such that for each $k \geq 1$, $Post_A^{\varpi_1\varpi_2\cdots\varpi_k}(\{q_1\})$ is nonempty. Proceed inductively in this manner, with $Z_{j+1} = Post_A^{\pi_j}(\{q_j\})$ compact and multirectangular, and $q_{j+1} \in Z_{j+1}$ given by Lemma 5.2.5, such that for each $k > j+1$, $Post_A^{\varpi_{j+1}\varpi_{j+2}\cdots\varpi_k}(\{q_{j+1}\}) \neq \emptyset$. Then

$$q_0 \overset{\varpi_0}{\rightarrow} q_1 \overset{\varpi_1}{\rightarrow} q_2 \overset{\varpi_2}{\rightarrow} \cdots$$

Therefore $(\bar{P}, \bar{\varpi}) \in EL^{div}(Z, A)$, and consequently $(\bar{P}, \bar{\pi}) \in L^{div}(Z, A)$. ■

Theorem 5.2.7 *For every rectangular automaton A with uniform dynamics and compact nondeterminism, $L^{div}(A) = L^{div}(M_A)$.*

Proof. The inclusion $L^{div}(A) \subset L^{div}(M_A)$ is immediate from Lemmas 3.3.7, 3.3.6, and 3.3.10. For the reverse, put $Z = Q_A^0$, and suppose

$$q_0 \overset{\varpi_0}{\rightarrow} q_1 \overset{\varpi_1}{\rightarrow} q_2 \overset{\varpi_2}{\rightarrow} \cdots$$

in M_A, where $q_0 \in lowhigh Z$. Then there exist states $q'_k \in U_{M_A}$, $k = 0, 1, 2 \ldots$, in the upper half space of M_A such that

$$q'_0 \overset{\varpi_0}{\rightarrow} q'_1 \overset{\varpi_1}{\rightarrow} q'_2 \overset{\varpi_2}{\rightarrow} \cdots,$$

where $q'_0 \in lowhigh Z$, and $\Pi(q'_i) = \Pi(q_i)$ for each $i \in \mathbb{N}$. Define $\bar{\varpi}' \in (E_A \cup \mathbb{R}_{\geq 0})^\omega$ by $\varpi'_k = \varpi_k$ if $\varpi_k \in \mathbb{R}_{\geq 0}$, and $\varpi'_k = e$ if ϖ_k is an edge of M_A derived from the edge e of A by an edge family $\Psi(e, \vec{\lambda}, \vec{\nu}, tp)$. Then by Lemmas 3.3.7, 3.3.6, and 3.3.10, for each k,

$$Post_A^{\varpi'_0\varpi'_1\cdots\varpi'_k}(Z) = \beta_A(Post_{M_A}^{\varpi_0\varpi_1\cdots\varpi_k}(lowhigh Z)) \supset \beta_A(\{q'_{k+1}\}) \neq \emptyset.$$

By Theorem 5.2.6, there exists a state $q \in Z$ such that $(\bar{P}, \bar{\varpi}') \in EL^{div}(\{q\}, A)$. ■

Corollary 5.2.8 *The divergent language emptiness problem for the class of initialized rectangular automata with compact nondeterminism is PSPACE-complete.*

Proof. By Theorems 5.2.7 and 5.1.5, and the proof of Corollary 3.4.2, using the fact that M_A need not be explicitly constructed in order to check the emptiness of its divergent language. ■

5.3 Bounded Nondeterminism

Let A be a rectangular automaton. Recall that $L^\omega(S_A^{\mathbb{R}})$ is the set of infinite timed words generated by A that are not necessarily divergent. Notice that if A has compact nondeterminism, then $L^\omega(S_A^{\mathbb{R}})$ is limit-closed. This is no longer the case if A only has bounded nondeterminism. Consider, for example, the timed automaton D in Figure 5.2. Every finite prefix of the infinite timed word $\rho = ((\Pi(v))^\omega, \sigma' \frac{1}{2} \sigma \frac{1}{4} \sigma \frac{1}{8} \sigma \cdots)$ is generated by a finite computation of D, and yet $\rho \notin L^\omega(S_A^{\mathbb{R}})$.

However, we show that $L^{div}(A)$, the set of *divergent* timed words generated by A, is limit-closed for all rectangular automata A with bounded nondeterminism. That is, whenever every finite prefix of a timed word ρ in which the time steps sum to infinity can be generated by A, then the infinite sequence ρ can be generated by A. The proof of limit-closure is greatly complicated by the lack of an analogue to Proposition 5.2.1 for bounded regions. The limit-closure of $L^{div}(A)$ is now proven by a detailed case analysis of the activity function.

The following technical lemma is used to establish the boundedness of all zones

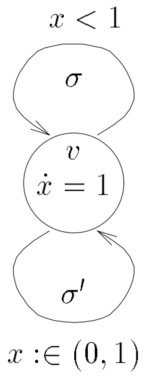

Figure 5.2: The need for time divergence

appearing in the proof of limit-closure.

Lemma 5.3.1 *Let A be a rectangular automaton with bounded nondeterminism. For every bounded multirectangular zone $Z \subset Q_A$, and every $\varpi \in \mathbb{R}_{\geq 0} \cup E_A$, the zone $Post_A^{\varpi}(Z)$ of ϖ-successors of Z is bounded and multirectangular.* ■

The next lemma gives the heart of the proof of limit-closure. It says that for 1-dimensional A with uniform bounded activity interval, $L^{div}(A)$ is limit-closed with respect to timed edge words without updates of the continuous state.

Lemma 5.3.2 *Let $\mathcal{I}$ be a finite set of nonempty intervals, and let R, Act be bounded intervals. Let $(t_k)_{k \in \mathbb{N}}$ be a sequence of positive real numbers with $\sum_{k=0}^{\infty} t_k = \infty$, and let $(I_k)_{k \in \mathbb{N}}$ be a sequence of intervals such that $I_0 = R$, and for every $k \geq 1$, I_k is the intersection of one or more members of $\mathcal{I}$. Suppose for each $k \in \mathbb{N}$, there is a finite sequence $x_0, x_1, \ldots, x_k$ of real numbers such that for all $0 \leq j \leq k$, $x_j \in I_j$, and for each $0 \leq j < k$, $\frac{x_{j+1} - x_j}{t_j} \in Act$. Then there is an infinite sequence $(x_k)_{k \in \mathbb{N}}$ such that for each $k \in \mathbb{N}$, $x_k \in I_k$ and $\frac{x_{k+1} - x_k}{t_k} \in Act$.*

Proof. Consider a continuous variable x, initialized nondeterministically to some value in R, and with $\dot{x} \in Act$. Call a finite sequence $x_0, x_1, \ldots, x_k$ *k-admissible* if for all $0 \leq j \leq k$, $x_j \in I_j$, and for each $0 \leq j < k$, $\frac{x_{j+1}-x_j}{t_j} \in Act$. Call an infinite sequence $(x_k)_{k\in\mathbb{N}}$ *admissible* if for each $k \in \mathbb{N}$, $x_k \in I_k$ and $\frac{x_{k+1}-x_k}{t_k} \in Act$.

Case 1: $0 \notin \overline{Act}$. Suppose $Act \subset (\epsilon, \infty)$, where $\epsilon > 0$. The case of $Act \subset (-\infty, \epsilon)$ is handled symmetrically. Let h be larger than all of the finite endpoints of the intervals in $\mathcal{I}$. The point here is that the speed of x is bounded below by ϵ, and so that once $\frac{h-\inf R}{\epsilon}$ time has passed, no matter what the initial value of x, the value of x will be greater than all of the finite bounds defining intervals from $\mathcal{I}$. Let m be large enough so that $\sum_{k=0}^{m-1} t_i > \frac{h-\inf R}{\epsilon}$. We claim that for every $x_0 \in R$ for which there exists an m-admissible finite sequence $x_0, x_1, \ldots, x_m$, there is in fact an admissible infinite sequence $(x_k)_{k\in\mathbb{N}}$ extending $x_0, x_1, \ldots, x_m$. By assumption, for every $k \in \mathbb{N}$, a k-admissible sequence exists. For any such sequence $y_0, y_1, \ldots, y_k$, it must be that $y_j > h$ for each $j > m$. It follows that since $\mathcal{I}$ is finite, and every I_i is an intersection of elements of $\mathcal{I}$, that for every $k \geq m$, $I_k \supset (h, \infty)$. Since Act contains some $\epsilon > 0$, any m-admissible finite sequence $x_0, x_1, \ldots, x_m$ can be extended to the admissible infinite sequence

$$x_0, x_1, \ldots, x_m, x_m + \epsilon t_m, x_m + \epsilon(t_m + t_{m+1}), \ldots$$

Case 2: $0 = \inf Act$. The case of $0 = \sup Act$ is handled symmetrically. Among the I_i are only finitely many distinct intervals, because there are only finitely many intersections of the finitely many elements of $\mathcal{I}$. Let $\mathcal{W}$ be the set $\{I \subset \mathbb{R} \mid I = I_i \text{ for infinitely many } i\}$. Then $\bigcap \mathcal{W} \neq \emptyset$, because $Act \cap (-\infty, 0) = \emptyset$, so x can never descend from an I_i to an I_j all of whose elements are less than those of I_i. Let m_1 be large enough so that for every $k \in \mathbb{N}$, $I_{m_1+k} \in \mathcal{W}$, and moreover that for every $W \in \mathcal{W}$, there is a $k < m_1$ with $I_k = W$. I.e., m_1 is large enough so that all elements of $\mathcal{W}$ have been met in the past, and only elements of $\mathcal{W}$ will be met in the future. Let $m_2 > m_1$ be large enough so that all elements of $\mathcal{W}$ are represented among $I_{m_1+1}, \ldots I_{m_2-1}$. Let $x_0, x_1, \ldots, x_{m_2}$ be an m_2-admissible sequence. Then $x_{m_1} \in \bigcap \mathcal{W}$, because x cannot decrease, each element of $\mathcal{W}$ contains at least one of the x_i with $i < m_1$, and each element of $\mathcal{W}$ contains at least one of the x_i with $m_1 < i$. If $0 \in Act$, then the infinite sequence $x_0, x_1, \ldots, x_{m_1}^{\omega}$ is admissible. If $0 \notin Act$, then $x_{m_1} < \sup \bigcap \mathcal{W}$. Let $\delta = (\sup \bigcap \mathcal{W}) - x_{m_1}$. For each $i > m_1$, let ϵ_i be so that $0 < \epsilon_i < \frac{\delta}{t_i 2^i}$. Then the infinite sequence

$$x_0, x_1, \ldots, x_{m_1}, x_{m_1} + \epsilon_{m_1+1} t_{m_1+1}, x_{m_1} + \epsilon_{m_1+1} t_{m_1+1} + \epsilon_{m_1+2} t_{m_1+2}, \ldots$$

is admissible.

Case 3: 0 is in the interior of Act **and** $\bigcap \mathcal{W} \neq \emptyset$**.** Since 0 is in the interior of Act, every trajectory can be slowed down to give another trajectory. Let m_1 be as in the previous paragraph. Since $0 \in Act$, whenever an $(m_1 + \ell)$-admissible finite sequence terminates in $\bigcap \mathcal{W}$, it can be extended to an admissible infinite sequence by repeating the last state ad infinitum. Such an $(m_1 + \ell)$-admissible sequence terminating in $\bigcap \mathcal{W}$ exists, because there exist $W_1, W_2 \in \mathcal{W}$ with $\inf W_1 = \inf \bigcap \mathcal{W}$ (with same strictness) and $\sup W_2 = \sup \bigcap \mathcal{W}$ (with same strictness). Any $(m + \ell)$-admissible finite sequence with ℓ large enough so that both W_1, W_2 each appear twice in $I_{m+1}, \ldots, I_{m+\ell-1}$ must have $m < i < k$ with $x_i \in W_1$ and $x_k \in W_2$. By slowing down the trajectory, $\bigcap \mathcal{W}$ can be reached: if $x_i \geq \sup \bigcap \mathcal{W}$ and $x_k \leq \inf \bigcap \mathcal{W}$, then for some j with $i \leq j < k$, $x_j \geq \sup \bigcap \mathcal{W}$ and $x_{j+1} < \sup \bigcap \mathcal{W}$. By letting y be any number such that $x_{j+1} < y$ and $y \in \bigcap \mathcal{W}$, the infinite sequence

$$x_0, x_1, \ldots, x_m, \ldots, x_j, y^\omega$$

is admissible.

Case 4: 0 is in the interior of Act **and** $\bigcap \mathcal{W} = \emptyset$**.** Let $W_1, W_2 \in \mathcal{W}$ be such that every element of W_1 is greater than every element of W_2. Let m_1 be as in the previous two paragraphs. Let $m_1 < p_1 < q_1 < p_2$ be so that $I_{p_1} = I_{p_2} = W_1$ and $I_{q_1} = W_2$. Let $x_0, x_1, \ldots, x_{p_2}$ be p_2-admissible. We will first show that for every $k \in \mathbb{N}$, there is a k-admissible finite sequence starting from x_0. This is obvious for $k \leq p_2$, so suppose $k > p_2$. Let $y_0, y_1, \ldots, y_k$ be k-admissible. We now have three cases.

Subcase 4a: $x_{p_1} = y_{p_1}$. In this case $x_0, x_1, \ldots, x_{p_1}, y_{p_1+1}, y_{p_1+2}, \ldots, y_k$ is a k-admissible sequence.

Subcase 4b: $x_{p_1} < y_{p_1}$. If $x_{q_1} < y_{q_1}$, then by slowing down, the x_i sequence can meet up with the y_i sequence somewhere along the descent from W_1 to W_2. If $x_{q_1} > y_{q_1}$, then for some $p_1 < j \leq q_1$, $x_j < y_j$ and $x_{j+1} \geq y_{j+1}$. Since $\frac{y_{j+1}-y_j}{t_j} \in Act$ and $\frac{x_{j+1}-x_j}{t_j} \in Act$, and

$$y_{j+1} - y_j < y_{j+1} - x_j < x_{j+1} - x_j,$$

it must be that $\frac{y_{j+1}-x_j}{t_j} \in Act$. Hence the finite sequence

$$x_0, x_1, \ldots, x_j, y_{j+1}, y_{j+2}, \ldots, y_k$$

is k-admissible.

Subcase 4c: $x_{p_1} > y_{p_1}$. If $x_{q_1} < y_{q_1}$, then by slowing down, the x_i sequence can meet up with the y_i sequence somewhere along the descent from W_1 to W_2. So suppose $x_{q_1} > y_{q_1}$. Now if $x_{p_2} > y_{p_2}$, then by slowing down, the x_i sequence can meet up with the y_i sequence somewhere along the ascent from W_2 to W_1. If $x_{p_2} < y_{p_2}$, the the y_i sequence must cross the x_i sequence as above, and the same $x_0, x_1, \ldots, x_j, y_{j+1}, y_{j+2}, \ldots, y_k$ construction provides a k-admissible finite sequence beginning with x_0. The subcase of $x_{p_1} > y_{p_1}$ is complete.

It remains to construct an admissible infinite sequence. Let $x_0 \in R$ be such that for every $k \in \mathbb{N}$, there is a k-admissible finite sequence starting with x_0. Let R_1 be the set of t_0-successors of x_0, i.e., $R_1 = \{y \in \mathbb{R} \mid \frac{y-x_0}{t_0} \in Act\}$. Then R_1 is bounded by Lemma 5.3.1. So applying what we have already proven to R_1, the time sequence $\lambda k.t_{k+1}$, and the interval sequence $\lambda_k.I_{k+1}$, there is an $x_1 \in R_1$ such that for every k, there is a k-admissible (with respect to $\lambda k.t_{k+1}$ and $\lambda_k.I_{k+1}$) sequence beginning from x_1. Continuing inductively, we form an admissible sequence beginning at x_0. ■

Now the proof of the main theorem consists of reducing to one dimension, eliminating updates, and applying Lemma 5.3.2.

Theorem 5.3.3 *Let A be a rectangular automaton uniform dynamics and bounded nondeterminism, and let $Z \subset Q_A$ be a bounded rectangular zone. Suppose ρ is a* *divergent timed word such that for every* $k \in \mathbb{N}$, $\rho\lceil_k \in L(Z, S_A^{\mathbb{R}})$. *Then there exists a state* $q \in Z$ *such that* $\rho \in L^{div}(\{q\}, A)$.

Proof. It suffices to prove the proposition for 1-dimensional A, for each component of a computation of a multi-dimensional A is independent of the other components—that is, an n-dimensional automaton A has the computation κ in the unlabeled timed transition system S_A^E iff each of the n 1-dimensional automata defined by restricting A to one continuous component has the corresponding component sequence of κ as a computation of its unlabeled timed transition system. Henceforth we assume that A is 1-dimensional.

Put $\rho = (\bar{P}, \bar{\pi})$. By König's lemma, there exists a timed edge word $\tau = (\bar{P}, \bar{\varpi})$ such that $event(\varpi_i) = \pi_i$ for all $\varpi_i \in E$, and $\tau\lceil_k \in L(Z, S_A^E)$ for every $k \in \mathbb{N}$.

If there exist infinitely many k with $1 \in update(\varpi_k)$, then by stringing together the pieces in between the updates, $\tau \in EL^{div}(\{q\}, A)$ for every state $q \in Z$ such that $Post^{\varpi_0\varpi_1\cdots\varpi_k}(q) \neq \emptyset$ for some k with $1 \in update(\varpi_k)$. So assume that $1 \in update(\varpi_k)$ for only finitely many k. It now suffices to assume that $1 \in update(\varpi_k)$ for no k. Because if $k_{\max} = \max\{k \in \mathbb{N} \mid 1 \in update(\varpi_k)\}$, then by proving the theorem with $Post^{\varpi_0\varpi_1\cdots\varpi_{k_{\max}}}(Z)$ in place of Z, and $\lambda p.(P_{\lambda p.1+k_{\max}+p}, \pi_{1+k_{\max}+p})$ in place of ρ, the result for Z and ρ follows by picking any $q \in Z$ such that $Post^{\varpi_0\varpi_1\cdots\varpi_{k_{\max}}}(\{q\})$ is nonempty. The proposition now follows from an application of Lemma 5.3.2, where the set $\mathcal{I}$ of intervals is the set of all nonempty values of inv_A, $preguard_A$, and $postguard_A$. ■

As Theorem 5.2.7 follows from Theorem 5.2.6, so the next theorem follows from Theorem 5.3.3.

Theorem 5.3.4 *For every rectangular automaton A with uniform dynamics and bounded nondeterminism, $L^{div}(A) = L^{div}(M_A)$.* ■

Theorem 5.3.5 *The divergent language emptiness problem for the class of initialized rectangular automata with bounded nondeterminism is PSPACE-complete.* ■

5.4 LTL Model Checking

We can now state and solve the model checking problem for linear temporal logic under the fairness condition of time divergence on the class of initialized rectangular automata with bounded nondeterminism.

Definition 5.4.1 A computation $\kappa = q_0\pi_0q_1\pi_1\ldots$ of the timed transition system $S_A^{\mathbb{R}}$ is *divergent* if $\sum\{\pi_i \mid \pi_i \in \mathbb{R}_{\geq 0}\} = \infty$. ■

Definition 5.4.2 A computation $\kappa = q_0\pi_0q_1\pi_1\ldots$ of the time-abstract transition system S_A^{abs} is *divergent* if there exists a divergent computation $\kappa = q'_0\pi'_0q'_1\pi'_1\ldots$ of the timed transition system $S_A^{\mathbb{R}}$ such that for every $i \in \mathbb{N}$, (1) $q'_i = q_i$, (2) if $\pi_i \in \Sigma_A$ then $\pi'_i = \pi_i$, and (3) if $\pi_i = time$ then $\pi'_i \in \mathbb{R}_{\geq 0}$. ■

Definition 5.4.3 The *model checking problem for linear temporal logic under time divergence* for a class $\mathcal{C}$ of rectangular automata is stated in the following way:

> *Given a rectangular automaton $A \in \mathcal{C}$ and a formula ϕ of linear temporal logic over the time-abstract transition system S_A^{abs}, let $\mathcal{F}$ be the fairness condition consisting of all divergent computations of S_A^{abs}. De-*

termine whether all initial states of A satisfy ϕ under the fairness condition $\mathcal{F}$. ■

See Definition 2.1.29 for the definition of satisfaction under a fairness condition. An equivalent formulation is the following: if $\phi = \forall\xi$, determine whether every divergent computation κ of S_A^{abs} from an initial state satisfies ξ.

Theorem 5.4.4 [AD94] *The model checking problem for linear temporal logic under time divergence on the class of multirate automata with attractors is PSPACE-complete.*

Proof. PSPACE-hardness follows from the PSPACE-hardness of the reachability problem. Let M be a multirate automaton with attractors. Let ϕ be a formula of linear temporal logic. There exists a Büchi automaton $B_{\neg\phi}$, of size singly exponential in the length of ϕ, such that the ω-language of B is the set of computations that do not satisfy ϕ [VW86]. There exists another Büchi automaton B_M, of size singly exponential in the size of M, such that the ω-language of B is the set of divergent computations of M [AD94]. It follows that the language of the product automaton $B_M \times B_{\neg\phi}$ is nonempty iff M does not satisfy ϕ. The size of the product singly exponential in the size of (M, ϕ). Since the nonemptiness problem for Büchi automata may be solved on-the-fly in nondeterministic logarithmic space (see Theorem 2.1.32), the emptiness of $B_M \times B_{\neg\phi}$ may be determined in space polynomial in the size of (M, ϕ). ■

Theorem 5.4.5 *The model checking problem for linear temporal logic under time divergence on the class of initialized rectangular automata with bounded nondeter-*

minism is PSPACE-complete.

Proof. By the proof of Corollary 3.4.2, the model checking problem for initialized rectangular automata may be reduced to the model checking problem for rectangular automata with uniform dynamics. The latter result follows from Theorems 5.3.4 and 5.4.4, using (1) the fact that the size of the bisimilarity quotient of M_A is singly exponential in the size of A, (even though M_A itself is singly exponential in the size of A), (2) the fact that M_A need not be explicitly constructed in order to check the emptiness of $B_{M_A} \times B_{\neg\phi}$ in the proof of Theorem 5.4.4, and (3) there is at most one silent move during any time step in M_A. ∎

Chapter 6

Simulation Equivalence

For infinite-state systems, such as rectangular automata, finding finite quotient spaces is often essential to the decidability of verification problems. In this chapter we study the similarity and bisimilarity quotient systems of the time-abstract transition systems of rectangular automata with uniform dynamics. We prove that bisimilarity degenerates to equality in two or more dimensions. Then we show that every two-dimensional positive rectangular automaton with closed uniform dynamics has finite time-abstract similarity quotient. In this case, similarity coincides with the intersection of the region equivalence relations defined by the extremal slopes. Next, we give our most surprising result: there exists a three-dimensional rectangular automaton A with uniform dynamics such that the only simulation on the time-abstract transition system S_A^{abs} is the equality relation. Last, we define the asynchronous transition system of a rectangular automaton, and show that asynchronous similarity degenerates to equality in four or more dimensions.

6.1 2D

6.1.1 Bisimilarity

Theorem 6.1.1 [Hen95] *There exists a two-dimensional rectangular automaton A with uniform dynamics such that the equality relation is the only bisimulation on the time-abstract transition system S_A^{abs}.*

Proof. The automaton A has only one vertex, v. Each variable drifts between slopes 1 and 2: $act(v) = [1,2]^2$. The state space is essentially the unit square: $inv(v) = [0,1]^2$. Every state is an initial state: $init(v) = [0,1]^2$. There are four edges. The first two, e_i, $i = 1,2$, test for $x_i = 1$, and perform no assignments: $update(e_i) = \emptyset$, $preguard(e_1) = postguard(e)_1 = \{1\} \times [0,1]$, and $preguard(e_2) = postguard(e)_2 = [0,1] \times \{1\}$. The second two, e'_i, $i = 1,2$, reset the ith component: $update(e'_i) = \{i\}$, $preguard(e'_i) = [0,1]^2$, $postguard(e'_1) = \{0\} \times [0,1]$, and $postguard(e'_2) = [0,1] \times \{0\}$. Each edge is its own label.

A simple algorithm for computing the bisimilarity quotient of a systems maintains a partition $\mathcal{R}$ of the state space [BFH90]. The partition is refined whenever there exist $R_1, R_2 \in \mathcal{R}$ and an event π such that both $R_1 \cap Pre^{\pi}(R_2)$ and $R_1 \setminus Pre^{\pi}(R_2)$ are nonempty. In this case, R_1 is split into these two parts. When no more splitting is possible, the bisimilarity quotient has been computed.

We show that every two distinct points of S_A are eventually put into different partition elements. Since there is only one vertex, we use $[0,1]^2$ as the state space for S_A to simplify the notation. The first action of the algorithm is to use Pre^{e_1} to split $[0,1]^2$ into $\{1\} \times [0,1]$ and $[0,1) \times [0,1]$. Next both classes are intersected with

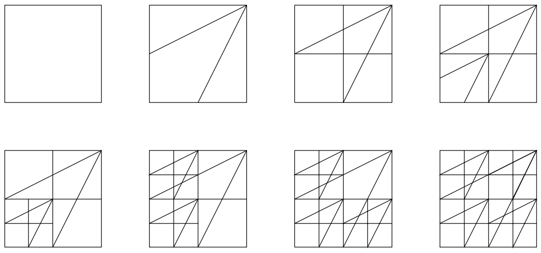

Figure 6.1: The only bisimulation is equality

$Pre^{e_2} = [0,1]\times\{1\}$. Then $Pre^{time}\{1\}\times[0,1)$ is split off, and then $Pre^{time}[0,1)\times\{1\}$. Thus far, our algorithm has computed the partition shown in the second square of Figure 6.1. Now $Pre^{e'_1}$ and $Pre^{e'_2}$ refine the partition as shown in the third square of the figure. At this point, the unit square has been subdivided into four squares of side length $\frac{1}{2}$. Further execution of *BisimApprox* subdivides each of these four squares into four squares of side length $\frac{1}{4}$. We show the subdivision of the lower left-hand corner in the fourth and fifth squares of Figure 6.1. In the sixth through eighth squares, we continue until the the unit square has been divided into sixteen squares of side length $\frac{1}{4}$. It follows that not only does the algorithm fail to terminate, but every point is separated from every other point. Thus the only bisimulation on S_A is equality. ■

6.1.2 Similarity

The analysis of two-dimensional rectangular automata requires some concepts from the theory of multirate automata that previously have only been touched upon.

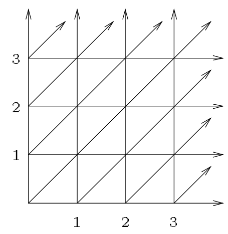

Figure 6.2: Region equivalence on the plane

Definition 6.1.2 Define the *region equivalence relation* $\equiv^{reg}$ on $\mathbb{R}^n$ by $\mathbf{x} \equiv^{reg} \mathbf{u}$ iff $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{u} \rfloor$, $\lceil \mathbf{x} \rceil = \lceil \mathbf{u} \rceil$, and for all $1 \leq i, j \leq n$, $\lfloor x_j - x_i \rfloor = \lfloor u_j - u_i \rfloor$. See Figure 6.2. ■

Definition 6.1.3 For $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{s} \in \mathbb{N}^n_{>0}$, define $\mathbf{s}^{-1}\mathbf{y}$ to be the n-vector $(y_1/s_1, \ldots, y_n/s_n)$. For a set $R \subset \mathbb{R}^n$, define $\mathbf{s}^{-1}R =_{\text{def}} \{\mathbf{s}^{-1}\mathbf{y} \mid \mathbf{y} \in R\}$. ■

If M is a multirate automaton M with activity rectangle $\{\mathbf{s}\}$, then by scaling by $\mathbf{s}^{-1}$, we can convert M into a timed automaton. Applying Theorem 3.1.6, we obtain a bisimulation on the time-abstract transition system S^{abs}_M.

Definition 6.1.4 Let ℓ be the least common multiple of the coordinate values of the vector $\mathbf{s} \in \mathbb{N}^n_{>0}$. We define the *skewed region equivalence relation* $\equiv^{reg}_{\mathbf{s}}$ on $\mathbb{R}^n$ by $\mathbf{y} \equiv^{reg}_{\mathbf{s}} \mathbf{y}'$ iff $\ell(\mathbf{s}^{-1}\mathbf{y}) \equiv^{reg} \ell(\mathbf{s}^{-1}\mathbf{y}')$. ■

Theorem 6.1.5 [NOSY93,ACHH93] *For every n-dimensional integral multirate automaton M with uniform activity rectangle $\{\mathbf{s}\}$, the equivalence relation defined by $(v, \mathbf{x}) \sim (w, \mathbf{y})$ iff $v = w$ and $\mathbf{x} \equiv^{reg}_{\mathbf{s}} \mathbf{y}$ is a bisimulation on the time-abstract transition system S^{abs}_M.* ■

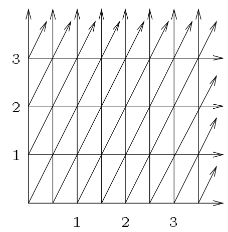

Figure 6.3: Skewed region equivalence $\equiv_{\mathbf{s}}^{reg}$, where $s_2 = 2s_1$

Theorem 6.1.6 *For every two-dimensional positive rectangular automaton A with closed uniform activity rectangle, the time-abstract transition system S_A^{abs} has finite similarity quotient.*

Proof. By redefining the time unit, it suffices to assume that all of the rational rectangles defining A are integral. There exist integers $a_i, b_i \in \mathbb{N}$, $i = 1, 2$, such that for every vertex $v \in V$,

$$act(v) = [a_1, b_1] \times [a_2, b_2].$$

We assume $a_1, a_2 > 0$, a similar analysis applies if this is not the case. The continuous dynamics of A are thus completely specified by the nondeterministic differential equation $\frac{dx_2}{dx_1} \in [\frac{a_2}{b_1}, \frac{b_2}{a_1}]$. We assume that $\frac{b_1}{a_2}$ and $\frac{b_2}{a_1}$ are positive integers. Again, similar analysis applies if this is not the case. Let $\mathbf{p}$ be the vector (b_1, a_2), and let $\mathbf{q}$ be the vector (a_1, b_2). We show that if $\mathbf{y} \equiv_{\mathbf{p}}^{reg} \mathbf{y}'$ and $\mathbf{y} \equiv_{\mathbf{q}}^{reg} \mathbf{y}'$, then for every vertex $v \in V_A$, $(v, \mathbf{y}) \equiv_{S_A^{abs}}^{sim} (v, \mathbf{y}')$.

Let $v \in V$ be any vertex. For a rectangle $B \subset \mathbb{R}^2$ and $\mathbf{y} \in [0,1]^2$, define $cone^B(\mathbf{y}) =_{\mathrm{def}} \{\mathbf{y}' \in [0,1]^2 \mid \mathbf{y} = \mathbf{y}' \vee \exists \delta > 0.\, \frac{\mathbf{y}' - \mathbf{y}}{\delta} \in B\}$. Put $cone(\mathbf{y}) =_{\mathrm{def}} cone^{act(v)}(\mathbf{y})$. The set $cone(\mathbf{y})$ is the set of all time-step successors of $\mathbf{y}$ in the

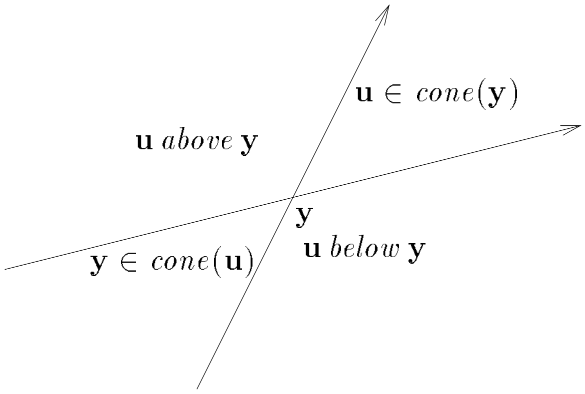

Figure 6.4: $cone(\mathbf{y})$ splits the unit square into four pieces

unit square. The cone of $\mathbf{y}$ splits the unit square into four pieces, as depicted in Figure 6.4. Define $ray^{\mathbf{s}}(\mathbf{y}) = \{\mathbf{y} + \delta\mathbf{s} \mid \delta \geq 0\}$, the ray of slope $\mathbf{s}$ beginning at $\mathbf{y}$. Define $\mathbf{u}\ below\ \mathbf{y}$ iff $\mathbf{u} \notin cone(\mathbf{y}) \setminus ray^{\mathbf{p}}(\mathbf{y})$ and $ray^{\mathbf{q}}(\mathbf{u}) \cap cone(\mathbf{y}) \neq \emptyset$. Define $\mathbf{u}\ above\ \mathbf{y}$ iff $\mathbf{u} \notin cone(\mathbf{y}) \setminus ray^{\mathbf{q}}(\mathbf{y})$ and $ray^{\mathbf{p}}(\mathbf{u}) \cap cone(\mathbf{y}) \neq \emptyset$. Then for all $\mathbf{u}, \mathbf{y} \in [0,1]^2$, either $\mathbf{u} \in cone(\mathbf{y})$, $\mathbf{y} \in cone(\mathbf{u})$, $\mathbf{u}\ below\ \mathbf{y}$, or $\mathbf{u}\ above\ \mathbf{y}$.

We define a transition system on the unit square that encapsulates the behavior of A. Let S be the transition system $([0,1]^2, \Sigma_S, \rightarrow, [0,1]^2, \Pi, \models)$. There are four atomic propositions: $\Pi = \{\pi_{00}, \pi_{01}, \pi_{10}, \pi_{11}\}$. The satisfaction relation is defined by $\mathbf{y} \models \pi_{ij}$ iff $y_i = j$. Thus the atomic propositions distinguish between the four corners, the four sides of the square without their endpoints, and the interior of the square. Thus for each of the atomic propositions π_{ij} there exists an integral rectangle $B^{ij} \subset [0,1]^2$ such that $B^{ij} = \{(x,y) \in [0,1]^2 \mid (x,y) \models \pi_{ij}\}$. Equivalence with respect to the atomic propositions is equivalent to equivalence with respect to inclusion in all integral rectangles. For each pair π_{ij} and $\pi_{k\ell}$ of atomic propositions, and each subset U of $\{1,2\}$, there is an event $\sigma(\pi_{ij}, U, \pi_{k\ell}) \in \Sigma_S$, corresponding to an edge e in a rectangular automaton with $preguard(e) = B^{ij}$, $postguard(e) = B^{k\ell}$, and $update(e) = U$. There is one additional edge label, $time$. Thus

$$\Sigma_S = \{time\} \cup \{\sigma(\pi_{ij}, U, \pi_{k\ell}) \mid \pi_{ij}, \pi_{k\ell} \in \Pi, U \subset \{1,2\}\}.$$

The transition relation is defined by $\mathbf{y} \stackrel{time}{\rightarrow} \mathbf{y}'$ iff $\mathbf{y}' \in cone(\mathbf{y})$, and $\mathbf{y} \stackrel{\sigma(\pi_{ij},U,\pi_{k\ell})}{\rightarrow} \mathbf{y}'$ iff $\mathbf{y} \models \pi_{ij}$, $\mathbf{y}' \models \pi_{k\ell}$, and for each $i \notin U$, $y_i = y'_i$.

An *integral unit square* is a subset of the plane of the form $[k, k+1] \times [\ell, \ell+1]$, where $k, \ell \in \mathbb{N}$. If $\preceq$ is a simulation relation on S, it follows that the relation $\preceq'$ defined on Q_A by $(v, \mathbf{y}) \preceq' (v', \mathbf{y}')$ iff $v = v'$ and $\mathbf{y} - \lfloor \mathbf{y} \rfloor \preceq \mathbf{y}' - \lfloor \mathbf{y}' \rfloor$ is a simulation on S_A^{abs}. For suppose $(v, \mathbf{y}) \preceq' (v, \mathbf{u})$. If $(v, \mathbf{y}) \stackrel{time}{\rightarrow} (v, \mathbf{y}')$, then there exist $\mathbf{y}_0, \ldots, \mathbf{y}_k$ such that $\mathbf{y} = \mathbf{y}_0, \mathbf{y}' = \mathbf{y}_k$, and for each i, $(v, \mathbf{y}_i) \stackrel{time}{\rightarrow} (v, \mathbf{y}_{i+1})$ and $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$ lie in the same integral unit square $[k_i, k_i+1] \times [\ell_i, \ell_i+1]$. If we put $\mathbf{y}'_i = \mathbf{y}_i - (k_i, \ell_i)$, then for each i, $\mathbf{y}'_i \in [0,1]^2$, and $\mathbf{y}'_i \stackrel{time}{\rightarrow} \mathbf{y}'_{i+1}$ in S. It follows that there exist $\mathbf{u}'_0, \ldots, \mathbf{u}'_k$ such that $\mathbf{u} = \mathbf{u}'_0 + [k_0, \ell_0]$, and for each i, $\mathbf{u}_i \stackrel{time}{\rightarrow} \mathbf{u}_{i+1}$ in S. If $\mathbf{u}_i =_{\mathrm{def}} \mathbf{u}'_i + [k_i, \ell_i]$, then $(v, \mathbf{u}_i) \stackrel{time}{\rightarrow} (v, \mathbf{u}_{i+1})$ for each i. It follows that $(v, \mathbf{u}) \stackrel{time}{\rightarrow} (v, \mathbf{u}_k)$ and $(v, \mathbf{y}') \preceq' (v, \mathbf{u}_k)$.

If $(v, \mathbf{y}) \stackrel{e}{\rightarrow} (v', \mathbf{y}')$ for some $e \in E$, then $\mathbf{y} \in preguard(e)$, $\mathbf{y}' \in postguard(e)$, and for each $i \notin update(e)$, $y_i = y'_i$. Put $U = update(e)$, and let $[k, k+1] \times [\ell, \ell+1]$, $[k', k'+1] \times [\ell', \ell'+1]$ be integral unit squares containing $\mathbf{y}$ and $\mathbf{y}'$ respectively. Let π_{ij}, $\pi_{i'j'}$ be the atomic propositions such that $B^{ij} = preguard(e) - (k, \ell)$ and $B^{i'j'} = postguard(e) - (k', \ell')$. It follows that $\mathbf{y} - (k, \ell) \stackrel{\sigma(\pi_{ij},U,\pi_{i'j'})}{\rightarrow} \mathbf{y}' - (k', \ell')$. Since $\mathbf{y} - (k, \ell) \preceq \mathbf{u} - (k, \ell)$, there exists a $\mathbf{u}' - (k', \ell')$ such that $\mathbf{u} - (k, \ell) \stackrel{\sigma(\pi_{ij},U,\pi_{i'j'})}{\rightarrow} \mathbf{u}' -$

(k', ℓ'). Then $(v, \mathbf{u}) \xrightarrow{e} (v', \mathbf{u}')$, and $(v', \mathbf{y}') \preceq' (v', \mathbf{u}')$.

We have proven that every simulation $\preceq$ on S extends naturally to a simulation $\preceq'$ on S_A^{abs}. By cutting off the simulation at h, where A is h-definable, we obtain a coarser simulation $\preceq''$. If the similarity quotient of S is finite, then the relation $\preceq''_S$ is a simulation relation on S_A^{abs} such that if we define $(v, \mathbf{y}) \equiv_S^{sim''} (v', \mathbf{y}')$ iff $(v, \mathbf{y}) \preceq''_S (v', \mathbf{y}')$ and $(v', \mathbf{y}') \preceq''_S (v, \mathbf{y})$, then $\equiv_S^{sim''}$ is an equivalence relation with finite quotient no coarser than the similarity quotient of S_A^{abs}. It follows that S_A^{abs} has finite similarity quotient.

We now proceed to find a simulation relation on S. Define the relation $\preceq$ on the unit square $[0, 1]^2$ by $\mathbf{y} \preceq \mathbf{u}$ iff the following four conditions obtain:

1. Either $\mathbf{u}$ *above* $\mathbf{y}$ and $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$, or $\mathbf{u}$ *below* $\mathbf{y}$ and $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}$, or $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$ and $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}$.

2. For $i = 1, 2$, $k = 0, 1$, $u_i = k$ iff $y_i = k$.

3. $u_2 > y_2$ implies $(0, u_2) \equiv_{\mathbf{p}}^{reg} (0, y_2)$, and $u_2 < y_2$ implies $(0, u_2) \equiv_{\mathbf{q}}^{reg} (0, y_2)$.

4. $u_1 > y_1$ implies $(u_1, 0) \equiv_{\mathbf{q}}^{reg} (y_1, 0)$, and $u_1 < y_1$ implies $(u_1, 0) \equiv_{\mathbf{p}}^{reg} (y_1, 0)$.

We prove that $\preceq$ is a simulation relation on S. The main idea is that if $\mathbf{u}$ is above $\mathbf{y}$, then to match a time transition from $\mathbf{y}$, a move at minimum slope should be made from $\mathbf{u}$. If $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$, then an appropriate minimal slope time transition can be selected. Similarly, if $\mathbf{u}$ is below $\mathbf{y}$, then a time transition from $\mathbf{y}$ should be matched by a time transition from $\mathbf{u}$ at maximum slope; if $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}$, then an appropriate maximum slope time transition is available.

We use one basic fact: if $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$, $u_2 \geq y_2$, and $u_1 \leq y_1$, then conditions 2-4 hold between $\mathbf{y}$ and $\mathbf{u}$.

Suppose $\mathbf{y} \preceq \mathbf{u}$. Notice that for each $\pi_{ij} \in \Pi$, $\mathbf{y} \models \pi_{ij}$ iff $\mathbf{u} \models \pi_{ij}$ by condition (2). It remains to show that $\preceq$ has the simulation property. Suppose $\mathbf{y} \stackrel{time}{\rightarrow} \mathbf{y}'$. If $\mathbf{y}' \in cone(\mathbf{u})$, then $\mathbf{u} \stackrel{time}{\rightarrow} \mathbf{y}'$ as well. So assume $\mathbf{y}' \notin cone(\mathbf{u})$. The proof is broken into cases, based upon which disjunct of (1) is satisfied. Suppose $\mathbf{u}$ *above* $\mathbf{y}$ and $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$. In this case, $\mathbf{u}$ *above* $\mathbf{y}'$ as well. We show that there exists a $\delta \geq 0$ such that $\mathbf{y}' \preceq \mathbf{u}'$, where $\mathbf{u}' = \mathbf{u} + \delta\mathbf{p}$.

An equivalence classes of $\equiv_{\mathbf{p}}^{reg}$ is an *interior region* if it is open, and a *boundary region* otherwise. Interior regions are bounded by lines of slope (0,1), (1,0), or $\mathbf{p}$. Boundary regions are either points, or line segments of slope (0,1), (1,0), or $\mathbf{p}$. It suffices to suppose that the line segment between $\mathbf{y}$ and $\mathbf{y}'$ contains points from at most two $\equiv_{\mathbf{p}}^{reg}$ equivalence classes, and if $\mathbf{y}$ lies in an interior region, and $\mathbf{y}'$ lies in a boundary region R, then $\mathbf{y}'$ is the only point in the intersection of R with the line segment from $\mathbf{y}$ to $\mathbf{y}'$. If not, then the transition from $\mathbf{y}$ to $\mathbf{y}'$ is broken into several time transitions, each with the above property; then a time transition from $\mathbf{u}$ is defined in a stepwise manner.

If $\mathbf{y}$ lies in an interior region, and $\mathbf{y}'$ lies in a boundary region R, it follows from $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$ that there exists exactly one $\delta \geq 0$ such that $\mathbf{u} + \delta\mathbf{p} \in R$. In this case condition (1) holds between $\mathbf{y}$ and $\mathbf{u}' =_{\mathrm{def}} \mathbf{u} + \delta$, because $\mathbf{u}'$ *above* $\mathbf{y}'$ and $\mathbf{u}' \equiv_{n}^{reg} \mathbf{y}'$. Condition (2) holds because $\equiv_{n}^{reg}$ respects equivalence up to the propositions in Π. If R is a point or a vertical or horizontal line segment, then conditions (3) and (4) hold immediately. Finally it is not possible for R to be a line segment of slope $\mathbf{p}$, because $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$ and the equivalence classes of $\equiv_{\mathbf{p}}^{reg}$ are convex.

Unless $\mathbf{y}' = \mathbf{y}$ (in which case we choose $\mathbf{u}' = \mathbf{u}$), it is not possible for $\mathbf{y}$ and $\mathbf{y}'$ to lie in the same boundary region. This would require $\mathbf{y}' = \mathbf{y} + \delta'\mathbf{p}$ for some $\delta' > 0$, which would further require that $\mathbf{y} = \mathbf{u} + \delta''\mathbf{p}$ for some $\delta \in \mathbb{R}$, which is contrary to supposition.

Suppose $\mathbf{y}$ lies in a boundary region R and $\mathbf{y}'$ lies in an interior region. Case: R is vertical. In this case $u_1 = y_1$ and $u_2 > y_2$. If $y_2' \geq u_2$, pick δ so that $\mathbf{u}' =_{\text{def}} \mathbf{u}+\delta\mathbf{p}$ satisfies $u_2' = y_2'$. Then $\mathbf{u}'$ *above* $\mathbf{y}'$ and by the convexity of $\equiv_{\mathbf{p}}^{reg}$-equivalence classes, $\mathbf{u}' \equiv_{\mathbf{p}}^{reg} \mathbf{y}'$. It follows from the basic fact that $\mathbf{y}' \preceq \mathbf{u}'$. If $y_2' < u_2$, the δ can be chosen to be sufficiently small so that the basic fact establishes $\mathbf{y}' \preceq \mathbf{u}'$.

Case: R is horizontal. In this case, the interior region R' containing $\mathbf{y}'$ is a right triangle with hypotenuse of slope $\mathbf{p}$. In this case $u_2 = y_2$ and $u_1 < y_1$. Pick δ so that $u_1' = y_1'$. It follows from the shape of R' that $\mathbf{u}' \equiv_{\mathbf{p}}^{reg} \mathbf{y}'$, and then from the basic fact that $\mathbf{y}' \preceq \mathbf{u}'$.

The remaining case, in which R is a line segment of slope $\mathbf{p}$, is impossible, because it implies that $\mathbf{u} \in cone(\mathbf{y})$ or vice versa, contrary to supposition.

Finally, suppose $\mathbf{y}$ and $\mathbf{y}'$ lie in the same interior region R. Case: $u_2 > y_2'$. In this case put $\mathbf{u}' =_{\text{def}} \mathbf{u}$. Then $\mathbf{u}$ *above* $\mathbf{y}'$. If $u_1 \leq y_1$, then $u_1 \leq y_1'$, so by the basic fact, $\mathbf{y}' \preceq \mathbf{u}$. Otherwise $u_1 > y_1$. By (4), $(u_1, 0) \equiv_{\mathbf{q}}^{reg} (y_1, 0)$. Since $y_1 \leq y_1' \leq u_1$, and since the $\equiv_{\mathbf{q}}^{reg}$-equivalence classes are convex, condition (4) is satisfied between $\mathbf{y}'$ and $\mathbf{u}$. Satisfaction of condition (3) follows from the fact that $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}'$. Thus $\mathbf{y}' \preceq \mathbf{u}$. Case: $u_2 \leq y_2'$. Let δ be such that $u_2' = y_2'$. It follows from the basic remark that $\mathbf{y}' \preceq \mathbf{u}'$.

This completes the analysis of $\stackrel{time}{\rightarrow}$ transitions for the case of $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}$ and $\mathbf{u}$ *above* $\mathbf{y}$. The case of $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}$ and $\mathbf{u}$ *below* $\mathbf{y}$ is symmetrical. In the final case, $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}'$, $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}'$, and $\mathbf{u} \in cone(\mathbf{y})$. Suppose $\mathbf{y} \stackrel{time}{\rightarrow} \mathbf{y}'$ and $\mathbf{y}' \notin cone(\mathbf{u})$. There are three cases: (a) on the line segment between $\mathbf{y}$ and $\mathbf{y}'$ there exists a point $\mathbf{y}^*$ such that $\mathbf{u} \in ray^{\mathbf{p}}(\mathbf{y}^*)$, (b) on the line segment between $\mathbf{y}$ and $\mathbf{y}'$ there exists a point $\mathbf{y}^*$ such that $\mathbf{u} \in ray^{\mathbf{q}}(\mathbf{y}^*)$, (c) neither (a) nor (b). If (a), then $\mathbf{u}$ *below* $\mathbf{y}^*$ and $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}^*$. In this case $\mathbf{u}'$ is picked as detailed above to counter a move from $\mathbf{y}^*$ to $\mathbf{y}'$. If (b), then $\mathbf{u}$ *above* $\mathbf{y}^*$ and $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}^*$, and again, $\mathbf{u}'$ is picked as detailed above to counter a move from $\mathbf{y}^*$ to $\mathbf{y}'$. If (c), then $\mathbf{u} \equiv_{\mathbf{p}}^{reg} \mathbf{y}'$ and $\mathbf{u} \equiv_{\mathbf{q}}^{reg} \mathbf{y}'$, and conditions (2)-(4) hold between $\mathbf{u}$ and $\mathbf{y}'$ Thus $\mathbf{u}'$ is chosen to be $\mathbf{u}$.

Then either $\mathbf{u} = \mathbf{y}^* + \delta\mathbf{p}$ or $\mathbf{u} = \mathbf{y}^* + \delta\mathbf{q}$ for some $\delta \geq 0$. If the former, use pick $\mathbf{u}'$ according to the strategy for a move from $\mathbf{y}^*$ to $\mathbf{y}$,

Now suppose $\mathbf{y} \stackrel{\sigma(\pi_{ij},U,\pi_{k\ell})}{\rightarrow} \mathbf{y}'$, where $\pi_{ij}, \pi_{k\ell} \in \Pi$ and $U \subset \{1,2\}$. The existence of a $\mathbf{u}'$ such that $\mathbf{u} \stackrel{\sigma(\pi_{ij},U,\pi_{k\ell})}{\rightarrow} \mathbf{u}'$ is guaranteed by conditions (3) and (4), and the fact that if $(0,z) \equiv_{\mathbf{s}}^{reg} (0,z')$ for some slope vector $\mathbf{s}$, then for every $w \in [0,1]$ there exists a $w' \in [0,1]$ such that $(w,z) \equiv_{\mathbf{s}}^{reg} (w',z')$ and $(w',z') \models \pi$ iff $(w,z) \models \pi$ for all $\pi \in \Pi$. ■

Corollary 6.1.7 *The model-checking problem for the universal fragment of CTL*, on the time-abstract transition system of a positive two-dimensional rectangular automaton A with closed uniform activity rectangle, can be solved in space polynomial in the size of A.*

Proof. By the proof of Theorem 6.1.6, there is a singly exponential number of equivalence classes in the similarity quotient of a rectangular automaton of the specified class. Inclusion in PSPACE follows from the fact that CTL* model-checking can be performed on-the-fly in space logarithmic in the size of the system (Theorem 2.1.27). ■

Since the universal fragment of CTL* subsumes linear temporal logic, the model-checking problem is already PSPACE-hard on finite graphs, and so the problem is PSPACE-complete for positive 2D rectangular automata with closed uniform activity rectangle.

Corollary 6.1.8 *The control decision problem, on the time-abstract transition system of a positive two-dimensional rectangular automaton A with closed uniform activity rectangle, can be solved in time exponential in the size of A.*

Proof. By Theorem 2.1.38. To make the time-abstract transition system S_A^{abs} standard, we add one atomic proposition π_B for every h-definable integral rectangle, where A is h-definable. Put $(v, \mathbf{x}) \models \pi_B$ iff $\mathbf{x} \in B$. The proof of Theorem 6.1.6 already takes these new atomic propositions into account. ■

6.2 3D and Beyond

6.2.1 Synchronous Analysis

Theorem 6.2.1 *There exists a three-dimensional rectangular automaton A with uniform dynamics such that the equality relation is the only simulation on the* *time-abstract transition system* S_A^{abs}.

Proof. The rectangular automaton A has a single vertex. Therefore we ignore the discrete part of the state, and think of a state of A as a vector in $\mathbb{R}^3$. The activity rectangle of A is $\{1\} \times \{1\} \times [1,2]$. The invariant rectangle of A is the unit cube $[0,1]^3$. For each of the finitely many triples $\kappa = (B, U, B')$, where $B, B' \subset [0,1]^3$ are integer-bounded rectangles, and $U \subset \{1,2,3\}$, the rectangular automaton A has an edge e_κ (from the sole vertex to itself) with $\mathit{preguard}(e_\kappa) = B$, $\mathit{update}(e_\kappa) = U$, $\mathit{postguard}(e_\kappa) = B'$, and $\mathit{event}(e_\kappa) = \kappa$. Thus the set of events coincides with this finite set of triples. Call this set $\mathcal{G}$.

The proof proceeds by proving that no element from a set B is simulated by any element from the set C, and vice versa. To prove this, we give a finite sequence $\bar{g}$ of guarded commands that can be executed by every element of B in such a way that the resulting state is not simulated by any state that can be produced by executing $\bar{g}$ from any element of C. Then we do the reverse.

We refer to the three coordinates as x, y, and z. Put $S = S_A^{abs}$. It suffices to show that for all vectors $(x,y,z), (x',y',z')$ in the unit cube $[0,1]^3$, $(x,y,z) \preceq_S (x',y',z')$ implies $(x,y,z) = (x',y',z')$. We do so in several steps. First we show that for every n, if $x \leq \frac{1}{2^n}$ and $x' > \frac{1}{2^n}$ then $(x,y,z) \not\preceq_S (x',y',z')$ and $(x',y',z') \not\preceq_S (x,y,z)$. We prove the analogous results for y and z simultaneously by induction. Second, we extend these results from numbers of the form $\frac{1}{2^n}$ to every dyadic rational $\frac{k}{2^n} \in [0,1]$. The theorem then follows from the density of the dyadic rationals.

Rather than writing out the triples explicitly, we save space by using a traditional guarded command notation. For example, $x = 1 \rightarrow z := 0$ denotes the triple $(\{1\} \times \mathbb{R} \times \mathbb{R}, \{3\}, \mathbb{R} \times \mathbb{R} \times \{0\})$. We define X to be the guarded command $\mathit{true} \rightarrow x := 0$, and similarly define Y and Z. For a guarded command g representing the triple κ, Define $\xrightarrow{g}$ by $\mathbf{u} \xrightarrow{g} \mathbf{u}'$ iff $\mathbf{u} \xrightarrow{\kappa} \mathbf{u}'$ in A (which is equivalent to $\mathbf{u} \xrightarrow{e_\kappa} \mathbf{u}'$). Again to save space, we write T instead of the abstract time label time. So $\mathbf{u} \xrightarrow{T} \mathbf{u}'$ iff $\mathbf{u} \xrightarrow{\mathit{time}} \mathbf{u}'$ iff $\mathbf{u} = \mathbf{u}'$ or there exists a time $t \in \mathbb{R}_{>0}$ with $\frac{\mathbf{u}' - \mathbf{u}}{t} \in \{1\} \times \{1\} \times [1, 2]$.

Given a predicate $\phi(x, y, z)$ with free variables in $\{x, y, z\}$, we define

$$[\![\phi]\!] =_{\mathrm{def}} \{\mathbf{u} \in [0,1]^3 \mid \phi[x := u_1; y := u_2; z := u_3] \text{ is true}\}.$$

If $\psi(x, y, z)$ is another such predicate, and $g \in \mathcal{G} \cup \{T\}$, define $[\![\phi]\!] \xrightarrow{g} [\![\psi]\!]$ iff for every $\mathbf{u} \in [\![\phi]\!]$, there exists a $\mathbf{u}' \in [\![\psi]\!]$ such that $\mathbf{u} \xrightarrow{g} \mathbf{u}'$. Extend the definition of $\xrightarrow{g}$ to strings $\bar{g} \in (\mathcal{G} \cup \{T\})^*$ in the natural way. Concatenation is denoted by a semicolon, so $g; g'$ is a string of length two. Define $\mathit{Post}_{\bar{g}}([\![\phi]\!])$ to be the set $\{\mathbf{u}' \in [0,1]^3 \mid \exists \mathbf{u} \in [\![\phi]\!].\, \mathbf{u} \xrightarrow{\bar{g}} \mathbf{u}'\}$ of $\bar{g}$-successors of elements of $[\![\phi]\!]$. If no element of $[\![\phi]\!]$ is simulated by any element of $[\![\psi]\!]$, we write $[\![\phi]\!] \not\preceq_S [\![\psi]\!]$. We say that three predicates ϕ, ψ, and θ with free variables in $\{x, y, z\}$ are *synchronously distinguishable* if $[\![\alpha]\!] \not\preceq_S [\![\beta]\!]$ for all $\alpha \neq \beta \in \{\phi, \psi, \theta\}$.

We prove by induction that for all $n \geq 1$,

- $1(n)$: $x < \frac{1}{2^n}$, $x = \frac{1}{2^n}$, and $x > \frac{1}{2^n}$ are synchronously distinguishable,
- $2(n)$: $y < \frac{1}{2^n}$, $y = \frac{1}{2^n}$, and $y > \frac{1}{2^n}$ are synchronously distinguishable,
- $3(n)$: $z < \frac{1}{2^n}$, $z = \frac{1}{2^n}$, and $z > \frac{1}{2^n}$ are synchronously distinguishable.

In fact we prove that $1(n+1) \wedge 2(n+1) \wedge 3(n)$ implies $1(n+2) \wedge 2(n+2) \wedge 3(n+1)$, which makes the base case somewhat lengthy. Since x and y occupy symmetrical positions, we prove results for x only, and refer to the x result when a y citation is needed.

A). $[\![x \leq \frac{1}{2}]\!] \not\preceq_S [\![x > \frac{1}{2}]\!]$:
$[\![x \leq \frac{1}{2}]\!] \stackrel{Y;Z;T;x=1\,\wedge\, z=1}{\longrightarrow} [\![true]\!]$, but $Post_{Y;Z;T;x=1\,\wedge\, z=1}([\![x > \frac{1}{2}]\!]) = \emptyset$.

B). $[\![x \geq \frac{1}{2}]\!] \not\preceq_S [\![0 \leq x < \frac{1}{2}]\!]$:
$[\![x \geq \frac{1}{2}]\!] \stackrel{Y;Z;T;x=1}{\longrightarrow} [\![y \leq \frac{1}{2}]\!]$, but $Post_{Y;Z;T;x=1}([\![x < \frac{1}{2}]\!]) \subset [\![y > \frac{1}{2}]\!]$. Thus B) follows from A).

C). 1(1) and 2(1): immediate from A) and B).

D). $[\![z \leq \frac{1}{2}]\!] \not\preceq_S [\![z > \frac{1}{2}]\!]$:
$[\![z \leq \frac{1}{2}]\!] \stackrel{X;Y;T;z=1}{\longrightarrow} [\![x = \frac{1}{2}]\!]$, but $Post_{X;Y;T;z=1}([\![z > \frac{1}{2}]\!]) \subset [\![x < \frac{1}{2}]\!]$. Thus D) follows from 1(1).

E). $[\![z \geq 2y]\!] \not\preceq_S [\![z < 2y]\!]$:
$[\![z \geq 2y]\!] \stackrel{X;T;z=1}{\longrightarrow} [\![y \leq \frac{1}{2}]\!]$, but $Post_{X;T;z=1}([\![z < 2y]\!]) \subset [\![y > \frac{1}{2}]\!]$. Thus E) follows from 2(1).

F). $[\![\frac{1}{2} \geq x \geq \frac{1}{4}]\!] \not\preceq_S [\![x < \frac{1}{4}]\!]$:
$[\![\frac{1}{2} \geq x \geq \frac{1}{4}]\!] \stackrel{Y;Z;T}{\longrightarrow} [\![x = \frac{1}{2} \;\wedge\; z \leq \frac{1}{2} \;\wedge\; z \geq 2y]\!]$, but

$$Post_{Y;Z;T}([\![x < \frac{1}{4}]\!]) \cap [\![x = \frac{1}{2} \;\wedge\; z \geq 2y]\!] \subset [\![z > \frac{1}{2}]\!].$$

Thus F) follows from 1(1), D), and E).

G). $[\![x \leq \frac{1}{4}]\!] \not\preceq_S [\![x > \frac{1}{4}]\!]$:

$[\![x \leq \frac{1}{4}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{1}{2} \wedge y \geq \frac{1}{4}]\!]$, but $Post_{Y;Z;T}([\![x > \frac{1}{4}]\!]) \cap [\![x = \frac{1}{2}]\!] \subset [\![y < \frac{1}{4}]\!]$. Thus G) follows from 1(1) and F).

H). 1(2) and 2(2): immediate from F), G), 1(1), and 2(1).

I). $[\![z \geq \frac{1}{2}]\!] \not\preceq_S [\![z < \frac{1}{2}]\!]$:

$[\![z \geq \frac{1}{2}]\!] \stackrel{X;Y;T;z=1}{\rightarrow} [\![x \leq \frac{1}{4}]\!]$, but $Post_{XYT;z=1}([\![z < \frac{1}{2}]\!]) \subset [\![x > \frac{1}{4}]\!]$. Thus I) follows from G).

J) 3(1): immediate from D) and I).

The proof of the base case is complete. Recall that the induction step consists of proving that $1(n+1) \wedge 2(n+1) \wedge 3(n)$ implies $1(n+2) \wedge 2(n+2) \wedge 3(n+1)$. The proof of this step is a slight variant of the proof of the base case. Assume $1(n+1) \wedge 2(n+1) \wedge 3(n)$.

K). $[\![z \leq \frac{1}{2^{n+1}}]\!] \not\preceq_S [\![z > \frac{1}{2^{n+1}}]\!]$:

$[\![z \leq \frac{1}{2^{n+1}}]\!] \stackrel{X;Y;T}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge z \leq \frac{1}{2^n}]\!]$, but

$$Post_{X;Y;T}([\![z > \frac{1}{2^{n+1}}]\!]) \cap [\![x = \frac{1}{2^{n+1}}]\!] \subset [\![z > \frac{1}{2^n}]\!].$$

Thus K) follows from $1(n+1)$ and $3(n)$.

L). $[\![\frac{1}{2^{n+1}} \geq x \geq \frac{1}{2^{n+2}}]\!] \not\preceq_S [\![x < \frac{1}{2^{n+2}}]\!]$:

$[\![\frac{1}{2^{n+1}} \geq x \geq \frac{1}{2^{n+2}}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge z \leq \frac{1}{2^{n+1}} \wedge z \geq 2y]\!]$, but

$$Post_{Y;Z;T}([\![x < \frac{1}{2^{n+2}}]\!]) \cap [\![x = \frac{1}{2^{n+1}} \wedge z \geq 2y]\!] \subset [\![z > \frac{1}{2^{n+1}}]\!].$$

Thus L) follows from $1(n+1)$, E), and K).

M). $[\![x \leq \frac{1}{2^{n+2}}]\!] \not\preceq_S [\![x > \frac{1}{2^{n+2}}]\!]$:
$[\![x \leq \frac{1}{2^{n+2}}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{1}{2^{n+1}} \wedge \frac{1}{2^{n+1}} \geq y \geq \frac{1}{2^{n+2}}]\!]$, but

$$Post_{Y;Z;T}([\![x > \frac{1}{2^{n+2}}]\!]) \cap [\![x = \frac{1}{2^{n+1}}]\!] \subset [\![y < \frac{1}{2^{n+2}}]\!].$$

Thus M) follows from $1(n+1)$ and L).

N). $1(n+2)$ and $2(n+2)$: immediate from L), M), $1(n+1)$, and $2(n+1)$.

O). $[\![\frac{1}{2^n} \geq z \geq \frac{1}{2^{n+1}}]\!] \not\preceq_S [\![z < \frac{1}{2^{n+1}}]\!]$:
$[\![\frac{1}{2^n} \geq z \geq \frac{1}{2^{n+1}}]\!] \stackrel{X;Y;T}{\rightarrow} [\![x \leq \frac{1}{2^{n+2}} \wedge z = \frac{1}{2^n}]\!]$, but

$$Post_{X;Y;T}([\![z < \frac{1}{2^{n+1}}]\!]) \cap [\![z = \frac{1}{2^n}]\!] \subset [\![x > \frac{1}{2^{n+2}}]\!].$$

Thus O) follows from $3(n)$ and N).

P). $3(n+1)$: immediate from K), O), and $3(n)$.

The induction is complete; $1(n)$, $2(n)$, and $3(n)$ hold for all $n \geq 1$.

We prove that the following three conditions hold for every dyadic rational $\frac{k}{2^n} \in [0,1]$:

$I(k,n)$**:** $x < \frac{k}{2^n}$, $x = \frac{k}{2^n}$, and $x > \frac{k}{2^n}$ are synchronously distinguishable,

$II(k,n)$**:** $y < \frac{k}{2^n}$, $y = \frac{k}{2^n}$, and $y > \frac{k}{2^n}$ are synchronously distinguishable,

$III(k,n)$**:** $z < \frac{k}{2^n}$, $z = \frac{k}{2^n}$, and $z > \frac{k}{2^n}$ are synchronously distinguishable.

The argument for general dyadic rationals is now straightforward by induction on the denominator. The base case is supplied by $1(1)$, $2(1)$, and $3(1)$, the synchronous distinguishability of $x = 0$, $0 < x < 1$, and $x = 1$, and the synchronous distinguishability of $z = 0$, $0 < z < 1$, and $z = 1$. Let $I(n) = \bigwedge_{k=1}^{n-1} I(k,n)$, $II(n) = \bigwedge_{k=1}^{n-1} II(k,n)$, and $III(n) = \bigwedge_{k=1}^{n-1} III(k,n)$. Assume $I(n)$, $II(n)$, and $III(n)$. If k is even, then $I(k,n+1)$, $II(k,n+1)$, and $III(k,n+1)$ follow immediately from the induction hypothesis, because in this case $\alpha(k,n+1)$ is equivalent to $\alpha(k/2,n)$ for $\alpha \in \{I, II, III\}$. So suppose k is odd.

Q). $[\![\frac{\frac{k+1}{2}}{2^n} \geq x \geq \frac{k}{2^{n+1}}]\!] \not\preceq_S [\![x < \frac{k}{2^{n+1}}]\!]$:
$[\![\frac{\frac{k+1}{2}}{2^n} \geq x \geq \frac{k}{2^{n+1}}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{\frac{k+1}{2}}{2^n} \wedge y \leq \frac{1}{2^{n+1}}]\!]$, but

$$Post_{Y;Z;T}([\![x < \frac{k}{2^{n+1}}]\!]) \cap [\![x = \frac{\frac{k+1}{2}}{2^n}]\!] \subset [\![y > \frac{1}{2^{n+1}}]\!].$$

Thus Q) follows from $I(n)$ and $2(n+1)$.

R). $[\![x \leq \frac{k}{2^{n+1}}]\!] \not\preceq_S [\![x > \frac{k}{2^{n+1}}]\!]$:
$[\![x \leq \frac{k}{2^{n+1}}]\!] \stackrel{Y;Z;T}{\rightarrow} [\![x = \frac{\frac{k+1}{2}}{2^n} \wedge y \geq \frac{1}{2^{n+1}}]\!]$, but

$$Post_{Y;Z;T}([\![x > \frac{k}{2^{n+1}}]\!]) \cap [\![x = \frac{\frac{k+1}{2}}{2^n}]\!] \subset [\![y < \frac{1}{2^{n+1}}]\!].$$

Thus R) follows from $I(n)$ and $2(n+1)$.

S). $I(n+1)$: immediate from Q), R), and $I(n)$.

T). $[\![\frac{\frac{k+1}{2}}{2^n} \geq z \geq \frac{k}{2^{n+1}}]\!] \not\preceq_S [\![z < \frac{k}{2^{n+1}}]\!]$:
$[\![\frac{\frac{k+1}{2}}{2^n} \geq z \geq \frac{k}{2^{n+1}}]\!] \stackrel{X;Y;T}{\rightarrow} [\![z = \frac{\frac{k+1}{2}}{2^n} \wedge x \leq \frac{1}{2^{n+2}}]\!]$, but

$$Post_{X;Y;T}([\![z < \frac{k}{2^{n+1}}]\!]) \cap [\![z = \frac{\frac{k+1}{2}}{2^n}]\!] \subset [\![x > \frac{1}{2^{n+2}}]\!].$$

Thus T) follows from $III(n)$ and $1(n+2)$.

U). $[\![z \leq \frac{k}{2^{n+1}}]\!] \not\preceq_S [\![z > \frac{k}{2^{n+1}}]\!]$:
$[\![z \leq \frac{k}{2^{n+1}}]\!] \stackrel{X;Y;T}{\rightarrow} [\![z = \frac{\frac{k+1}{2}}{2^n} \wedge x \geq \frac{1}{2^{n+1}}]\!]$, but

$$Post_{X;Y;T}([\![z > \frac{k}{2^{n+1}}]\!]) \cap [\![z = \frac{\frac{k+1}{2}}{2^n}]\!] \subset [\![x < \frac{1}{2^{n+1}}]\!].$$

Thus U) follows from $III(n)$ and $1(n+1)$.

V). $III(n+1)$: immediate from T), U), and $III(n)$.

The induction is complete.

Finally suppose $(x, y, z) \neq (x', y', z')$, where $(x, y, z), (x', y', z') \in [0, 1]^3$. If $x < x'$ (resp. $x > x'$), then there is a dyadic rational $\frac{k}{2^n}$ with $x < \frac{k}{2^n} < x'$ (resp. $x > \frac{k}{2^n} > x'$). By $I(k, n)$, $(x, y, z) \not\preceq_S (x', y', z')$. Similar remarks apply to $y \neq y'$ or $z \neq z'$. ■

6.2.2 Asynchronous Analysis

By Theorem 6.2.1, there is no hope of finding a finite similarity quotient on the time-abstract transition system of a rectangular automaton in three or more dimensions. In this section we define the asynchronous transition system S_A^{asyn} of a rectangular automaton A, for which simulation relations are necessarily coarser than on the time-abstract transition system S_A^{abs}. Time is treated as a silent action in the asynchronous transition system. We prove that in four or more dimensions, the simulation equivalence relation on S_A^{asyn} degenerates to equality. In the next chapter, the asynchronous transition system is applied to give a positive result.

Definition 6.2.2 Let A be a rectangular automaton, with or without silent moves. Let $\Pi_A = \{\pi_v \mid v \in V_A\}$, and define $(w, \mathbf{x}) \models_A \pi_v$ iff $v = w$. For each event $\sigma \in \Sigma$, define the binary relation $\overset{\sigma}{\rightsquigarrow}$ on Q_A by $(v, \mathbf{x}) \overset{\sigma}{\rightsquigarrow} (w, \mathbf{y})$ iff there exists a state $(v', \mathbf{x}') \in Q_A$ and a time $t \geq 0$ such that $(v, \mathbf{x}) \xrightarrow{t} (v', \mathbf{x}')$ and $(v', \mathbf{x}') \xrightarrow{\sigma} (w, \mathbf{y})$. The rectangular automaton A defines the *asynchronous* transition system

$$S_A^{asyn} =_{\mathrm{def}} (Q_A, \Sigma, \leadsto, Q_A^0, \Pi_A, \models_A). \blacksquare$$

Theorem 6.2.3 [Hen95] *There exists a two-dimensional rectangular automaton A with uniform dynamics such that the equality relation is the only bisimulation on the asynchronous transition system S_A^{asyn}.*

Proof. Similar to the proof of Theorem 6.1.1. $\blacksquare$

Since every time predecessor of a point simulates that point, equality is never the only simulation on the asynchronous transition system of a rectangular automaton. Nevertheless, asynchronous simulation equivalence coincides with equality. The proof of the next theorem is similar to the proof of Theorem 6.2.1, but it is slightly more complicated due to the lack of synchrony.

Theorem 6.2.4 *There exists a four-dimensional rectangular automaton A with uniform dynamics such that simulation equivalence on the asynchronous transition system S_A^{asyn} coincides with equality.* $\blacksquare$

Proof. The rectangular automaton A has a single vertex. Therefore we ignore the discrete part of the state, and think of a state of A as a vector in $\mathbb{R}^4$. The activity rectangle of A is $\{1\}^3 \times [1,2]$. The invariant rectangle of A is the unit cube $[0,1]^4$. For each of the finitely many triples $\kappa = (B, U, B')$, where $B, B' \subset [0,1]^4$ are integer-bounded rectangles, and $U \subset \{1,2,3,4\}$, the rectangular automaton A has an edge e_κ (from the sole vertex to itself) with $preguard(e_\kappa) = B$, $update(e_\kappa) = U$, $postguard(e_\kappa) = B'$, and $event(e_\kappa) = \kappa$. Thus the set of events coincides with this finite set of triples. Call this set $\mathcal{G}$.

Put $R = \{1\}^3 \times [1,2]$, and $S = S_A^{asyn}$. For a guarded command g representing the triple κ, Define $\overset{g}{\leadsto}$ by $\mathbf{u}\overset{g}{\leadsto}\mathbf{u}'$ iff $\mathbf{u}\overset{\kappa}{\leadsto}\mathbf{u}'$.

We refer to the four coordinates as w, x, y, and z. For example, $x = 1 \rightarrow z := 0$ denotes the guarded command $(\mathbb{R} \times \{1\} \times \mathbb{R} \times \mathbb{R}, \{4\}, \mathbb{R}^3 \times \{0\})$. Given a predicate $\phi(w, x, y, z)$, define

$$[\![\phi]\!] =_{\text{def}} \{\mathbf{u} \in [0,1]^4 \mid \phi[w := u_1; x := u_2; y := u_3; z := u_4] \text{ is true}\}.$$

If ψ is another predicate on $[0,1]^4$, and g is a guarded command, define $[\![\phi]\!]\overset{g}{\leadsto}[\![\psi]\!]$ iff for every $\mathbf{u} \in [\![\phi]\!]$, there exists a $\mathbf{u}' \in [\![\psi]\!]$ such that $\mathbf{u}\overset{g}{\leadsto}\mathbf{u}'$. Extend the definition of $\overset{\bar{g}}{\leadsto}$ to strings $\bar{g} \in \mathcal{G}^*$ in the natural way. Concatenation is indicated by a semicolon. Define

$$Post_{\bar{g}}([\![\phi]\!]) =_{\text{def}} \{\mathbf{u}' \in [0,1]^4 \mid \exists \mathbf{u} \in [\![\phi]\!].\ \mathbf{u}\overset{\bar{g}}{\leadsto}\mathbf{u}'\}$$

of asynchronous $\bar{g}$-successors of elements of $[\![\phi]\!]$. If no element of $[\![\phi]\!]$ is simulated by any element of $[\![\psi]\!]$ in the asynchronous transition system S, we write $[\![\phi]\!] \not\preceq_S [\![\psi]\!]$. We say that three predicates ϕ, ψ, and θ with free variables in $\{w, x, y, z\}$ are *asynchronously distinguishable* if $[\![\alpha]\!] \not\preceq_S [\![\beta]\!]$ for all $\alpha \neq \beta \in \{\phi, \psi, \theta\}$.

We first prove by induction that for all $n \geq 1$,

- $1(n)$: For all $a \neq b \in \{w, x, y\}$, $a < \frac{1}{2^n} \wedge b = 0$, $a = \frac{1}{2^n} \wedge b = 0$, and $a > \frac{1}{2^n} \wedge b = 0$ are asynchronously distinguishable,
- $2(n)$: For all $b \in \{w, x, y\}$, $z < \frac{1}{2^n} \wedge b = 0$, $z = \frac{1}{2^n} \wedge b = 0$, and $z > \frac{1}{2^n} \wedge b = 0$ are asynchronously distinguishable.

In fact we prove that $1(n+1) \wedge 2(n)$ implies $1(n+2) \wedge 2(n+1)$. It then follows quickly that for every dyadic rational $\frac{k}{2^n} \in [0,1]$,

- $3(k,n)$: For all $a \neq b \in \{w,x,y\}$, $a < \frac{k}{2^n} \wedge b = 0$, $a = \frac{k}{2^n} \wedge b = 0$, and $a > \frac{k}{2^n} \wedge b = 0$ are asynchronously distinguishable,

- $4(k,n)$: For all $b \in \{w,x,y\}$, $z < \frac{k}{2^n} \wedge b = 0$, $z = \frac{k}{2^n} \wedge b = 0$, and $z > \frac{k}{2^n} \wedge b = 0$ are asynchronously distinguishable.

Finally, we prove that for every dyadic rational $\frac{k}{2^n} \in [0,1]$,

- $5(k,n)$: For all $a \in \{w,x,y\}$, $[\![a < \frac{k}{2^n}]\!] \not\preceq_S [\![a \geq \frac{k}{2^n}]\!]$,

- $6(k,n)$: $[\![z < \frac{2^k}{n}]\!] \not\preceq_S [\![z \geq \frac{k}{2^n}]\!]$.

It follows from the density of the dyadic rationals that asynchronous simulation equivalence coincides with equality.

Since w, x, and y occupy symmetrical positions, we only prove results for one (usually x), and refer to this proof when a symmetrical result is needed. We write W for the guarded command $true \rightarrow w := 0$, and similarly for x, y, and z.

A). $[\![x \leq \frac{1}{2} \wedge z = 0]\!] \not\preceq_S [\![x > \frac{1}{2}]\!]$:
$[\![x \leq \frac{1}{2} \wedge z = 0]\!] \overset{W;Y;x=1 \wedge z=1}{\rightsquigarrow} [\![true]\!]$ but $Post_{W;Y;x=1 \wedge z=1}([\![x > \frac{1}{2}]\!]) = \emptyset$.

B). $[\![x \geq \frac{1}{2} \wedge y = 0]\!] \not\preceq_S [\![x < \frac{1}{2} \wedge y = 0]\!]$:
$[\![x \geq \frac{1}{2} \wedge y = 0]\!] \overset{W;Z;x=1 \rightarrow z:=0}{\rightsquigarrow} [\![y \leq \frac{1}{2} \wedge z = 0]\!]$, but

$$Post_{W;Z;x=1 \rightarrow z:=0}([\![x < \frac{1}{2} \wedge y = 0]\!]) \subset [\![y > \frac{1}{2}]\!].$$

Thus B) follows from A).

C). $[\![x \leq \frac{1}{2} \wedge y = 0]\!] \not\preceq_S [\![x > \frac{1}{2} \wedge y = 0]\!]$:
$[\![x \leq \frac{1}{2} \wedge y = 0]\!] \overset{W;Z;x=1 \rightarrow x:=0}{\rightsquigarrow} [\![y \geq \frac{1}{2} \wedge x = 0]\!]$, but

$$Post_{W;Z;x=1 \rightarrow x:=0}([\![x > \frac{1}{2} \wedge y = 0]\!]) \subset [\![y < \frac{1}{2} \wedge x = 0]\!].$$

Thus C) follows from B).

D). 1(1): immediate from B) and C).

E). $[\![1 \geq z \geq 2y \geq 0]\!] \not\preceq_S [\![0 \leq z < 2y \leq 1]\!]$:

$[\![1 \geq z \geq 2y \geq 0]\!] \overset{W;X;z=1\rightarrow x:=0}{\rightsquigarrow} [\![y \leq \frac{1}{2} \wedge x = 0]\!]$, but

$$Post_{W;X;z=1\rightarrow x:=0}([\![0 \leq z < 2y \leq 1]\!]) \subset [\![y > \frac{1}{2} \wedge x = 0]\!].$$

Thus E) follows from C).

F). $[\![z \leq \frac{1}{2} \wedge x = 0]\!] \not\preceq_S [\![z > \frac{1}{2}]\!]$:

$[\![z \leq \frac{1}{2} \wedge x = 0]\!] \overset{W;Y;z=1\rightarrow y:=0}{\rightsquigarrow} [\![x \geq \frac{1}{2} \wedge y = 0]\!]$ but

$$Post_{W;Y;z=1\rightarrow y:=0}([\![z > \frac{1}{2}]\!]) \subset [\![x < \frac{1}{2} \wedge y = 0]\!].$$

Thus F) follows from B).

G). $[\![\frac{1}{2} \geq x \geq \frac{1}{4} \wedge y = 0]\!] \not\preceq_S [\![x < \frac{1}{4} \wedge y = 0]\!]$:

$$[\![\frac{1}{2} \geq x \geq \frac{1}{4} \wedge y = 0]\!] \overset{W;Z;W}{\rightsquigarrow} [\![x = \frac{1}{2} \wedge w = 0 \wedge z \leq \frac{1}{2} \wedge 1 \geq z \geq 2y \geq 0]\!],$$

but

$Post_{W;Z;W}([\![x < \frac{1}{4} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2} \wedge w = 0 \wedge 1 \geq z \geq 2y \geq 0]\!] \subset [\![z > \frac{1}{2}]\!]$.

Thus G) follows from 1(1), E), and F).

H). $[\![x \leq \frac{1}{4} \wedge y = 0]\!] \not\preceq_S [\![x > \frac{1}{4} \wedge y = 0]\!]$:

$[\![x \leq \frac{1}{4} \wedge y = 0]\!] \overset{W;Z;W}{\rightsquigarrow} [\![x = \frac{1}{2} \wedge w = 0 \wedge y \geq \frac{1}{4}]\!]$, but

$$Post_{W;Z;W}([\![x > \frac{1}{4} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2} \wedge w = 0]\!] \subset [\![y < \frac{1}{4} \wedge w = 0]\!].$$

Hence H) follows from 1(1) and G).

I). 1(2): immediate from G), H), and 1(1).

J). $[\![z \geq \frac{1}{2} \wedge x = 0]\!] \not\preceq_S [\![z < \frac{1}{2} \wedge x = 0]\!]$:

$[\![z \geq \frac{1}{2} \wedge x = 0]\!] \overset{W;Y;z=1\rightarrow y:=0}{\rightsquigarrow} [\![x \leq \frac{1}{4} \wedge y = 0]\!]$, but

$$Post_{W;Y;z=1\rightarrow y:=0}([\![z < \frac{1}{2} \wedge x = 0]\!]) \subset [\![x > \frac{1}{4} \wedge y = 0]\!].$$

Thus J) follows from 1(2).

K). 2(1): immediate from F) and J).

This completes the base case. We now turn to the induction step. Assume that $1(n+1)$ and $2(n)$ hold.

L). $[\![z \leq \frac{1}{2^{n+1}} \wedge y = 0]\!] \not\preceq_S [\![z > \frac{1}{2^{n+1}} \wedge y = 0]\!]$:

$$[\![z \leq \frac{1}{2^{n+1}} \wedge y = 0]\!] \overset{W;X;Y}{\rightsquigarrow} [\![z = \frac{1}{2^n} \wedge x \geq \frac{1}{2^{n+1}} \wedge y = 0]\!],$$

but

$$Post_{W;X;Y}([\![z > \frac{1}{2^{n+1}} \wedge y = 0]\!]) \cap [\![z = \frac{1}{2^n} \wedge y = 0]\!] \subset [\![x < \frac{1}{2^{n+1}} \wedge y = 0]\!].$$

Thus L) follows from $1(n+1)$ and $2(n)$.

M). $[\![\frac{1}{2^{n+1}} \geq x \geq \frac{1}{2^{n+2}} \wedge y = 0]\!] \not\preceq_S [\![x < \frac{1}{2^{n+2}} \wedge y = 0]\!]$:

$[\![\frac{1}{2^{n+1}} \geq x \geq \frac{1}{2^{n+2}} \wedge y = 0]\!] \overset{W;Z;W}{\rightsquigarrow}$

$$[\![x = \frac{1}{2^{n+1}} \wedge 1 \geq z \geq 2y \geq 0 \wedge z \leq \frac{1}{2^{n+1}} \wedge w = 0]\!],$$

but $Post_{W;Z;W}([\![x < \frac{1}{2^{n+2}} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2^{n+1}} \wedge w = 0 \wedge 1 \geq z \geq 2y \geq 0]\!]$ is a subset of $[\![z > \frac{1}{2^{n+1}} \wedge w = 0]\!]$. Thus M) follows from $1(n+1)$, E), and L).

N). $[\![x \le \frac{1}{2^{n+2}} \wedge y = 0]\!] \not\preceq_S [\![x > \frac{1}{2^{n+2}} \wedge y = 0]\!]$:

$$[\![x \le \frac{1}{2^{n+2}} \wedge y = 0]\!] \overset{W;Z;W}{\leadsto} [\![x = \frac{1}{2^{n+1}} \wedge \frac{1}{2^{n+1}} \ge y \ge \frac{1}{2^{n+2}} \wedge w = 0]\!],$$

but $Post_{W;Z;W}([\![x > \frac{1}{2^{n+2}} \wedge y = 0]\!]) \cap [\![x = \frac{1}{2^{n+1}} \wedge w = 0]\!] \subset [\![y < \frac{1}{2^{n+2}} \wedge w = 0]\!]$. Thus N) follows from $1(n+1)$ and M).

O). $1(n+2)$: immediate from M), N), and $1(n+1)$.

P). $[\![\frac{1}{2^n} \ge z \ge \frac{1}{2^{n+1}} \wedge y = 0]\!] \not\preceq_S [\![z < \frac{1}{2^{n+1}} \wedge y = 0]\!]$:

$$[\![\frac{1}{2^n} \ge z \ge \frac{1}{2^{n+1}} \wedge y = 0]\!] \overset{W;X;W}{\leadsto} [\![z = \frac{1}{2^n} \wedge y \le \frac{1}{2^{n+2}} \wedge w = 0]\!],$$

but $Post_{W;X;W}([\![z < \frac{1}{2^{n+1}} \wedge y = 0]\!]) \cap [\![z = \frac{1}{2^n} \wedge w = 0]\!] \subset [\![y > \frac{1}{2^{n+2}} \wedge w = 0]\!]$. Thus P) follows from $2(n)$ and $1(n+2)$.

Q). $2(n+1)$: immediate from L), P), and $2(n)$.

The proof of $1(n)$ and $2(n)$ for all n is complete. The truth of $3(k,n)$ and $4(k,n)$ follows by the same argument used to prove $I(k,n)$, $II(k,n)$, and $III(k,n)$ in the proof of Theorem 6.2.1, except that the case of $\frac{k}{2^n} = 1$ must be handled differently.

R). $[\![a = 1 \wedge b = 0]\!] \not\preceq_S [\![a < 1 \wedge b = 0]\!]$:
$[\![a = 1 \wedge b = 0]\!] \overset{a=1 \wedge b=0}{\leadsto} [\![true]\!]$, but $Post_{a=1 \wedge b=0}([\![a < 1 \wedge b = 0]\!]) = \emptyset$.

S). $[\![a < 1 \wedge b = 0]\!] \not\preceq_S [\![a = 1 \wedge b = 0]\!]$: Given $\mathbf{x} \in [\![a < 1 \wedge b = 0]\!]$, let n be large enough so that $a < 1 - \frac{1}{2^n}$. Then $\mathbf{x} \overset{c:=0;a=1\rightarrow b=0}{\leadsto} [\![c \ge \frac{1}{2^n} \wedge b = 0]\!]$ but

$$Post_{c:=0;a=1\rightarrow b=0}(a = 1 \wedge b = 0) \cap [\![c \ge \frac{1}{2^n} \wedge b = 0]\!] = \emptyset.$$

Thus $3(2^n, n)$ and $4(2^n, n)$ follow from $1(n)$ and $2(n)$ respectively.

We omit the inductive step of the proof of $3(k,n)$ and $4(k,n)$, and turn to $5(k,n)$ and $6(k,n)$.

T). $[\![x < \frac{k}{2^n}]\!] \not\preceq_S [\![x \geq \frac{k}{2^n}]\!]$:
$[\![x < \frac{k}{2^n}]\!] \overset{Y}{\rightsquigarrow} [\![x < \frac{k}{2^n} \wedge y = 0]\!]$, but $Post_Y(x \geq \frac{k}{2^n}) \subset [\![x \geq \frac{k}{2^n} \wedge y = 0]\!]$. Thus $5(k,n)$ follows from $3(k,n)$.

U). $[\![z < \frac{k}{2^n}]\!] \not\preceq_S [\![z \geq \frac{k}{2^n}]\!]$:
$[\![z < \frac{k}{2^n}]\!] \overset{Y}{\rightsquigarrow} [\![z < \frac{k}{2^n} \wedge y = 0]\!]$, but $Post_Y(z \geq \frac{k}{2^n}) \subset [\![z \geq \frac{k}{2^n} \wedge y = 0]\!]$. Thus $6(k,n)$ follows from $4(k,n)$. ■

Chapter 7

Language Equivalence

In this chapter we consider the following question:

> *Let $\mathcal{C}$ be the class of all n-dimensional closed integral rectangular automata with uniform activity rectangle R. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, under what conditions is it the case that for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $L(\{(v, \mathbf{x})\}, S_A^{asyn}) = L(\{(v, \mathbf{y})\}, S_A^{asyn})$?*

This is the finitary discrete-language equivalence problem. In two dimensions, the results of Section 6.1 may be applied to obtain an exact characterization of language equivalence. For three or more dimensions, we derive sufficient conditions for language equivalence in Section 7.3, and necessary conditions in Section 7.4; We do not obtain an exact characterization, although if the activity rectangle is a cube, then the necessary conditions and sufficient conditions are quite close.

An easy sufficient condition may be derived from the translation from A into N_A (see Section 3.2). Since this translation is language-preserving, it follows that $\mathbf{x}$

and $\mathbf{y}$ are language equivalent with respect to the class $\mathcal{C}$ above iff $(\mathbf{x}, \mathbf{x})$ and $(\mathbf{y}, \mathbf{y})$ are language equivalent with respect to the class of multirate automata with attractors. The latter can be decided using the proof of Theorem 3.1.6.

We derive improved sufficient conditions by observing that the multirate automaton N_A is *one-sided*, meaning that each continuous variable is either exclusively bounded from above, or exclusively bounded from below, by preguards. The one-sidedness of N_A is inherited by the timed automaton T_{N_A}. We study the class OTA of one-sided timed automata, and find that language equivalence with respect to OTA is coarser than language equivalence with respect to the entire class of timed automata.

7.1 Definitions

We use the following convention:

Convention 7.1.1 In this chapter, all rectangles and all rectangular automata are assumed to be integral. ■

In particular, our convention applies to the activity, invariant, guard, and preguard rectangles defining a rectangular automaton.

Definition 7.1.2 Let $\mathcal{C}$ be a class of n-dimensional closed rectangular automata with uniform activity rectangle R. For $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$, we write $\mathbf{x} \preceq^{\mathcal{C}}_{\mathrm{lang}} \mathbf{u}$ if for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $L((v, \mathbf{x}), S_A^{asyn}) \subset L((v, \mathbf{u}), S_A^{asyn})$. We write $\mathbf{x} \approx^{\mathcal{C}}_{\mathrm{lang}} \mathbf{u}$ if $\mathbf{x} \preceq^{\mathcal{C}}_{\mathrm{lang}} \mathbf{u}$ and $\mathbf{u} \preceq^{\mathcal{C}}_{\mathrm{lang}} \mathbf{x}$. This equivalence relation is known as *language equivalence* with respect to $\mathcal{C}$.

Definition 7.1.3 Let $\mathcal{C}$ be a class of n-dimensional closed rectangular automata with uniform activity rectangle R. For $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$, we say that $\mathbf{u}$ *synchronously simulates* $\mathbf{x}$, and we write $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$, if for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $(v, \mathbf{x}) \preceq_{S^{abs}_A} (v, \mathbf{u})$. We write $\mathbf{x} \approx^{\mathcal{C}}_{\text{syn}} \mathbf{u}$ if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{u}$ and $\mathbf{u} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{x}$. This equivalence relation is known as *synchronous simulation equivalence* with respect to $\mathcal{C}$.

Similarly, we say that $\mathbf{u}$ *asynchronously simulates* $\mathbf{x}$, and we write $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$, if for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $(v, \mathbf{x}) \preceq_{S^{asyn}_A} (v, \mathbf{u})$. We write $\mathbf{x} \approx^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$ if $\mathbf{x} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$ and $\mathbf{u} \preceq^{\mathcal{C}}_{\text{asyn}} \mathbf{x}$. This equivalence relation is known as *asynchronous simulation equivalence* with respect to $\mathcal{C}$. ■

Definition 7.1.4 Let $\mathcal{C}$ be a class of n-dimensional closed rectangular automata with uniform activity rectangle R. For $\mathbf{x}, \mathbf{u} \in \mathbb{R}^n$, we say that $\mathbf{u}$ and $\mathbf{x}$ are *synchronously bisimilar*, and we write $\mathbf{x} \cong^{\mathcal{C}}_{\text{syn}} \mathbf{u}$, if for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $(v, \mathbf{x}) \equiv^{bis}_{S^{abs}_A} (v, \mathbf{u})$. This equivalence relation is known as *synchronous bisimulation equivalence* with respect to $\mathcal{C}$. We say that $\mathbf{u}$ and $\mathbf{x}$ are *asynchronously bisimilar*, and we write $\mathbf{x} \cong^{\mathcal{C}}_{\text{asyn}} \mathbf{u}$, if for every automaton $A \in \mathcal{C}$, and every vertex $v \in V_A$, $(v, \mathbf{x}) \equiv^{bis}_{S^{asyn}_A} (v, \mathbf{u})$. This equivalence relation is known as *asynchronous bisimulation equivalence* with respect to $\mathcal{C}$. ■

7.2 One-Sided Timed Automata

Definition 7.2.1 Define the *region equivalence relation* $\equiv^{reg}$ on $\mathbb{R}^n$ by $\mathbf{x} \equiv^{reg} \mathbf{u}$ iff $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{u} \rfloor$, $\lceil \mathbf{x} \rceil = \lceil \mathbf{u} \rceil$, and for all $1 \leq i, j \leq n$, $\lfloor x_j - x_i \rfloor = \lfloor u_j - u_i \rfloor$. ■

Theorem 7.2.2 [AD94] *Language equivalence with respect to the class of all integral timed automata with attractors coincides with the region equivalence relation.* ■

Definition 7.2.3 Let T be an n-dimensional timed automaton with attractors. We say that coordinate i is an *upper-bounded clock* if for each edge $e \in E_T$ and each vertex $v \in V_T$, $preguard(e)_i$ and $inv(v)_i$ (which are intervals) are unbounded from below. This means that the ith coordinate of the continuous state of T is constrained by guards and invariants only from above. We say that coordinate j is a *lower-bounded clock* if for each edge $e \in E_T$, and each vertex $v \in V_T$, $preguard(e)_j$ and $inv(v)_j$ are unbounded from above. This means that the jth coordinate of the continuous state of T is constrained by guards and invariants only from below. The timed automaton T is *upper-bounded* (resp. *lower-bounded*) if each coordinate is an upper-bounded (resp. lower-bounded) clock. The timed automaton T is *one-sided* if every coordinate is either an upper-bounded clock or a lower-bounded clock. ■

Theorem 7.2.4 *Let $R \subset \mathbb{R}^n$ be a compact rectangle. For every vector $\mathbf{z} \in \mathbb{R}^n$, there exist vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ such that for every closed rectangular automaton A with uniform activity rectangle R, there exists a $2n$-dimensional one-sided timed automaton T_A with attractors such that for every vertex v, $L((v, \mathbf{z}), S_A^{asyn}) = L((v, (\mathbf{x}, \mathbf{y})), S_{T_A}^{asyn})$.*

Proof. The construction of N_A from A as given in Definition 3.2.2 requires that A be compact, but this requirement was made for ease of exposition. When the activity rectangle R of A is compact, it is easy to extend the construction to closed A.

The theorem then follows from Theorem 3.2.9 and the proof of Corollary 3.1.7. Let $R = \prod_{i=1}^{n}[c_i, d_i]$, and let λ be the least common multiple of $\{c_1, d_1, \ldots, c_n, d_n\}$. Then $\mathbf{x} = (z_1\frac{\lambda}{c_1}, \ldots, z_n\frac{\lambda}{c_n})$ and $\mathbf{y} = (z_1\frac{\lambda}{d_1}, \ldots, z_n\frac{\lambda}{d_n})$. ∎

Definition 7.2.5 A *guarded command* is a triple $g = (B, U, B')$, where $B, B' \subset \mathbb{R}^n$ are closed rectangles, and $U \subset \{1, \ldots, n\}$. The guarded command g defines the binary relation $jump_g$ on $\mathbb{R}^n$ by $(\mathbf{x}, \mathbf{x}') \in jump_g$ iff (1) $\mathbf{x} \in B$, (2) for each $i \in U$, $x'_i \in B'_i$, and (3) for each $i \notin U$, $x'_i = closest(x_i, B'_i)$. The guarded command g defines the binary relation $delayed_g$ on $\mathbb{R}^n$ by $(\mathbf{x}, \mathbf{x}') \in delayed_g$ iff there exists an $\mathbf{x}'' \in \mathbb{R}^n$ such that (1) $(\mathbf{x}'', \mathbf{x}') \in jump_g$ and (2) there exists a time $t \in \mathbb{R}_{\geq 0}$ such that $\mathbf{x}'' = \mathbf{x} + t\mathbf{1}$. ∎

A guarded command (B, U, B') should be interpreted as an edge e in a timed automaton with attractors, where $preguard(e) = B$, $postguard(e) = B'$, and $update(e) = U$. Then $(\mathbf{x}, \mathbf{x}') \in jump_g$ iff $(v, \mathbf{x}) \xrightarrow{e} (w, \mathbf{x}')$, where $e = (v, w)$. The $delayed_g$ relation uses the asynchronous transition relation: $(\mathbf{x}, \mathbf{x}') \in delayed_g$ iff $(v, \mathbf{x}) \overset{e}{\rightsquigarrow} (w, \mathbf{x}')$. Thus simulation with respect to a class $\mathcal{C}$ of timed automata with attractors is really just simulation with respect to the infinite set of events consisting of all guarded commands appearing in $\mathcal{C}$.

Definition 7.2.6 A guarded command (B, U, B') *appears* in the timed automaton with attractors T if there exists an edge $e \in E_T$ with $preguard_T(e) = B$, $postguard_T(e) = B'$, and $update_T(e) = U$. ∎

We begin the analysis of one-sided timed automata by first considering automata whose clocks are all of the same type. For a vector $\mathbf{x} \in \mathbb{R}^n$ and a time

$t \geq 0$, define $\mathbf{x} + t$ to be the vector $(x_1 + t, \ldots, x_n + t)$.

7.2.1 Upper-Bounded Timed Automata

For the class of upper-bounded timed automata, language equivalence and both types of simulation equivalence coincide, while both types of bisimulation equivalence are much finer. The latter two coincide with region equivalence.

Proposition 7.2.7 *Let $\mathcal{C}$ be the class of upper-bounded closed integral timed automata with attractors. For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, the following four conditions are equivalent:*

- $\mathbf{x} \approx^{\mathcal{C}}_{\mathrm{lang}} \mathbf{x}'$ *(i.e., clock vectors $\mathbf{x}$ and $\mathbf{x}'$ are language equivalent with respect to $\mathcal{C}$),*
- $\mathbf{x} \approx^{\mathcal{C}}_{\mathrm{syn}} \mathbf{x}'$ *(i.e., clock vectors $\mathbf{x}$ and $\mathbf{x}'$ are synchronously simulation equivalent with respect to $\mathcal{C}$),*
- $\mathbf{x} \approx^{\mathcal{C}}_{\mathrm{asyn}} \mathbf{x}'$ *(i.e., clock vectors $\mathbf{x}$ and $\mathbf{x}'$ are asynchronously simulation equivalent with respect to $\mathcal{C}$),*
- $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$.

In two or more dimensions, synchronous and asynchronous bisimulation equivalence with respect to $\mathcal{C}$ coincide with region equivalence.

Proof. Call a guarded command *upper-bounded* if it appears in some $T \in \mathcal{C}$. Every guard B of an upper-bounded guarded command is of the form $\prod_{i=1}^{n}(-\infty, q_i]$, where each $q_i \in \mathbb{Z} \cup \{\infty\}$. It follows that $\mathbf{x} \in B$ iff $\mathbf{x} \leq \mathbf{q}$ coordinatewise. Thus for any upper-bounded guarded command $g = (B, U, B')$, the domain of $jump_g$ is $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \leq \mathbf{q}\}$. Put $\mathbf{x} \preceq \mathbf{x}'$ iff $\lceil \mathbf{x}' \rceil \leq \lceil \mathbf{x} \rceil$. Then $\mathbf{x} \preceq \mathbf{x}'$ iff for every upper-bounded guarded command g, $\mathbf{x} \in \mathrm{dom}\ delayed_g$ implies $\mathbf{x}' \in \mathrm{dom}\ delayed_g$. Thus $\mathbf{x} \preceq^{\mathcal{C}}_{\mathrm{lang}} \mathbf{x}'$ implies $\mathbf{x} \preceq \mathbf{x}'$. On the other hand, $\preceq$ is a synchronous simulation. For suppose $\mathbf{x} \preceq \mathbf{u}$. If $(\mathbf{x}, \mathbf{x}') \in jump_g$, let $\mathbf{u}'$ be the unique vector such that $(\mathbf{u}, \mathbf{u}') \in jump_g$ and for each $i \in U$, $u'_i = x'_i$. Then $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$. Time actions of duration 0 preserve the simulation: for every $t \geq 0$, $\lceil \mathbf{u} + 0 \rceil \leq \lceil \mathbf{x} + t \rceil$. This proves the equivalence of the four conditions in the statement of the proposition.

Since region equivalence is a synchronous bisimulation, it suffices to show that asynchronous bisimulation equivalence coincides with region equivalence. Suppose $\mathbf{x} \cong^{\mathcal{C}}_{\mathrm{asyn}} \mathbf{u}$. Since $\mathbf{x} \approx^{\mathcal{C}}_{\mathrm{asyn}} \mathbf{u}$, it follows from the above that $\lceil \mathbf{x} \rceil = \lceil \mathbf{u} \rceil$. Suppose that $\lfloor \mathbf{x} \rfloor \neq \lfloor \mathbf{u} \rfloor$. Without loss of generality, suppose $x_i < u_i$ and $u_i \in \mathbb{Z}$. Consider the upper-bounded guarded command g traditionally denoted by the expression $z_i \leq u_i \rightarrow z_j := 0$, where z_i refers to the ith clock variable, and $j \neq i$. There is a unique $\mathbf{u}'$ such that $(\mathbf{u}, \mathbf{u}') \in delayed_g$. Let $\mathbf{x}' = \mathbf{x}[x_j := 0]$. Then $(\mathbf{x}, \mathbf{x}') \in delayed_g$. We claim $\mathbf{x}' \not\cong^{\mathcal{C}}_{\mathrm{asyn}} \mathbf{u}'$. This is because for small enough $t > 0$, $\lceil x'_i + t \rceil = \lceil x_i + t \rceil = u_i$ and $\lceil x'_j + t \rceil = 1$; but for every $t' > 0$, $\lceil u'_i + t' \rceil > u_i$, and for $t' = 0$, $\lceil u'_j + t' \rceil = 0$. Thus if g' is the trivial guarded command $(\mathbb{R}^n, \emptyset, \mathbb{R}^n)$, then $(\mathbf{x}', \mathbf{x}' + t) \in delayed_{g'}$, but no $delayed_{g'}$-successor of $\mathbf{u}'$ is asynchronously bisimulation equivalent to $\mathbf{x}' + t$. We have proven by contradiction that $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{u} \rfloor$. To complete the proof, we show that the fractional parts of the coordinates of $\mathbf{x}$ and $\mathbf{y}$ must have the same relative order. Suppose $\lfloor x_j - x_i \rfloor > \lfloor u_j - u_i \rfloor$. Then because $\lceil \mathbf{x} \rceil = \lceil \mathbf{u} \rceil$ and $\lfloor \mathbf{x} \rfloor = \lfloor \mathbf{u} \rfloor$, the fractional part of x_j is greater than the fractional part of x_i, while the reverse holds for u_j and u_i. Thus there exists a time $t \geq 0$ (namely, $\lceil x_j \rceil - x_j + \epsilon$ for small enough ϵ) such that for every time $t' \geq 0$, $\lceil \mathbf{x} + t \rceil \neq \lceil \mathbf{u} + t' \rceil$. Thus again there is a $delayed_{g'}$-successor of $\mathbf{x}$ that is not asynchronously bisimulation equivalent to any $delayed_{g'}$-successor of $\mathbf{u}$. ■

7.2.2 Lower-Bounded Timed Automata

For the class of lower-bounded timed automata with attractors, the asynchronous notions of equivalence collapse to the universal relation, while the synchronous equivalences are much finer.

Proposition 7.2.8 *Let $\mathcal{C}$ be the class of lower-bounded closed integral timed automata with attractors.*

1. *Language equivalence, asynchronous simulation equivalence, and asynchronous bisimulation equivalence with respect to $\mathcal{C}$ coincide with the universal relation $\mathbb{R}^n \times \mathbb{R}^n$.*
2. *Clock vectors $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^n$ are synchronously simulation equivalent with respect to $\mathcal{C}$ iff $\lfloor \mathbf{y} \rfloor = \lfloor \mathbf{y}' \rfloor$.*
3. *In two or more dimensions, synchronous bisimulation equivalence with respect to $\mathcal{C}$ coincides with region equivalence.*

Proof. Call a guarded command *lower-bounded* if it appears in some $T \in \mathcal{C}$. Every guard B of a lower-bounded guarded command is of the form $\prod_{j=1}^{n} [p_j, \infty)$, where each $p_j \in \mathbb{Z} \cup \{-\infty\}$. It follows that $\mathbf{y} \in B$ iff $\mathbf{y} \geq \mathbf{p}$ coordinatewise. Thus for any lower-bounded guarded command $g = (B, U, B')$, the domain of $jump_g$ is $\{\mathbf{y} \in \mathbb{R}^n \mid \mathbf{y} \geq \mathbf{p}\}$. But since for any $\mathbf{y}$ there is a $t \geq 0$ such that $\mathbf{y} + t \geq \mathbf{p}$, the domain of $delayed_g$ is $\mathbb{R}^n$. This proves (1).

Let $\mathcal{C}$ be the class of all lower bounded timed automata. To prove (2), put $\mathbf{y} \preceq \mathbf{y}'$ iff $\lfloor \mathbf{y} \rfloor \leq \lfloor \mathbf{y}' \rfloor$. Then $\mathbf{y} \preceq \mathbf{y}'$ iff for every lower bound guarded command g, $\mathbf{y} \in \text{dom } jump_g$ implies $\mathbf{y}' \in \text{dom } jump_g$. Thus $\mathbf{y} \preceq^{\mathcal{C}}_{\text{syn}} \mathbf{y}'$ implies $\mathbf{y} \preceq \mathbf{y}'$. On the other hand, $\preceq$ is a synchronous simulation. For if $\mathbf{y} \preceq \mathbf{v}$, then (a) if $(\mathbf{y}, \mathbf{y}') \in jump_g$, then $\lfloor \mathbf{y}' \rfloor \leq \lfloor \mathbf{v}' \rfloor$, where $\mathbf{v}'$ is the unique vector such that $(\mathbf{v}, \mathbf{v}') \in jump_g$ and $v'_i = y'_i$ for $i \in U$, and (b) for every time $t \geq 0$, there exists a time $t' \geq 0$ such that $\lfloor \mathbf{y} + t \rfloor \leq \lfloor \mathbf{v} + t' \rfloor$. This proves (2). The proof of (3) is almost identical to the proof of the last assertion of Proposition 7.2.7. ∎

7.2.3 One-Sided Timed Automata

Let T be an $(n+m)$-dimensional one-sided closed integral timed automaton with attractors, and with n upper-bounded clocks and m lower-bounded clocks. After suitable rearrangement, we can assume that coordinates 1 through n are upper-bounded clocks and that coordinates $n+1$ through $n+m$ are lower-bounded clocks. Then for each guarded command $guard_T(e) = (B, U, B')$, it follows that the guard B is of the form

$$\prod_{i=1}^{n} (-\infty, q_i] \times \prod_{j=1}^{m} [p_j, \infty),$$

where each $q_i \in \mathbb{Z} \cup \{\infty\}$ and each $p_j \in \mathbb{Z} \cup \{-\infty\}$. So we represent preguards B of one-sided timed automata by pairs of vectors $(\mathbf{q}, \mathbf{p}) \in (\mathbb{Z} \cup \{\infty\})^n \times (\mathbb{Z} \cup \{-\infty\})^m$. We call such preguards and the guarded commands they comprise *one-sided*. We write a vector $\mathbf{u} \in \mathbb{R}^{n+m}$ as the pair $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (u_1, \ldots, u_n) \in \mathbb{R}^n$ and $\mathbf{y} = (u_{n+1}, \ldots, u_{n+m}) \in \mathbb{R}^m$. We use the notational convention that i ranges over $\{1, \ldots, n\}$ and j ranges over $\{1, \ldots, m\}$.

Definition 7.2.9 The binary relation $\preceq_1$ on $\mathbb{R}^{n+m}$ is defined by $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$ iff $\lceil \mathbf{x}' \rceil \leq \lceil \mathbf{x} \rceil$ and $\forall i. \forall j. \lfloor y'_j - x'_i \rfloor \geq \lfloor y_j - x_i \rfloor$. ■

Put $(\mathbf{x}, \mathbf{y}) \approx_1 (\mathbf{x}', \mathbf{y}')$ iff $(\mathbf{x}, \mathbf{y}) \preceq_1 (\mathbf{x}', \mathbf{y}')$ and $(\mathbf{x}', \mathbf{y}') \preceq_1 (\mathbf{x}, \mathbf{y})$. The difference between region equivalence and $\approx_1$-equivalence is that the latter only considers the relative order of fractional parts of upper-bounded clocks with respect to the fractional parts of lower-bounded clocks. The relative order of the fractional parts of two different lower-bounded clocks (or two different upper-bounded clocks) is irrelevant to $\approx_1$-equivalence.

Lemma 7.2.10 *For every one-sided guarded command $g = ((\mathbf{q}, \mathbf{p}), U, B')$,*

$$\text{dom } \mathit{delayed}_g = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \textit{ and } (\forall i)(\forall j)[p_j - q_i \leq y_j - x_i]\}.$$

Proof.

$$\begin{aligned}
\text{dom } \mathit{delayed}_g &= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid (\exists t \geq 0)[(\mathbf{x}, \mathbf{y}) + t \in \text{dom } \mathit{jump}_g]\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid (\exists t \geq 0)[(\mathbf{x} + t) \leq \mathbf{q} \text{ and } (\mathbf{y} + t) \geq \mathbf{p}]\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } \max_j p_j - y_j \leq \min_i q_i - x_i\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } (\forall i)(\forall j)[p_j - y_j \leq q_i - x_i]\} \\
&= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+m} \mid \mathbf{x} \leq \mathbf{q} \text{ and } (\forall i)(\forall j)[p_j - q_i \leq y_j - x_i]\}. \blacksquare
\end{aligned}$$

Lemma 7.2.11 *For all* $(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\in\mathbb{R}^{n+m}$, $(\mathbf{x},\mathbf{y})\preceq_1(\mathbf{x}',\mathbf{y}')$ *iff for every one-sided guarded command* $g=(B,U,B')$, $(\mathbf{x},\mathbf{y})\in \mathrm{dom}\ \mathit{delayed}_g$ *implies* $(\mathbf{x}',\mathbf{y}')\in \mathrm{dom}\ \mathit{delayed}_g$.

Proof. Suppose $(\mathbf{x},\mathbf{y})\preceq_1(\mathbf{x}',\mathbf{y}')$. Let $g=((\mathbf{q},\mathbf{p}),\rho)$. Since $(\mathbf{x},\mathbf{y})\preceq_1(\mathbf{x}',\mathbf{y}')$, $\lceil\mathbf{x}'\rceil\leq\lceil\mathbf{x}\rceil\leq\mathbf{q}$, and for each i and j, $y'_j-x'_i\geq\lfloor y_j-x_i\rfloor\geq p_j-q_i$. The *only if* now follows from Lemma 7.2.10.

We now prove the *if*. Case 1: $\lceil\mathbf{x}'\rceil\not\leq\lceil\mathbf{x}\rceil$. Let B be the one-sided guard $(\lceil\mathbf{x}\rceil,(-\infty,\ldots,-\infty))$, and let $g=(B,\emptyset,B)$. Then $(\mathbf{x},\mathbf{y})\in \mathrm{dom}\ \mathit{delayed}_g$ and $(\mathbf{x}',\mathbf{y}')\notin \mathrm{dom}\ \mathit{delayed}_g$. Case 2: $\lfloor y'_j-x'_i\rfloor<\lfloor y_j-x_i\rfloor$. Let $q_i,p_j\in\mathbb{Z}$ be such that $p_j-q_i=\lfloor y_j-x_i\rfloor$ and $x_i\leq q_i$. Let B be the one-sided guard $((\infty,\ldots,\infty,q_i,\infty,\ldots,\infty)(-\infty,\ldots,-\infty,p_j,-\infty,\ldots,-\infty))$, and let $g=(B,\emptyset,B)$. By Lemma 7.2.10, $(\mathbf{x},\mathbf{y})\in \mathrm{dom}\ \mathit{delayed}_g$ and $(\mathbf{x}',\mathbf{y}')\notin \mathrm{dom}\ \mathit{delayed}_g$. ∎

Synchronous Analysis.

It is not the case that $(\mathbf{x},\mathbf{y})\preceq_1(\mathbf{x}',\mathbf{y}')$ implies that for every one-sided guarded command g, $(\mathbf{x},\mathbf{y})\in \mathrm{dom}\ \mathit{jump}_g$ implies $(\mathbf{x}',\mathbf{y}')\in \mathrm{dom}\ \mathit{jump}_g$. For this we need the additional condition that $\lfloor\mathbf{y}'\rfloor\geq\lfloor\mathbf{y}\rfloor$.

Definition 7.2.12 The binary relation $\preceq_2$ on $\mathbb{R}^{n+m}$ is defined by $(\mathbf{x},\mathbf{y})\preceq_2(\mathbf{x}',\mathbf{y}')$ iff $(\mathbf{x},\mathbf{y})\preceq_1(\mathbf{x}',\mathbf{y}')$ and $\lfloor\mathbf{y}'\rfloor\geq\lfloor\mathbf{y}\rfloor$. ∎

In order to prove that $\preceq_2$ is a synchronous simulation, we need a few basic lemmas about differences of floors and ceilings.

Lemma 7.2.13 *For all* $a,b\in\mathbb{R}$, $\lfloor a\rfloor-\lfloor b\rfloor\geq\lfloor a-b\rfloor$. ∎

Lemma 7.2.14 *For all $a, b \in \mathbb{R}$, $\lceil a \rceil - \lceil b \rceil \geq \lfloor a - b \rfloor$.* ■

Lemma 7.2.15 *For all $a, b \in \mathbb{R}$, $\lfloor a - b \rfloor \geq \lfloor a \rfloor - \lceil b \rceil$.* ■

Proposition 7.2.16 *The relation $\preceq_2$ is the largest synchronous simulation on $\mathbb{R}^{n+m}$ with respect to the class of one-sided closed integral timed automata with attractors.*

Proof. Let $\mathcal{C}$ be the class of one-sided closed integral timed automata with attractors. From the proofs of Propositions 7.2.7 and 7.2.8 we know that $(\mathbf{x}, \mathbf{y}) \preceq^{\mathcal{C}}_{\text{syn}} (\mathbf{u}, \mathbf{v})$ implies both $\lceil \mathbf{u} \rceil \leq \lceil \mathbf{x} \rceil$ and $\lfloor \mathbf{v} \rfloor \geq \lfloor \mathbf{y} \rfloor$. That $(\mathbf{x}, \mathbf{y}) \preceq^{\mathcal{C}}_{\text{syn}} (\mathbf{u}, \mathbf{v})$ implies $\forall i. \forall j. \lfloor v_j - u_i \rfloor \geq \lfloor y_j - u_i \rfloor$ follows from Lemma 7.2.11. For if $\lfloor v_j - u_i \rfloor < \lfloor y_j - u_i \rfloor$, then there exists a guarded command g such that $(\mathbf{x}, \mathbf{y}) \in \text{dom } \mathit{delayed}_g$ and $(\mathbf{u}, \mathbf{v}) \notin \text{dom } \mathit{delayed}_g$. Thus there exists a time $t \geq 0$ such that for all $t' \geq 0$, $(\mathbf{x} + t, \mathbf{y} + t) \not\preceq^{\mathcal{C}}_{\text{syn}} (\mathbf{u} + t', \mathbf{v} + t')$. We have proven that $\preceq^{\mathcal{C}}_{\text{syn}} \subset \preceq_2$, so if $\preceq_2$ is a synchronous simulation, then it is the largest.

We now show that $\preceq_2$ is a synchronous simulation. Suppose $(\mathbf{x}, \mathbf{y}) \preceq_2 (\mathbf{u}, \mathbf{v})$. Analysis of time transitions is simple. For $t \in \mathbb{R}_{\geq 0}$, let B be the one-sided guard $(\lceil \mathbf{x} + t \rceil, \lfloor \mathbf{y} + t \rfloor)$, and let $g = (B, \emptyset, B)$. Since $(\mathbf{x}, \mathbf{y}) \in \text{dom } \mathit{delayed}_g$, it follows from Lemma 7.2.11 that $(\mathbf{u}, \mathbf{v}) \in \text{dom } \mathit{delayed}_g$ as well. Thus there exists a $t' \in \mathbb{R}_{\geq 0}$ such that $\mathbf{u} + t' \leq \lceil \mathbf{x} + t \rceil$ and $\mathbf{v} + t' \geq \lfloor \mathbf{y} + t \rfloor$. For the final requirement, $\lfloor (v_j + t') - (u_i + t') \rfloor = \lfloor v_j - u_i \rfloor \geq \lfloor y_j - x_i \rfloor = \lfloor (y_j + t) - (x_i + t) \rfloor$. Therefore $(\mathbf{x} + t, \mathbf{y} + t) \preceq_2 (\mathbf{u} + t', \mathbf{v} + t')$.

Analysis of guarded command transitions is more difficult. Let $g = (B, U, B')$ be a one-sided guarded command, and suppose $((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) \in \mathit{jump}_g$. Let

$(\mathbf{u}', \mathbf{v}')$ be the unique vector such that (1) $((\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}')) \in \mathit{jump}_g$, (2) for each $i \in U$, $u'_i = \lfloor x'_i \rfloor$, and (3) for each $j \in U$, $v'_j = \lceil y'_j \rceil$. We will show that $(\mathbf{x}', \mathbf{y}') \preceq_2 (\mathbf{u}', \mathbf{v}')$.

First we show that $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$. For $i \in U$, $\lceil u'_i \rceil \leq \lceil x'_i \rceil$ because $u'_i = \lfloor x'_i \rfloor$. For $i \notin U$, $u'_i = \mathit{closest}(u_i, B'_i)$ and $x'_i = \mathit{closest}(x_i, B'_i)$. If $u_i \leq x_i$ it follows that $u'_i \leq x'_i$, and therefore that $\lceil u'_i \rceil \leq \lceil x'_i \rceil$. So suppose $u_i > x_i$. Since $\mathbf{u} \leq \lceil \mathbf{x} \rceil$, we have $\lceil x_i \rceil \geq u_i > x_i$. Case 1: $x_i \in B'_i$. Since $x_i \notin \mathbb{Z}$ it follows that $\lceil x_i \rceil \in B'_i$, and therefore that $u_i \in B'_i$. Therefore $x'_i = x_i$ and $u'_i = u_i$, and so $\lceil u'_i \rceil \leq \lceil x'_i \rceil$. Case 2: $x_i \notin B'_i$, $u_i \in B'_i$. It follows that $u'_i = u_i = \lceil x_i \rceil = \min B'_i = x'_i$, so $\lceil u'_i \rceil = \lceil x'_i \rceil$. Case 3: $x_i, u_i \notin B'_i$. In this case either $\min B'_i \geq u_i > x_i$, or $\max B'_i < x_i < u_i$. In either case, $x'_i = u'_i$, and so $\lceil u'_i \rceil = \lceil x'_i \rceil$.

It follows by symmetry that $\lceil \mathbf{v}' \rceil \geq \lceil \mathbf{y}' \rceil$.

Finally we show $\forall i. \forall j. \lfloor v'_j - u'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor$. Case 1: $i, j \in U$. Here $v'_j \geq y'_j$ and $u'_i \leq x'_i$, so $\lfloor v'_j - u'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor$ follows immediately. Case 2: $i \in U$, $j \notin U$. Here $u'_i \in \mathbb{Z}$, and $u'_i = \lfloor x'_i \rfloor$. Therefore

$$\lfloor v'_j - u'_i \rfloor = \lfloor v'_j \rfloor - u'_i \geq \lfloor y'_j \rfloor - u'_i = \lfloor y'_j \rfloor - \lfloor x'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor,$$

where the first inequality follows from $\lfloor \mathbf{v}' \rfloor \geq \lfloor \mathbf{y}' \rfloor$, and the last inequality follows from Lemma 7.2.13. Case 3: $i \notin U$, $j \in U$. Here $v'_j = \lceil y'_j \rceil$. Therefore

$$\lfloor v'_j - u'_i \rfloor = v'_j - \lceil u'_i \rceil = \lceil y'_j \rceil - \lceil u'_i \rceil \geq \lceil y'_j \rceil - \lceil x'_i \rceil \geq \lfloor y'_j - x'_i \rfloor,$$

where the second inequality follows from $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$, and the last inequality follows from Lemma 7.2.14. Case 4: $i, j \notin U$. If $x'_i \in \mathbb{Z}$, then

$$\lfloor y'_j - x'_i \rfloor = \lfloor y'_j \rfloor - x'_i \leq \lfloor y'_j \rfloor - \lceil u'_i \rceil \leq \lfloor v'_j \rfloor - \lceil u'_i \rceil \leq \lfloor v'_j - u'_i \rfloor,$$

where the first inequality follows from $\lceil \mathbf{u}' \rceil \leq \lceil \mathbf{x}' \rceil$, the second follows from $\lfloor \mathbf{v}' \rfloor \geq \lfloor \mathbf{y}' \rfloor$, and the third follows from Lemma 7.2.15. If $y'_j \in \mathbb{Z}$, then

$$\lfloor y'_j - x'_i \rfloor = y'_j - \lceil x'_i \rceil \leq y'_j - \lceil u'_i \rceil \leq \lfloor v'_j \rfloor - \lceil u'_i \rceil \leq \lfloor v'_j - u'_i \rfloor,$$

with the same attributions as above. So suppose $x'_i, y'_j \notin \mathbb{Z}$. Since $i, j \notin U$, it follows that $x'_i \in B'_i$ and $y'_j \in B'_j$, and so $x'_i = x_i$ and $y'_j = y_j$. Subcase A: $u_i \in B'_i$ and $v_j \in B'_j$. Here $u'_i = u_i$ and $v'_j = v_j$, so $\lfloor v'_j - u'_i \rfloor = \lfloor v_j - u_i \rfloor \geq \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$. Subcase B: $u_i \in B'_i$ and $v_j \notin B'_j$. Since $[\lfloor y_j \rfloor, \lceil y_j \rceil] \subset B'_j$ and $\lfloor v_j \rfloor \geq \lfloor y_j \rfloor$, it follows that $v_j > \lceil y_j \rceil = \lceil y'_j \rceil$, and therefore that $v'_j \geq \lceil y'_j \rceil$. Thus

$$\lfloor v'_j - u'_i \rfloor \geq \lfloor \lceil y'_j \rceil - u'_i \rfloor \geq \lfloor \lceil y'_j \rceil - \lceil x'_i \rceil \rfloor = \lceil y'_j \rceil - \lceil x'_i \rceil \geq \lfloor y'_j - x'_i \rfloor,$$

where the last inequality follows from Lemma 7.2.14. Subcase C: $u_i \notin B'_i$ and $v_j \in B'_j$. Since $[\lfloor x_i \rfloor, \lceil x_i \rceil] \subset B'_i$ and $\lceil u_i \rceil \leq \lceil x_i \rceil$, it follows that $u_i < \lfloor x_i \rfloor = \lfloor x'_i \rfloor$, and therefore that $u'_i \leq \lfloor x'_i \rfloor$. Thus

$$\lfloor v'_j - u'_i \rfloor \geq \lfloor v'_j - \lfloor x'_i \rfloor \rfloor = \lfloor v'_j \rfloor - \lfloor x'_i \rfloor \geq \lfloor y'_j \rfloor - \lfloor x'_i \rfloor \geq \lfloor y'_j - x'_i \rfloor,$$

where the last inequality follows from Lemma 7.2.13. ∎

Asynchronous Analysis.

It turns out that synchronous simulation equivalence is more discriminating than asynchronous simulation equivalence. The latter coincides with language equivalence.

Proposition 7.2.17 *The relation $\preceq_1$ is the largest asynchronous simulation on $\mathbb{R}^{n+m}$ with respect to the class of one-sided closed integral timed automata with attractors.*

Proof. We show that $\preceq_1$ is an asynchronous simulation. That it is the largest follows from Lemma 7.2.11. Suppose $(\mathbf{x},\mathbf{y}) \preceq_1 (\mathbf{u},\mathbf{v})$ and $((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) \in \mathit{delayed}_g$. We must show that there exists a $(\mathbf{u}',\mathbf{v}') \in \mathbb{R}^{n+m}$ such that $((\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')) \in \mathit{delayed}_g$ and $(\mathbf{x}',\mathbf{y}') \preceq_1 (\mathbf{u}',\mathbf{v}')$.

There exists a $t \geq 0$ such that $((\mathbf{x}+t,\mathbf{y}+t),(\mathbf{x}',\mathbf{y}')) \in \mathit{jump}_g$. Let h be any one-sided guarded command with guard $(\lceil \mathbf{x}+t \rceil, \lfloor \mathbf{y}+t \rfloor)$. Then $(\mathbf{x},\mathbf{y}) \in \mathrm{dom}\ \mathit{delayed}_h$. By Lemma 7.2.11, $(\mathbf{u},\mathbf{v}) \in \mathrm{dom}\ \mathit{delayed}_h$ as well. Therefore there exists a time $t' \geq 0$ such that $(\mathbf{u}+t',\mathbf{v}+t') \in \mathrm{dom}\ \mathit{jump}_h$. Then $\lceil \mathbf{u}+t' \rceil \leq \lceil \mathbf{x}+t \rceil$, $\lfloor \mathbf{v}+t' \rfloor \geq \lfloor \mathbf{y}+t \rfloor$, and for all i and j, $\lfloor (v_j+t')-(u_i+t') \rfloor = \lfloor v_j - u_i \rfloor \geq \lfloor y_j - x_i \rfloor = \lfloor (y_j+t)-(x_i+t) \rfloor$. So $(\mathbf{x}+t,\mathbf{y}+t) \preceq_2 (\mathbf{u}+t',\mathbf{v}+t')$. It follows that there exists a $(\mathbf{u}',\mathbf{v}')$ such that $((\mathbf{u}+t',\mathbf{v}+t'),(\mathbf{u}',\mathbf{v}')) \in \mathit{jump}_g$ and $(\mathbf{x}',\mathbf{y}') \preceq_2 (\mathbf{u}',\mathbf{v}')$. Immediately from the definitions follow $((\mathbf{u},\mathbf{v}),(\mathbf{u}',\mathbf{v}')) \in \mathit{delayed}_g$ and $(\mathbf{x}',\mathbf{y}') \preceq_1 (\mathbf{u}',\mathbf{v}')$. ■

Theorem 7.2.18 *Let $\mathcal{C}$ be the class of closed integral one-sided timed automata with attractors. For all $(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}') \in \mathbb{R}^{n+m}$, the following are equivalent:*

- $(\mathbf{x},\mathbf{y}) \approx^{\mathcal{C}}_{\mathrm{lang}} (\mathbf{x}',\mathbf{y}')$ *(i.e., clock vectors $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ are language equivalent with respect to $\mathcal{C}$),*
- $(\mathbf{x},\mathbf{y}) \approx^{\mathcal{C}}_{\mathrm{asyn}} (\mathbf{x}',\mathbf{y}')$ *(i.e., clock vectors $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ are asynchronously simulation equivalent with respect to $\mathcal{C}$),*
- *1.* $\lceil \mathbf{x} \rceil = \lceil \mathbf{x}' \rceil$, *and*

  2. $\forall i \in \{1,\ldots,n\}.\forall j \in \{1,\ldots,m\}.\ \lfloor y_j - x_i \rfloor = \lfloor y'_j - x'_i \rfloor$.

Moreover, $(\mathbf{x},\mathbf{y}) \approx^{\mathcal{C}}_{\mathrm{syn}} (\mathbf{x}',\mathbf{y}')$ *iff* $(\mathbf{x},\mathbf{y}) \approx^{\mathcal{C}}_{\mathrm{asyn}} (\mathbf{x}',\mathbf{y}')$ *and* $\lceil \mathbf{y} \rceil = \lceil \mathbf{y}' \rceil$. *Synchronous and asynchronous bisimulation equivalence with respect to $\mathcal{C}$ coincide with region* *equivalence.*

Proof. The equivalence of the first three statements follows immediately from Proposition 7.2.17 and Lemma 7.2.11. The characterization of synchronous simulation equivalence is a restatement of Proposition 7.2.16. The characterizations of synchronous and asynchronous bisimulation equivalence follow from Proposition 7.2.7 and the fact that region equivalence is a synchronous bisimulation. ■

Size of the Language Equivalence Quotient.

The coarseness of an equivalence relation on $\mathbb{R}^n$ is measured by the number of equivalence classes per unit volume. Let $\mathit{Regions}(n)$ be the number of region equivalence classes on n dimensions where all clocks are constrained to lie in the interval $(0, 1)$, and let $\mathit{OneSidedRegions}(n)$ be the number of language equivalence classes for one-sided timed automata with $n/2$ upper-bounded clocks and $n/2$ lower-bounded clocks, where all clocks are constrained to lie in the interval $(0, 1)$, as defined by Theorem 7.2.18. These classes are called *regions* and *one-sided regions* respectively. While the region equivalence of $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ requires that the relative ordering of the fractional parts of each pair of coordinates be identical for the two vectors, one-sided region equivalence only requires the same relative ordering of the fractional parts of pairs of coordinates for which one is a lower-bounded clock and one is an upper-bounded clock. It follows that one-sided region equivalence is considerably coarser than region equivalence.

Figure 7.1 illustrates the difference between regions and one-sided regions. Any one of the following four conditions bars the vectors $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ from being

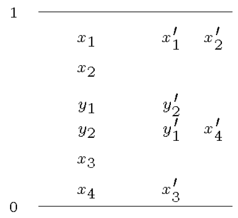

Figure 7.1: Vectors $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ are one-sided region equivalent but not region equivalent

region equivalent: (1) $x_1 > x_2$ but $x'_1 = x'_2$, (2) $y_1 > y_2$ but $y'_1 < y'_2$, (3) $x_3 > x_4$ but $x'_3 < x'_4$, and (4) $y_1 > x_4$ but $y'_1 = x'_4$. However, the relative order of the fractional parts of two upper-bounded clocks (or two lower-bounded clocks) is irrelevant to one-sided region equivalence. Therefore (1), (2), and (3) do not prevent the one-sided region equivalence of the two vectors. Nor does condition (4) prevent it, because $\lfloor y_1 - x_4 \rfloor = \lfloor y'_1 - x'_4 \rfloor$. The two vectors are in fact one-sided region equivalent.

Using characterizations of $Regions(n)$ and $OneSidedRegions(n)$ in terms of Stirling numbers of the second kind (see, e.g., [GKP89]), we show that, while the number of one-sided regions is still exponential, it is less than the number of regions by a multiplicative exponential factor.

Theorem 7.2.19 $\frac{Regions(2n)}{OneSidedRegions(2n)} = \Omega(2^n)$.

Proof. Let S_k^n be the number of ways to partition a set of n elements into k subsets, and let C_k^n be the number of k-element subsets of a set with n elements. Each region in $(0,1)^{2n}$ defines a partition of $\{1, \ldots, 2n\}$ into k subsets, where each subset defines a set of coordinates with the same value, and a permutation of the partition classes, giving the relative ordering of these values. Therefore

$$Regions(2n) = \sum_{k=1}^{2n} S_k^{2n} k!.$$

Each one-sided region with an upper-bounded clock having the highest value determines a partition of the upper-bounded clocks into k subsets, and a permutation thereof. The upper-bounded clock partition is interleaved with a partition of the lower-bounded clocks into k or $k-1$ subsets. The same analysis applies if a lower-bounded clock has the highest value. It follows that

$$\begin{aligned} OneSidedRegions(2n) &= 2\textstyle\sum_{k=1}^{n}(S_k^n)^2 k!^2 + 2\sum_{k=2}^{n} S_k^n S_{k-1}^n k!(k-1)! \\ &\le 4(\textstyle\sum_{k=1}^{n} S_k^n k!)^2 \\ &= 4Regions(n)^2. \end{aligned}$$

Therefore $OneSidedRegions(2n) = O(Regions(n)^2)$. Since every pair of n-dimensional regions can be used to form a distinct $2n$-dimensional region by placing one "on top" of the other, $\frac{Regions(2n)}{Regions(n)^2} \ge C_n^{2n}$. Now $C_n^{2n} = \frac{(2n)!}{n!^2}$, $(2n)! = \Omega((\frac{2n}{e})^{2n})$, and $n! = O(n(\frac{n}{e})^n)$. So

$$\frac{Regions(2n)}{OneSidedRegions(2n)} = \Omega(\frac{(2n)!}{n!^2}) = \Omega(\frac{(\frac{2n}{e})^{2n}}{n^2(\frac{n}{e})^{2n}}) = \Omega(\frac{2^{2n}}{n^2}) = \Omega(2^n).\blacksquare$$

7.3 Sufficient Conditions

From Theorems 7.2.2 and 7.2.4, and the proof of Corollary 3.1.7, we obtain the following basic sufficient condition for language equivalence.

Corollary 7.3.1 *Let $R = \prod_{k=1}^{n}[c_k, d_k]$, and let λ be the least common multiple of $\{c_1, d_1, \ldots, c_n, d_n\}$. For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, if*

1. $\forall i \in \{1, \ldots, n\}.\ \lceil x_i \frac{\lambda}{c_i} \rceil = \lceil x'_i \frac{\lambda}{c_i} \rceil$, *and*

2. $\forall i, j \in \{1, \ldots, n\}.$

   - $\lfloor x_j \frac{\lambda}{d_j} - x_i \frac{\lambda}{c_i} \rfloor = \lfloor x'_j \frac{\lambda}{d_j} - x'_i \frac{\lambda}{c_i} \rfloor$,
   - $\lfloor x_j \frac{\lambda}{c_j} - x_i \frac{\lambda}{d_i} \rfloor = \lfloor x'_j \frac{\lambda}{c_j} - x'_i \frac{\lambda}{d_i} \rfloor$,
   - $\lfloor x_j \frac{\lambda}{d_j} - x_i \frac{\lambda}{d_i} \rfloor = \lfloor x'_j \frac{\lambda}{d_j} - x'_i \frac{\lambda}{d_i} \rfloor$,
   - $\lfloor x_j \frac{\lambda}{c_j} - x_i \frac{\lambda}{c_i} \rfloor = \lfloor x'_j \frac{\lambda}{c_j} - x'_i \frac{\lambda}{c_i} \rfloor$.

then $\mathbf{x}$ and $\mathbf{x}'$ are language equivalent with respect to the class of closed integral rectangular automata with uniform activity rectangle R. ∎

Using Theorem 7.2.18, we obtain a much better sufficient condition:

Theorem 7.3.2 *Let $R = \prod_{k=1}^{n}[c_k, d_k]$ be an integral rectangle, and let and let λ be the least common multiple of $\{c_1, d_1, \ldots, c_n, d_n\}$. For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, if*

1. $\forall i \in \{1, \ldots, n\}.\ \lceil x_i \frac{\lambda}{c_i} \rceil = \lceil x'_i \frac{\lambda}{c_i} \rceil$, *and*

2. $\forall i, j \in \{1, \ldots, n\}.\ \lfloor x_j \frac{\lambda}{d_j} - x_i \frac{\lambda}{c_i} \rfloor = \lfloor x'_j \frac{\lambda}{d_j} - x'_i \frac{\lambda}{c_i} \rfloor$,

then $\mathbf{x}$ and $\mathbf{x}'$ are language equivalent with respect to the class of closed integral rectangular automata with uniform activity rectangle R.

Proof. From Theorems 7.2.4 and 7.2.18. ∎

7.4 Necessary Conditions

In this section, we fix an activity rectangle $R = \prod_{i=1}^{n}[c_i, d_i]$, where $c_i \geq 1$ for each i. Put $\lambda = \operatorname{lcm}\{c_1, d_1, \ldots, c_n, d_n\}$. Let $\mathcal{C}$ be the class of closed rectangular automata with uniform activity rectangle R.

Convention 7.4.1 All indices in this section range over $\{1, \ldots, n\}$. Thus in an expression such as $\forall i_1, \ldots, i_k, j_1, \ldots, j_m.\, \phi$, the quantification is over all $(k+m)$-vectors of elements of $\{1, \ldots, n\}$. ■

To derive necessary conditions for language equivalence, we need only deterministic assignments.

Definition 7.4.2 An *integer guarded command* on $\mathbb{R}^n$ is a pair $g = (B, \rho)$ where B is a closed integral n-dimensional rectangle and ρ is a partial function from $\{1, \ldots, n\}$ to $\mathbb{Z}$. The set of all integer guarded commands on $\mathbb{R}^n$ is denoted by $\mathcal{G}_n$. Define the binary relation $jump_g$ on $\mathbb{R}^n$ by $(\mathbf{x}, \mathbf{u}) \in jump_g$ iff $u_i = \rho(i)$ for all $i \in \operatorname{dom} \rho$ and $u_i = x_i$ for all $i \notin \operatorname{dom} \rho$. Define the binary relation $delayed_g^R$ on $\mathbb{R}^n$ by $(\mathbf{x}, \mathbf{u}) \in delayed_g^R$ iff either $(\mathbf{x}, \mathbf{u}) \in jump_g$ or there exist a $\mathbf{y} \in \mathbb{R}^n$ and a $t \in \mathbb{R}_{>0}$ such that $\frac{\mathbf{y}-\mathbf{x}}{t} \in R$ and $(\mathbf{y}, \mathbf{u}) \in jump_g$. For a string $\bar{g} = g_1 \cdots g_s \in \mathcal{G}_n^*$ of guarded commands, define $delayed_{\bar{g}}^R = delayed_{g_s}^R \circ \cdots \circ delayed_{g_1}^R$. ■

Convention 7.4.3 When dealing with integer guarded commands, it is convenient to refer to a closed rectangle $B = \prod_{i=1}^{n}[p_i, q_i]$ as the pair $(\mathbf{p}, \mathbf{q}) \in (\mathbb{Z} \cup \{-\infty\})^n \times (\mathbb{Z} \cup \{\infty\})^n$. Thus the integer guarded command (B, ρ) is written $(\mathbf{p}, \mathbf{q}, \rho)$. ■

Proposition 7.4.4 *For every integer guarded command $g = (\mathbf{p}, \mathbf{q}, \rho)$,*

$$\text{dom } \mathit{delayed}_g^R = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \leq \mathbf{q} \textit{ and } \forall i.\, \forall j.\, p_i \frac{\lambda}{d_i} - q_j \frac{\lambda}{c_j} \leq x_i \frac{\lambda}{d_i} - x_j \frac{\lambda}{c_j}\}.$$

Proof.

$$\begin{aligned} \text{dom } \mathit{delayed}_g^R &= \{\mathbf{x} \in \mathbb{R}^n \mid \exists t \geq 0.\, \forall i.\, [p_i, q_i] \cap (x_i + t[c_i, d_i]) \neq \emptyset\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \exists t \geq 0.\, \forall i.\, p_i \leq x_i + t d_i \text{ and } q_i \geq x_i + t c_i\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \exists t \geq 0.\, \forall i.\, \tfrac{p_i - x_i}{d_i} \leq t \leq \tfrac{q_i - x_i}{c_i}\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \leq \mathbf{q} \text{ and } \forall i.\, \forall j.\, \tfrac{p_i - x_i}{d_i} \leq \tfrac{q_j - x_j}{c_j}\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \leq \mathbf{q} \text{ and } \forall i.\, \forall j.\, p_i c_j - q_j d_i \leq x_i c_j - x_j d_i\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \leq \mathbf{q} \text{ and } \forall i.\, \forall j.\, p_i \tfrac{\lambda}{d_i} - q_j \tfrac{\lambda}{c_j} \leq x_i \tfrac{\lambda}{d_i} - x_j \tfrac{\lambda}{c_j}\}. \blacksquare \end{aligned}$$

Lemma 7.4.5 *Let $x \in \mathbb{R}$, $m, n \in \mathbb{N}_{>0}$, and $g = \gcd(m, n)$. For every $h \in \mathbb{Z}$ and every $\epsilon > 0$, there exist $p, q \in \mathbb{Z}$ such that $x \leq \epsilon q$ and $pm - qn = hg$.*

Proof. By elementary properties of greatest common divisors, there exist $p', q' \in \mathbb{Z}$ such that $p'm - q'n = hg$. For all $i \in \mathbb{N}$, $(p' + in)m - (q' + im)n = p'm - q'n = hg$. By the property of Archimedes, there exists an $i_0 \in \mathbb{N}$ with $x \leq \epsilon(q' + i_0 m)$. Put $p = p' + i_0 n$ and $q = q' + i_0 m$. $\blacksquare$

Definition 7.4.6 For distinct indices $i_1, \ldots, i_k, j_1, \ldots, j_m$, define

$$\gamma_{i_1, \ldots, i_k}^{j_1, \ldots, j_m} =_{\text{def}} \gcd\left(\frac{\lambda}{d_{i_1}}, \ldots, \frac{\lambda}{d_{i_k}}, \frac{\lambda}{c_{j_1}}, \ldots, \frac{\lambda}{c_{i_m}}\right). \blacksquare$$

Definition 7.4.7 For $k \in \mathbb{Z}$, define $\lfloor \cdot \rfloor_k : \mathbb{R} \to \mathbb{Z}$ and $\lceil \cdot \rceil_k : \mathbb{R} \to \mathbb{Z}$ by $\lfloor x \rfloor_k = \max\{ik \mid i \in \mathbb{Z} \text{ and } ik \leq x\}$ and $\lceil x \rceil_k = \min\{ik \mid i \in \mathbb{Z} \text{ and } ik \geq x\}$. $\blacksquare$

Lemma 7.4.8 $\forall i.\, \forall j \neq i.\, \forall \mathbf{x} \in \mathbb{R}^n.\, \exists g \in \mathcal{G}_n.\, [\mathbf{x} \in \text{dom } \mathit{delayed}_g^R \wedge \forall \mathbf{x}' \in \mathbb{R}^n.$

$$\lfloor x'_i \frac{\lambda}{d_i} - x'_j \frac{\lambda}{c_j} \rfloor_{\gamma_i^j} < \lfloor x_i \frac{\lambda}{d_i} - x_j \frac{\lambda}{c_j} \rfloor_{\gamma_i^j} \Rightarrow \mathbf{x}' \notin \text{dom } \mathit{delayed}_g^R].$$

Proof. Let $i \neq j$ and $\mathbf{x} \in \mathbb{R}^n$ be given. By Lemma 7.4.5, there exist $p_i, q_j \in \mathbb{Z}$ such that $x_j \leq q_j$ and $p_i \frac{\lambda}{d_i} - q_j \frac{\lambda}{c_j} = \lfloor x_i \frac{\lambda}{d_i} - x_j \frac{\lambda}{c_j} \rfloor_{\gamma_i^j}$. Put $p_\mu = -\infty$ for $\mu \neq i$, and $q_\nu = \infty$ for $\nu \neq i$. Let g be the integer guarded command $(\mathbf{p}, \mathbf{q}, \emptyset)$. By Proposition 7.4.4, $\mathbf{x} \in \text{dom } \mathit{delayed}_g^R$. But for every $\mathbf{x}'$ as above, $x'_i \frac{\lambda}{d_i} - x'_j \frac{\lambda}{c_j} < p_i \frac{\lambda}{d_i} - q_j \frac{\lambda}{c_j}$, since the latter is a multiple of γ_i^j. Thus by Proposition 7.4.4 again, $\mathbf{x}' \notin \text{dom } \mathit{delayed}_g^R$. ∎

Notice that Lemma 7.4.8 fails for $i = j$, because $p_i \leq q_i$ is necessary for a guarded command $(\mathbf{p}, \mathbf{q}, \emptyset)$. Lemma 7.4.5 cannot be extended to furnish $p \leq q$ along with its other consequences.

Corollary 7.4.9 $\forall i.\, \forall j \neq i.\, \forall \mathbf{x} \in \mathbb{R}^n.\, \exists \bar{g} \in \mathcal{G}_n^*.\, [\mathbf{x} \in \text{dom } \mathit{delayed}_{\bar{g}}^R \wedge \forall \mathbf{x}' \in \mathbb{R}^n.$

$$\lceil x'_j \frac{\lambda}{c_j} \rceil_{\gamma_i^j} > \lceil x_j \frac{\lambda}{c_j} \rceil_{\gamma_i^j} \Rightarrow \mathbf{x}' \notin \text{dom } \mathit{delayed}_{\bar{g}}^R].$$

Proof. Let $\mathbf{x} \in \mathbb{R}^n$. Let $u_i = 0$ and $u_k = x_k$ for all $k \neq i$. Apply Lemma 7.4.8 to $\mathbf{u}$ to produce the integer guarded command g. Let g' be the integer guarded command $(\mathbb{R}^n, \{(i, 0)\})$. Then $(\mathbf{x}, \mathbf{u}) \in \mathit{jump}_{g'}$ and $\mathbf{u} \in \text{dom } \mathit{delayed}_g^R$, so $\mathbf{x} \in \text{dom } \mathit{delayed}_{g'g}^R$. But for any $\mathbf{x}'$ as above, and any $\mathbf{u}'$ such that $(\mathbf{x}', \mathbf{u}') \in \mathit{delayed}_{g'}$,

$$\lfloor u'_i \frac{\lambda}{d_i} - u'_j \frac{\lambda}{c_j} \rfloor_{\gamma_i^j} = -\lceil u'_j \frac{\lambda}{c_j} \rceil_{\gamma_i^j} \leq -\lceil x'_j \frac{\lambda}{c_j} \rceil_{\gamma_i^j} < -\lceil x_j \frac{\lambda}{c_j} \rceil_{\gamma_i^j} = \lfloor u_i \frac{\lambda}{d_i} - u_j \frac{\lambda}{c_j} \rfloor_{\gamma_i^j}.$$

By definition of g, $\mathbf{u}' \notin \mathit{delayed}_g^R$. Thus $\mathbf{x}' \notin \text{dom } \mathit{delayed}_{g'g}^R$. ∎

Definition 7.4.10 For $k, m \geq 1$, let $\Theta(k, m)$ be the sentence

$$\forall \textit{ distinct } i_1, \ldots, i_k, j_1, \ldots, j_m. \forall \mathbf{x} \in \mathbb{R}^n. \exists \bar{g} \in \mathcal{G}_n. [\mathbf{x} \in \textit{delayed}_{\bar{g}}^R \wedge \forall \mathbf{x}' \in \mathbb{R}^n.$$

$$\lfloor x'_{i_1} \tfrac{\lambda}{d_{i_1}} - x'_{j_1} \tfrac{\lambda}{c_{j_1}} \rfloor_{\gamma_{i_1,\ldots,i_k}^{j_1,\ldots,j_m}} < \lfloor x_{i_1} \tfrac{\lambda}{d_{i_1}} - x_{j_1} \tfrac{\lambda}{c_{j_1}} \rfloor_{\gamma_{i_1,\ldots,i_k}^{j_1,\ldots,j_m}} \Rightarrow \mathbf{x}' \notin \text{dom } \textit{delayed}_{\bar{g}}^R],$$

where the *distinct* modifier specifies $|\{i_1, \ldots, i_k, j_i, \ldots, j_m\}| = k+m$. For $k, m \geq 1$, let $\Upsilon(k, m)$ be the sentence

$$\forall \textit{ distinct } i_1, \ldots, i_k, j_1, \ldots, j_m. \forall \mathbf{x} \in \mathbb{R}^n. \exists \bar{g} \in \mathcal{G}_n. [\mathbf{x} \in \textit{delayed}_{\bar{g}}^R \wedge$$

$$\forall \mathbf{x}' \in \mathbb{R}^n. \lceil x'_{j_1} \tfrac{\lambda}{c_{j_1}} \rceil_{\gamma_{i_1,\ldots,i_k}^{j_1,\ldots,j_m}} > \lceil x_{j_1} \tfrac{\lambda}{c_{j_1}} \rceil_{\gamma_{i_1,\ldots,i_k}^{j_1,\ldots,j_m}} \Rightarrow \mathbf{x}' \notin \text{dom } \textit{delayed}_{\bar{g}}^R]. \blacksquare$$

Corollary 7.4.11 $\Theta(1, 1) \wedge \Upsilon(1, 1)$.

Proof. Lemma 7.4.8 is $\Theta(1, 1)$, Corollary 7.4.9 is $\Upsilon(1, 1)$. $\blacksquare$

Lemma 7.4.12 $\forall k, m \geq 1. \Theta(k, m) \Rightarrow \Theta(k + 1, m)$.

Proof. Let k, m be given, and suppose $\mathbf{x} \in \mathbb{R}^n$. Since

$$\gamma_{i_1,\ldots,i_{k+1}}^{j_1,\ldots,j_m} = \text{gcd } (\frac{\lambda}{d_{i_1}}, \gamma_{i_2,\ldots,i_{k+1}}^{j_1,\ldots,j_m}),$$

it follows from Lemma 7.4.5 that there exist $p'_{i_1}, q'_{j_1} \in \mathbb{Z}$ such that $x_{j_1} \frac{\lambda}{c_{j_1}} \leq q'_{j_1} \gamma_{i_2,\ldots,i_{k+1}}^{j_1,\ldots,j_m}$ and $p'_{i_1} \frac{\lambda}{d_{i_1}} - q'_{j_1} \gamma_{i_2,\ldots,i_{k+1}}^{j_1,\ldots,j_m} = \lfloor x_{i_1} \frac{\lambda}{d_{i_1}} - x_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma_{i_1,\ldots,i_{k+1}}^{j_1,\ldots,j_m}}$. Put $p_{i_1} = p'_{i_1}$, and $p_i = -\infty$ for $i \neq i_1$. Put $q_{j_1} = \lceil q_{j_1} \gamma_{i_2,\ldots,i_{k+1}}^{j_1,\ldots,j_m} \frac{c_{j_1}}{\lambda} \rceil$ and $q_j = \infty$ for $j \neq j_1$. Let g' be the guarded command $(\mathbf{p}, \mathbf{q}, \{(i_2, 0)\})$. By the proof of Proposition 7.4.4 (applied to rectangles with rational endpoints), there exists a $\mathbf{u} \in \mathbb{R}^n$ such that $(\mathbf{x}, \mathbf{u}) \in \textit{delayed}_{g'}^R$ and $u_{j_1} \frac{\lambda}{c_{j_1}} \leq q'_{j_1} \gamma_{i_2,\ldots,i_{k+1}}^{j_1,\ldots,j_m}$. Let $\bar{g}$ be the string of integer guarded commands obtained by application of $\Theta(k, m)$ to $i_2, \ldots, i_{k+1}, j_1, \ldots, j_m$ and $\mathbf{u}$. Then

$$\lfloor x'_{i_1} \frac{\lambda}{d_{i_1}} - x'_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma_{i_1,\ldots,i_{k+1}}^{j_1,\ldots,j_m}} < \lfloor x_{i_1} \frac{\lambda}{d_{i_1}} - x_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma_{i_1,\ldots,i_{k+1}}^{j_1,\ldots,j_m}} \Rightarrow \mathbf{x}' \notin \text{dom } \textit{delayed}_{g'\bar{g}}^R.$$

For if $(\mathbf{x}', \mathbf{u}') \in \mathit{delayed}^R_{g'}$, then $u'_{j_1}\frac{\lambda}{c_{j_1}} > q'_{j_1}\gamma^{j_1,\ldots,j_m}_{i_2,\ldots,i_{k+1}}$, by the proof of Proposition 7.4.4. Thus

$$\begin{aligned}
\lfloor u'_{i_2}\frac{\lambda}{d_{i_2}} - u'_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_m}_{i_2,\ldots,i_{k+1}}} &= \lfloor -u'_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_m}_{i_2,\ldots,i_{k+1}}} \\
&< -q'_{j_1}\gamma^{j_1,\ldots,j_m}_{i_2,\ldots,i_{k+1}} \\
&\leq \lfloor -u_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_m}_{i_2,\ldots,i_{k+1}}} \\
&= \lfloor u_{i_2}\frac{\lambda}{d_{i_2}} - u_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_m}_{i_2,\ldots,i_{k+1}}}.
\end{aligned}$$

By definition of $\bar{g}$, $\mathbf{u}' \notin \mathit{delayed}^R_{\bar{g}}$. ■

Lemma 7.4.13 $\forall k, m \geq 1.\, \Theta(k, m) \Rightarrow \Theta(k, m+1)$.

Proof. Similar to the proof of the previous lemma. Let the indices be given, and let $\mathbf{x} \in \mathbb{R}^n$. Let $p'_{i_1}, q'_{j_1} \in \mathbb{Z}$ be such that $x_{j_1} \leq q_{j_1}$ and $p'_{i_1}\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k} - q'_{j_1}\frac{\lambda}{c_{j_1}} = \lfloor x_{i_1}\frac{\lambda}{d_{i_1}} - x_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_{m+1}}_{i_1,\ldots,i_k}}$. Put $p_{i_1} = \lfloor p'_{i_1}\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k}\frac{d_{i_1}}{\lambda}\rfloor$ and $p_i = -\infty$ for $i \neq i_1$. Put $q_{j_1} = q'_{j_1}$ and $q_j = \infty$ for $j \neq j_1$. Let g' be the guarded command $(\mathbf{p}, \mathbf{q}, \{(j_2, 0)\})$. Then there exists a $\mathbf{u} \in \mathbb{R}^n$ such that $(\mathbf{x}, \mathbf{u}) \in \mathit{delayed}^R_{g'}$ and $u_{i_1}\frac{\lambda}{d_{i_1}} \geq p'_{i_1}\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k}$. Let $\bar{g}$ be the string of integer guarded commands obtained by application of $\Theta(k, m)$ to $i_2, \ldots, i_{k+1}, j_1, \ldots, j_m$ and $\mathbf{u}$. Then

$$\lfloor x'_{i_1}\frac{\lambda}{d_{i_1}} - x'_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_{m+1}}_{i_1,\ldots,i_k}} < \lfloor x_{i_1}\frac{\lambda}{d_{i_1}} - x_{j_1}\frac{\lambda}{c_{j_1}}\rfloor_{\gamma^{j_1,\ldots,j_{m+1}}_{i_1,\ldots,i_k}} \Rightarrow \mathbf{x}' \notin \operatorname{dom}\, \mathit{delayed}^R_{g'\bar{g}}.$$

For if $(\mathbf{x}, \mathbf{u}) \in \mathit{delayed}^R_{g'}$, then $u'_{i_1}\frac{\lambda}{d_{i_1}} < p'_{i_1}\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k}$. Thus

$$\lfloor u'_{i_1}\frac{\lambda}{d_{i_1}} - u'_{j_2}\frac{\lambda}{c_{j_2}}\rfloor_{\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k}} < p'_{i_1}\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k} \leq \lfloor u_{i_1}\frac{\lambda}{d_{i_1}} - u_{j_2}\frac{\lambda}{c_{j_2}}\rfloor_{\gamma^{j_2,\ldots,j_{m+1}}_{i_1,\ldots,i_k}}.$$

By definition of $\bar{g}$, $\mathbf{u}' \notin \mathit{delayed}^R_{\bar{g}}$. ■

Lemma 7.4.14 $\forall k, m \geq 1.\, \Theta(k, m) \Rightarrow \Upsilon(k, m)$.

Proof. As Corollary 7.4.9 follows from Lemma 7.4.8. Let the indices and $\mathbf{x} \in \mathbb{R}^n$ be given. Let g' be the integer guarded command $(\mathbb{R}^n, \{(i_1, 0)\})$. Let $(\mathbf{x}, \mathbf{u}) \in jump_{g'}$. Let $\bar{g}$ be the string of integer guarded commands obtained by application of $\Theta(k, m)$ to $\mathbf{u}$. Then $\mathbf{x} \in \text{dom } delayed^R_{g'\bar{g}}$. However if $(\mathbf{x}', \mathbf{u}') \in delayed^R_{g'}$ and $\lceil x'_{j_1} \frac{\lambda}{c_{j_1}} \rceil_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}} > \lceil x_{j_1} \frac{\lambda}{c_{j_1}} \rceil_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}}$, then

$$\lfloor u'_{i_1} \frac{\lambda}{d_{i_1}} - u'_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}} < \lfloor u_{i_1} \frac{\lambda}{d_{i_1}} - u_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}}.$$

By definition of $\bar{g}$, $\mathbf{u}' \notin \text{dom } delayed^R_{\bar{g}}$. ∎

Lemma 7.4.15 $\forall k, m \geq 1.\, \Theta(k, m) \wedge \Upsilon(k, m)$.

Proof. By induction, from Lemmas 7.4.8, 7.4.12, 7.4.13, and 7.4.14. ∎

Theorem 7.4.16 *Let $\mathcal{C}$ be the class of closed integral rectangular automata with uniform integral activity rectangle $R = \prod_{i=1}^n [c_i, d_i]$, where $c_i \geq 1$ for each i. Let λ be the least common multiple of $\{c_1, d_1, \ldots, c_n, d_n\}$. For all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, $\mathbf{x} \approx^{\mathcal{C}}_{\text{lang}} \mathbf{x}'$ implies that for all distinct $i_1, \ldots, i_k, j_1, \ldots, j_m$,*

1. $\lceil x'_{j_1} \frac{\lambda}{c_{j_1}} \rceil_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}} = \lceil x_{j_1} \frac{\lambda}{c_{j_1}} \rceil_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}}$, *and*
2. $\lfloor x'_{i_1} \frac{\lambda}{d_{i_1}} - x'_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}} = \lfloor x_{i_1} \frac{\lambda}{d_{i_1}} - x_{j_1} \frac{\lambda}{c_{j_1}} \rfloor_{\gamma^{j_1,\ldots,j_m}_{i_1,\ldots,i_k}}$.

Proof. Immediate from Lemma 7.4.15. ∎

Compare Theorem 7.4.16 with Theorem 7.3.2. In two dimensions, when $R = [c, d]^2$ is a square, then the necessary condition of Theorem 7.4.16 coincides with the exact condition for simulation equivalence given in Theorem 6.1.6. This provides a second proof that synchronous simulation equivalence coincides with language equivalence in this case.

7.5 Activity Cubes

When the activity rectangle is a cube, the necessary conditions and sufficient conditions are quite close. Activity cubes are useful for modeling systems with drifting clocks in which the clock drift is uniform.

Definition 7.5.1 A *cube* is a rectangle of the form $[c, d]^n$, where $c, d \in \mathbb{R}$. ■

Corollary 7.5.2 *Let* $R = [c, d]^n$ *be an integral cube with* $n \geq 2$*, and let* $\mathcal{C}$ *be the class of closed integral rectangular automata with uniform activity rectangle* R*. For all* $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$*,* $\mathbf{x} \approx^{\mathcal{C}}_{\text{lang}} \mathbf{x}'$ *implies that*

1. $\forall i \in \{1, \ldots, n\}.\ \lceil x_i \frac{\lambda}{c} \rceil = \lceil x'_i \frac{\lambda}{c} \rceil$*, and*

2. $\forall$*distinct* $i, j \in \{1, \ldots, n\}.\ \lfloor x_j \frac{\lambda}{d} - x_i \frac{\lambda}{c} \rfloor = \lfloor x'_j \frac{\lambda}{d} - x'_i \frac{\lambda}{c} \rfloor$.

Proof. Put $\lambda = \text{lcm}\,\{c, d\}$. The corollary is immediate from Theorem 7.4.16, using the fact that $\gcd\,(\frac{\lambda}{c}, \frac{\lambda}{d}) = 1$. ■

Looking at Theorem 7.3.2 as applied to cubes, we see that the only difference is in the quantification over i, j in (2): for the sufficient condition we have $\forall i, j$; for the necessary condition we have $\forall$*distinct* i, j.

Chapter 8

Discrete Time

The theory of rectangular automata in the discrete time domain is particularly pleasing. The discrete-time reachability problem is PSPACE-complete for the class of positive rectangular automata. But if either positivity or rectangularity is relaxed, then it is easy to code undecidable problems into discrete-time reachability problems. Therefore the class of positive rectangular automata lies at the boundary of decidability.

Every positive rectangular automaton A has finite bisimilarity quotient: for n-dimensional, h-definable, integral A, the number of states in the bisimilarity quotient of the discrete-time transition system of A is no greater than $|V|(h+2)^n$. From this follows the PSPACE-completeness of the reachability problem and the CTL* model-checking problem. Moreover, the discrete time domain admits a compelling version of the control problem, which is EXPTIME-complete.

We use the following convention:

Convention 8.0.3 In this chapter, we assume for convenience that all rectangles and all rectangular automata are integral. However, all of the complexity results that we derive apply unchanged to non-integral rectangular automata. ■

8.1 The Discrete-Time Transition System

Definition 8.1.1 Let A be a triangular automaton. Let $\Pi_A = \{\pi_v \mid v \in V_A\}$, and define $(w, \mathbf{x}) \models_A \pi_v$ iff $v = w$. The triangular automaton A defines the *discrete-time* transition system

$$S_A^{\mathbb{N}} =_{\text{def}} (Q_A, \Sigma \cup \{1\}, \rightarrow, Q_A^0, \Pi_A, \models_A). \blacksquare$$

Notice that time steps of duration 1 only appear in the discrete-time transition system. The time step relation $\xrightarrow{1}$ reduces to a very simple form: for states $(v, \mathbf{x}), (v, \mathbf{y}) \in Q_A$, $(v, \mathbf{x}) \xrightarrow{1} (v, \mathbf{y})$ iff $\mathbf{y} - \mathbf{x} \in act(v)$.

In discrete time, rectangular automata cannot distinguish between two vectors $\mathbf{y}$ and $\mathbf{z}$ whose integer parts are the same, so long as the components of the two vectors having integral values coincide. More precisely, two states of a rectangular automaton are bisimilar if they have the same discrete part, and if the floors and ceilings of their continuous components coincide.

Definition 8.1.2 Define the *discrete equivalence relation* $\sim_1^{dis}$ on $\mathbb{R}$ by $x \sim_1^{dis} y$ iff $\lfloor x \rfloor = \lfloor y \rfloor$ and $\lceil x \rceil = \lceil y \rceil$. Define $\sim_n^{dis}$ on $\mathbb{R}^n$ by $\mathbf{x} \sim_n^{dis} \mathbf{y}$ iff $x_i \sim_1^{dis} y_i$ for all $1 \leq i \leq n$. ■

Lemma 8.1.3 *For all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} \sim_n^{dis} \mathbf{y}$ *iff for every rectangle* $B \subset \mathbb{R}^n$, $\mathbf{x} \in B$ *iff* $\mathbf{y} \in B$. ■

Proof. By Convention 8.0.3. ■

Definition 8.1.4 Let A be an n-dimensional rectangular automaton. The equivalence relation $\equiv^A$ on Q_A is defined by $(v, \mathbf{y}) \equiv^A (w, \mathbf{z})$ iff $v = w$ and $\mathbf{y} \sim_n^{dis} \mathbf{z}$. ■

Theorem 8.1.5 *For every rectangular automaton* A*, the equivalence relation* $\equiv^A$ *is a bisimulation on* $S_A^{\mathbb{N}}$.

Proof. Suppose $(v, \mathbf{x}) \equiv^A (v, \mathbf{y})$ and $(v, \mathbf{x}) \xrightarrow{\pi} (w, \mathbf{x}')$. First assume that $\pi \in \Sigma$. In this case there exists an edge $e = (v, w)$ such that $event(e) = \pi$, $\mathbf{x} \in preguard(e)$, $\mathbf{x}' \in postguard(e)$, and for each $i \notin update(e)$, $x_i = x'_i$. Define $\mathbf{y}'$ by $y'_i = y_i$ for $i \notin update(e)$, and $y'_i = x'_i$ for $i \in update(e)$. By Lemma 8.1.3, $\mathbf{y} \in preguard(e)$ and $\mathbf{y}' \in postguard(e) \cap inv(w)$. It follows that $(v, \mathbf{y}) \xrightarrow{\pi} (w, \mathbf{y}')$.

Next suppose $\pi = 1$. Then $\mathbf{x}' - \mathbf{x} \in act(v)$. We must show that there exists a vector $\mathbf{y}' \in inv(v)$ such that $\mathbf{y}' - \mathbf{y} \in act(v)$ and $\mathbf{y}' \sim_n^{dis} \mathbf{x}'$ (notice that by Lemma 8.1.3, $\mathbf{y}' \sim_n^{dis} \mathbf{x}'$ implies $\mathbf{y}' \in inv(v)$). We do this one coordinate at a time. If $x_i \in \mathbb{Z}$ then $y_i = x_i$ because $\mathbf{x} \sim_n^{dis} \mathbf{y}$. So put $y'_i = x'_i$, when $y'_i - y_i = x'_i - x_i \in act(v)_i$. If $x_i \notin \mathbb{Z}$ then $\lfloor x_i \rfloor < x_i, y_i < \lceil x_i \rceil$. The set $act(v)_i$ is an interval, say with endpoints $a, b \in \mathbb{Z}$ (it is easy to extend the argument to the case of unbounded $act(v)_i$). Thus $act(v)_i$ contains the open interval (a, b), and $x'_i \in [x_i + a, x_i + b]$. We show that there exists a number $c \in (a, b)$ such that $y_i + c \sim_1^{dis} x'_i$. Since $a, b \in \mathbb{Z}$ and $y_i \sim_1^{dis} x_i$, it follows that $y_i + a \sim_1^{dis} x_i + a$ and $y_i + b \sim_1^{dis} x_i + b$. Thus the closed interval $[y_i + a, y_i + b]$ intersects the same $\sim_1^{dis}$-equivalence classes as does

$[x_i + a, x_i + b]$. Since neither $y_i + a$ nor $y_i + b$ is an integer, the same holds true for the open interval $(y_i + a, y_i + b)$. Therefore there exists a number $c \in (a, b)$ such that $y_i + c \sim_1^{dis} x_i'$. ■

For h-definable positive automata, all continuous values greater than h are indistinguishable.

Definition 8.1.6 Let $h \in \mathbb{N}_{>0}$. Define the equivalence relation $\sim_n^{dis_h}$ on the positive orthant $\mathbb{R}^n_{\geq 0}$ by $\mathbf{y} \sim_n^{dis_h} \mathbf{z}$ iff for each $1 \leq i \leq n$, either $y_i \sim_1^{dis} z_i$ or both y_i and z_i are greater than h. ■

Lemma 8.1.7 *For all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n_{\geq 0}$, $\mathbf{x} \sim_n^{dis_h} \mathbf{y}$ *iff for every h-definable rectangle* $B \subset \mathbb{R}^n_{\geq 0}$, $\mathbf{x} \in B$ *iff* $\mathbf{y} \in B$. ■

Theorem 8.1.8 *For every n-dimensional, h-definable, positive rectangular automaton A, the bisimilarity quotient of $S_A^{\mathbb{N}}$ has no more than $|V_A|(2h+2)^n$ states.*

Proof. Define the equivalence relation $\equiv_h^A$ on Q_A by $(v, \mathbf{x}) \equiv_h^A (w, \mathbf{y})$ iff $v = w$ and $\mathbf{x} \sim_n^{dis_h} \mathbf{y}$. We show that $\equiv_h^A$ is a bisimulation on S_A. Suppose $(v, \mathbf{x}) \equiv_h^A (v, \mathbf{y})$ and $(v, \mathbf{x}) \xrightarrow{\sigma} (w, \mathbf{x}')$. First assume that $\sigma \in \Sigma$. In this case there exists an edge $e = (v, w)$ such that $event(e) = \sigma$, $\mathbf{x} \in preguard(e)$, $\mathbf{x}' \in postguard(e)$, and for each $i \notin update(e)$, $x_i = x_i'$. Define $\mathbf{y}'$ by $y_i' = y_i$ for $i \notin update(e)$, and $y_i' = x_i'$ for $i \in update(e)$. By Lemma 8.1.7, $\mathbf{y} \in preguard(e)$ and $\mathbf{y}' \in postguard(e) \cap inv(v')$. It follows that $(w, \mathbf{y}) \xrightarrow{\sigma} (v', \mathbf{y}')$.

Next suppose that $\sigma = 1$. Then $\mathbf{x}' - \mathbf{x} \in act(v)$. Recall the definition of projection (Definitions 2.2.13 and 2.2.14) from Chapter 2. Let $J \subset \{1, \ldots, n\}$ be the set of coordinates j such that $x_j \leq h$. Then $\mathbf{x}\lceil_J \sim^{dis}_{|J|} \mathbf{y}\lceil_J$. By Theorem 8.1.5, there exists a vector $\mathbf{y}' \in \mathbb{R}^n$ such that

1. $(v, \mathbf{y}\lceil_J) \xrightarrow{1} (v, \mathbf{y}'\lceil_J)$ in $A\lceil_J$,

2. $\mathbf{x}'\lceil_J \sim^{dis}_{|J|} \mathbf{y}'\lceil_J$, and

3. for all $i \notin J$, $y'_i - y_i \in act(v)_i$.

Now for $j \in J$, $x'_j \sim^{dis}_1 y'_j$, and for $i \notin J$, $x'_i, y'_i > h$ *because A is positive*. It follows that $\mathbf{x}' \sim^{dis_h}_n \mathbf{y}'$, and therefore that $(v, \mathbf{x}') \equiv^A_h (v, \mathbf{y}')$. By Lemma 8.1.7, $\mathbf{y}' \in inv(v)$, and so $(v, \mathbf{y}) \xrightarrow{1} (v, \mathbf{y}')$.

Thus $\equiv^A_h$ is a bisimulation on S_A. The number of $\equiv^A_h$-equivalence classes is $|V_A|(2h+2)^n$. ■

8.2 Reachability

The discrete-time reachability problem asks whether a given rectangular zone is reachable in the discrete-time transition system of a given rectangular automaton. For positive rectangular automata, this problem is PSPACE-complete, but for the entire class of rectangular automata, and for the class of positive triangular automata, the problem is undecidable. Thus the positive rectangular automata form a maximal class for which the reachability problem is decidable. Another such class (incomparable with the class of positive rectangular automata), is the class of initialized rectangular automata. The decidability of this class follows from arguments presented in Chapter 3.

Definition 8.2.1 The *discrete-time reachability problem* for a class $\mathcal{C}$ of triangular automata is stated in the following way:

> *Given a triangular automaton $A \in \mathcal{C}$ and a* rectangular *zone $Z \subset Q_A$, determine whether Z is reachable in the discrete-time transition system $S_A^{\mathbb{N}}$.* ■

8.2.1 Decidability

Due to the finite bisimilarity quotient, the discrete-time reachability problem is decidable for the class of positive rectangular automata,

Theorem 8.2.2 *The discrete-time reachability problem is PSPACE-complete for the class of positive rectangular automata.*

Proof. PSPACE-hardness follows from the PSPACE-hardness of discrete-time reachability on the subclass of timed automata [Alu91]. We now prove inclusion in PSPACE. Given A and $Z \subset Q_A$, let $\hat{A}$ and $\hat{v} \in V_{\hat{A}}$ be defined as in the proof of Theorem 3.2.10, so that Z is reachable in $S_A^{\mathbb{N}}$ iff $\{\hat{v}\} \times inv_{\hat{A}}(\hat{v})$ is reachable in $S_{\hat{A}}^{\mathbb{N}}$. The number of states in the bisimilarity quotient of $S_{\hat{A}}^{\mathbb{N}}$ is singly exponential in the size of A and Z. It follows that the reachability of $\{\hat{v}\} \times inv_{\hat{A}}(\hat{v})$ in $S_{\hat{A}}^{\mathbb{N}}$ can be computed in space polynomial in the size of A and Z. ■

Using the construction of Chapter 3 of the multirate automaton M_A, which is independent of the time domain, we obtain the following proposition.

Proposition 8.2.3 [PV94] *The discrete-time reachability problem for the class of initialized rectangular automata is PSPACE-complete.* ■

8.2.2 Undecidability

In this section we show that both conditions, positivity and rectangularity, are required for a decidable reachability problem. It follows that the class of positive rectangular automata is a maximal class for which the discrete-time control problem is decidable.

Theorem 8.2.4 *The discrete-time reachability problem is undecidable for the class of rectangular automata.*

Proof. We reduce the halting problem for two-counter machines, which is known to be undecidable [HU79], to the reachability problem on the class of rectangular automata. A two-counter machine M consists of a finite control, and two counters, C_0 and C_1, which range over the natural numbers. The machine M can perform three operations: increment a counter, decrement a counter (if the counter is nonzero), and test whether a counter has value 0, branching according to the result of the test. The machine M halts if it enters its final state q_f.

Given M, we define a rectangular automaton A and a control mode v_f such that v_f is reachable iff M halts. The rectangular automaton A is three-dimensional. It is convenient to number the coordinates 0, 1, and 2. Variables 0 and 1 represent the two counters. Variable 2 is used to keep track of time: for every vertex $v \in V_A$, $act(v)_2 = \{1\}$. Thus variable 2 is a precise clock. To increment variable i, we use a vertex v in which $act(v)_i = \{1\}$ and $act(v)_{1-i} = \{0\}$. Using variable 2 and an invariant, we can force exactly one time unit to pass in this vertex. See Figure 8.1. To test if $C_i = 0$, we use a vertex v in which $act(v)_0 = act(v)_1 = \{0\}$, along with

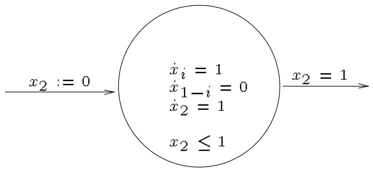

Figure 8.1: Incrementing x_i

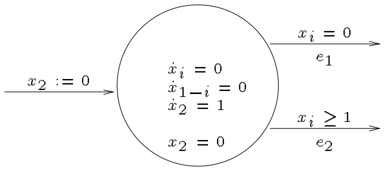

Figure 8.2: Testing $x_i = 0$

two outgoing edges e_1 and e_2, where $preguard(e_1)_i = \{0\}$ and $preguard(e_2)_i = [1, \infty)$. See Figure 8.2. Finally, to decrement variable i, we first test to make sure it is greater than 0, and then we use a vertex v in which $act(v)_i = \{-1\}$ and $act(v)_{1-i} = \{0\}$. See Figure 8.3. By encoding the current state of M into the control modes of A, we may find a control mode v_f of A with the desired property. ■

Theorem 8.2.5 *The discrete-time reachability problem is undecidable for the class of positive triangular automata.*

Proof. Again we use a reduction from the halting problem for two-counter machines. Due to positivity, we cannot decrement by using a slope of -1. The construction requires a fourth continuous variable, variable 3. The value of counter C_i corresponds to the difference between variable i and variable 3. To increment variable i, we use a vertex v in which $act(v)_i = 1$ and $act(v)_{1-i} = act(v)_3 = 0$, as in the

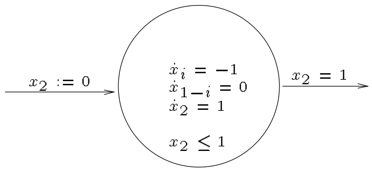

Figure 8.3: Decrementing x_i

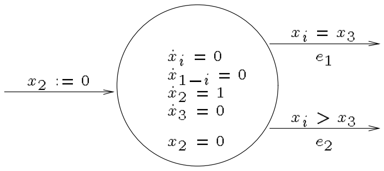

Figure 8.4: Using a triangle to test $x_i = 0$

proof of Theorem 8.2.4. To test if $C_i = 0$, we use the triangles $\{\mathbf{x} \in \mathbb{R}^4 \mid x_i = x_3\}$, and $\{\mathbf{x} \in \mathbb{R}^4 \mid x_i > x_3\}$ as preguards (it is possible to avoid the strict comparison $x_i > x_3$ by the introduction of a fifth variable). See Figure 8.4. To decrement variable i, we use a vertex v in which $act(v)_i = \{0\}$ and $act(v)_{1-i} = act(v)_3 = \{1\}$. See Figure 8.5. ■

8.3 CTL* Model Checking

Definition 8.3.1 A computation $\kappa = q_0\pi_0q_1\pi_1\ldots$ of the discrete-time transition system $S_A^{\mathbb{N}}$ is *divergent* if $\pi_i = 1$ for infinitely many i. ■

Definition 8.3.2 The *discrete-time model checking problem for CTL* under time divergence* for a class $\mathcal{C}$ of rectangular automata is stated in the following way:

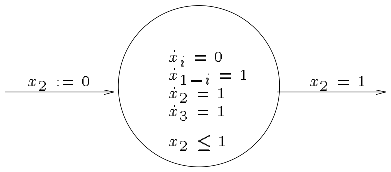

Figure 8.5: Decrementing x_i using only nonnegative slopes

Given a rectangular automaton $A \in \mathcal{C}$ and a state formula ϕ of CTL over the time-abstract transition system $S_A^{\mathbb{N}}$, let $\mathcal{F}$ be the fairness condition consisting of all divergent computations of $S_A^{\mathbb{N}}$. Determine whether all initial states of A satisfy ϕ under the fairness condition $\mathcal{F}$.* ■

Theorem 8.3.3 *The discrete-time model checking problem for CTL* under time divergence for the class of positive rectangular automata is PSPACE-complete.*

Proof. By Theorems 8.1.8 and 2.1.27. Divergence in discrete time can be detected by a Büchi acceptance condition. ■

If desired, specification variables can be added to the logic, as in the logic TCTL [ACD93].

8.4 Controller Synthesis

A natural control model for real-time systems involves a controller that samples the system state once per time unit, and then issues a command based upon its measurement. Given a rectangular automaton A, we define the transitions system $S_C(A)$, on which any control map behaves in this manner.

Definition 8.4.1 With a rectangular automaton A we associate the *control transition system*

$$S_C(A) =_{\rm def} (Q_A \times \{control, time\}, \Sigma \cup \{1\}, \Rightarrow, Q_0 \times \{control\}, \Pi_A, \models'),$$

where $((v, \mathbf{x}), \lambda) \models' \pi$ iff $(v, \mathbf{x}) \models_A \pi$, and the relation $\Rightarrow$ is defined by

- $((v, \mathbf{x}), control) \overset{\sigma}{\Rightarrow} ((w, \mathbf{y}), time)$ iff $(v, \mathbf{x}) \xrightarrow{\sigma} (w, \mathbf{y})$ in S_A, for $\sigma \in \Sigma$,
- $((v, \mathbf{x}), time) \overset{1}{\Rightarrow} ((w, \mathbf{y}), control)$ iff $(v, \mathbf{x}) \xrightarrow{1} (w, \mathbf{y})$ in S_A. ∎

Thus in the control transition system $S_C(A)$, the controller and the system take turns. First the controller specifies an edge step, and then one time unit passes in a time step.

Definition 8.4.2 Let $\mathcal{C}$ be a class of triangular automata. The *discrete-time control decision problem on* $\mathcal{C}$ is stated in the following way:

> *Given a triangular automaton $A \in \mathcal{C}$ and a vertex $v \in V$, determine whether there exists a control map κ on $S_C(A)$ such that in the closed-loop system $\kappa(S_C(A))$, the region*
>
> $$\{(v', \mathbf{y}) \in Q_A \mid v' = v\} \times \{control, time\}$$
>
> *is not reachable.*

If such a control map κ exists, we say that v is *avoidable*, if not, then v is *unavoidable*. *Discrete-time controller synthesis* further requires the computation of the control map κ, when v is avoidable. ∎

We solve the discrete-time control decision problem for positive rectangular automata by finding a rectangular automaton *Control*(A) such that $S^{\mathbb{N}}_{Control(A)}$ is isomorphic to $S_C(A)$, and then applying Proposition 2.1.36 and Theorem 8.1.8.

Theorem 8.4.3 *On the class of positive rectangular automata, the discrete-time control decision problem is EXPTIME-complete, and discrete-time controller synthesis is computable in EXPTIME.*

Proof. Let A be a positive, n-dimensional rectangular automaton. We define an $(n+1)$-dimensional rectangular automaton *Control*(A) with the following components:

Control graph. Identical to that of A: $V_{Control(A)} = V_A$ and $E_{Control(A)} = E_A$.

Invariants. For every vertex v, $inv_{Control(A)}(v) = inv_A(v) \times [0, 1]$.

Activity rectangles. For every vertex v, $act_{Control(A)}(v) = act_A(v) \times \{1\}$.

Initial conditions. For every vertex v, $init_{Control(A)}(v) = init_A(v) \times \{1\}$.

Discrete actions. For every edge e, $preguard_{Control(A)}(e) = preguard_A(e) \times \{1\}$, $update_{Control(A)}(e) = update_A(e) \cup \{n+1\}$, and $postguard_{Control(A)}(e) = postguard_A(e) \times \{0\}$.

Events. Identical to those of A: $\Sigma_{Control(A)} = \Sigma_A$ and $event_{Control(A)} = event_A$.

Continuous component $n+1$ of *Control*(A) is an auxiliary variable, which we will call c, that is used to force time steps and edge steps to alternate. The initial condition of each vertex specifies $c = 1$. The invariant of each vertex specifies

$0 \leq c \leq 1$, and the activity of each vertex specifies that $\frac{dc}{dt} = 1$. Thus initially, no time step is enabled, and so an edge step must be taken. The preguard of each edge e specifies $c = 1$, and the postguard specifies $c = 0$. Thus after e is traversed, $c = 0$, and so no edge is enabled. Then, after a time step of duration 1, $c = 1$, time steps are disabled, and edge steps are enabled. Thus time steps of duration 1 alternate with edge steps in $S^{\mathbb{N}}_{Control(A)}$.

It follows that the map $\alpha : Q_{Control(A)} \to Q_A \times \{control, time\}$ defined by $\alpha(v, \mathbf{y}, 0) = (v, \mathbf{y}, time)$ and $\alpha(v, \mathbf{y}, 1) = (v, \mathbf{y}, control)$ is an isomorphism between the transition systems $S^{\mathbb{N}}_{Control(A)}$ and $S_C(A)$. Therefore it suffices to solve control problems on the former rather than the latter. By Theorem 8.1.8, the bisimilarity quotient of $S^{\mathbb{N}}_{Control(A)}$ has $O(|V|(2h+2)^{n+1})$ equivalence classes, where A is h-definable. By the proof of Proposition 2.1.36, the control decision problem and controller synthesis can be computed in time polynomial in the number of bisimulation equivalence classes. Therefore the discrete-time control decision problem and controller synthesis can be solved in exponential time.

EXPTIME-hardness follows a reduction from the halting problem for alternating Turing Machines using polynomial space [CKS81]. Let M be an alternating Turing Machine using space $p(k)$, and let s be an input word. We assume strict alternation between AND and OR branches. The set of states of M is the disjoint union $U = U_0 \cup U_1 \cup \{u_f\}$. The set U_0 is the set of AND-branching states, the set U_1 is the set of OR-branching states, and u_f is the unique accepting state, from which there is no further computation. Each transition t of M consists of a source state $u \in U_i$ ($i \in \{0, 1\}$), a symbol σ from the alphabet Σ of M, and a list of triples $(u_j, \sigma_j, d_j)_{j \in J}$, where $u_j \in U_{1-i}$ is a target state, $\sigma_j \in \Sigma$ is written on the current tape cell, and $d_j \in \{-1, 1\}$ gives the direction moved by the tape head. If $u \in U_0$, then all of the triples must produce accepting computation trees for M to accept its input. If $u \in U_1$, then M accepts its input if some triple produces an accepting computation tree.

We define a positive rectangular automaton A with a vertex u_f such that M accepts s iff u_f is unavoidable. The rectangular automaton A uses $p(|s|)$ variables $x_1, \ldots, x_{p(|s|)}$, one to store the contents of each tape cell. The set of vertices of A is $\{u_f\} \cup (U \times \{1, \ldots, p(|s|)\})$. For states of the latter form, the first component gives the state of M, and the second gives the position of the tape head. The activity rectangles are defined by $act_A(v) = \{\mathbf{0}\}$ for every vertex v. Thus $\xrightarrow{1}$ transitions have no effect and henceforth we ignore them. This allows us to assume that the tape alphabet of M is $\{0, 1, 2, \ldots, k\}$ for some k, where 0 is the "blank" tape symbol. The correspondence that is maintained in the translation is this: the ith continuous component holds the contents of the ith tape cell. The initial condition is defined by $init(u, i) = \emptyset$ except if u is the initial state u_0 of M and $i = 1$, when $init(u_0, 1) = \prod_{j=1}^{|s|}\{s_j\} \wedge \bigwedge_{j=|s|+1}^{p(|s|)}\{0\}$. This initializes the correspondence. The invariant conditions are trivial: $inv(u, i) = \mathbb{R}_{\geq 0}^{p(|s|)}$ for all u and i.

For every transition $t = (u, \sigma, (u_j, \sigma_j, d_j)_{j \in J})$ of M, and every $1 \leq i \leq p(|s|)$, we define in A one edge $e_{t,j}$ for each $j \in J$. If u_j is the accepting state u_f, then edge $e_{t,j}$ has source vertex (u, i) and target vertex u_f. Otherwise edge $e_{t,j}$ has source (u, i) and target $(u_j, i + d_j)$ (we assume that some coding scheme is used in M so that the tape head never "falls off" the edge of the tape). The guards and update set are defined by $preguard_A(e_{t,j}) = \mathbb{R}^{i-1} \times \{\sigma\} \times \mathbb{R}^{p(|s|)-i}$, $update_A(e_{t,j}) = \{i\}$, and $postguard_A(e_{t,j}) = \mathbb{R}^{i-1} \times \{\sigma_j\} \times \mathbb{R}^{p(|s|)-i}$. Thus the correspondence is maintained: when σ_j is written on the ith tape cell, the value of the ith continuous component changes to σ_j. If $u \in U_0$ is an AND state, then $event_A(e_{t,j}) = (t, j)$. If $u \in U_1$ is an OR state, then $event_A(e_{t,j}) = 0$.

Thus in automaton states corresponding to AND configurations of M, a control map chooses which of the available transitions to take. In automaton states corresponding to OR configurations of M, only one event is available, and so the control map is powerless. A control map κ that witnesses the avoidability of u_f is also a witness for the rejection of s by M. It follows that M accepts input s iff u_f is unavoidable. ■

The discrete-time reachability problem is trivially reducible to the discrete-time control problem, because control mode v is reachable in automaton T iff v is unavoidable in the automaton $\hat{T}$ defined by replacing the event alphabet of T by the singleton $\{0\}$, and adding idling edges $e_v = (v, v)$ to each vertex $v \in V$, where $preguard_{\hat{T}}(e_v) = postguard_{\hat{T}}(e_v) = \mathbb{R}^n$ and $update_{\hat{T}}(e) = \emptyset$.

Corollary 8.4.4 *The discrete-time control decision problem is undecidable both on the class of rectangular automata and on the class of positive triangular automata.*

Chapter 9

Future Work

The main gap in the theory of rectangular hybrid automata as presented in this dissertation is an adequate statement of, and solution to, the dense time controller synthesis problem. One natural model, used in the work of Nerode and Kohn, is for the controller to sample the continuous state once every Δ time units. We used this model in the discrete-time case, where there was no mystery as to the proper value for Δ. In dense time, successively smaller values for Δ can be analyzed until either an adequate controller is found or Δ becomes too small to be of practical value. Another model for dense-time control allows the controller to react arbitrarily quickly, as in the differential games of Isaacs. The creation of a satisfactory model of this sort is greatly complicated by the requirement of time divergence. In particular, if a sequence-based semantics is used, the controller must be prevented from issuing an infinite number of controls actions in a finite amount of time.

The hybrid automaton is a very synchronous formalism for the specification of hybrid systems, because the invariants and guards of a hybrid automaton can force events to occur at precise time instants. Exact measurements drive the undecidability results of Chapter 4, and all of the undecidability results for hybrid automata of which the author is familiar. A similar problem is encountered in the real-time logic MITL, which introduces time-bounded temporal operators of the form $\Diamond_{[a,b]}$ to propositional linear temporal logic. The formula $\Diamond_{[a,b]}\phi$ is true in the present iff ϕ is true at some instant between a and b seconds in the future. The satisfiability problem for this logic is decidable iff the case of $a = b$ is disallowed.[1] It may be that a more asynchronous model would have more pleasant computational characteristics.

[1] MITL (without $a = b$) can still express exact timing properties: the formula $\Diamond_{[a,b]}\phi \wedge \Box_{[a,b)}\neg\phi$ is true at present iff the next time that ϕ is true is in exactly b seconds.

Bibliography

[ACD90] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking for real-time systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, pages 414–425. IEEE Computer Society Press, 1990.

[ACD93] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.

[ACH93] R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing accumulated delays in real-time systems. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 181–193. Springer-Verlag, 1993.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 209–229. Springer-Verlag, 1993.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 1–13. Springer-Verlag, 1994.

[AFH96] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AHH96] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[AK96] R. Alur and R.P. Kurshan. Timing analysis in COSPAN. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 220–231. Springer-Verlag, 1996.

[AL88] M. Abadi and L. Lamport. The existence of refinement mappings. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 165–175. IEEE Computer Society Press, 1988.

[AL92] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 1–27. Springer-Verlag, 1992.

[Alu91] R. Alur. *Techniques for Automatic Verification of Real-time Systems.* PhD thesis, Stanford University, 1991.

[ALW89] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *ICALP 89: Automata, Languages, and Programming*, Lecture Notes in Computer Science 372, pages 1–17. Springer-Verlag, 1989.

[AM94] A. Anuchitanukul and Z. Manna. Realizability and synthesis of reactive modules. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 156–168. Springer-Verlag, 1994.

[AMP95a] E. Asarin, O. Maler, and A. Pnueli. On the analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–64, 1995.

[AMP95b] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999. Springer-Verlag, 1995.

[Bal94] Felice Balarin. *Iterative Methods for Formal Verification of Discrete Event Systems.* PhD thesis, University of California Berkeley, 1994.

[BBLS92] S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property-preserving simulations. In G. von Bochmann and D.K. Probst, editors, *CAV 92: Computer-aided Verification*, Lecture Notes in Computer Science 663, pages 260–273. Springer-Verlag, 1992.

[BCG88] M.C. Browne, E.M. Clarke, and O. Grümberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[BCM+92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[BER94a] A. Bouajjani, R. Echahed, and R. Robbana. Verification of context-free timed systems using linear hybrid observers. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science, pages 118–131. Springer-Verlag, 1994.

[BER94b] A. Bouajjani, R. Echahed, and R. Robbana. Verifying invariance properties of timed systems with duration variables. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 193–210. Springer-Verlag, 1994.

[BES93] A. Bouajjani, R. Echahed, and J. Sifakis. On model checking for real-time properties with durations. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science*, pages 147–159. IEEE Computer Society Press, 1993.

[BFH90] A. Bouajjani, J.-C. Fernandez, and N. Halbwachs. Minimal model generation. In R.P. Kurshan and E.M. Clarke, editors, *CAV 90: Computer-aided Verification*, Lecture Notes in Computer Science 531, pages 197–203. Springer-Verlag, 1990.

[BH88] Y. Brave and M. Heymann. Formulation and control of real-time discrete-event processes. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 1131–1132, 1988.

[BL69] J.R. Büchi and L.H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the AMS*, 138:295–311, 1969.

[BLL+96] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL: a tool-suite for automatic verification of real-time systems. In

R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 232–243. Springer-Verlag, 1996.

[BLR95] A. Bouajjani, Y. Lakhnech, and R. Robbana. From duration calculus to linear hybrid automata. In *Proceedings of the Conference on Computer-Aided Verification*, Lecture Notes in Computer Science 939, pages 196–210. Springer-Verlag, 1995.

[BMP83] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. *Acta Informatica*, 20:207–226, 1983.

[BR95] A. Bouajjani and R. Robbana. Verifying ω-regular properties for subclasses of linear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 437–450. Springer-Verlag, 1995.

[Bry86] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[BW94] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete event systems. *IEEE Transactions on Automatic Control*, 39(2):329–342, February 1994.

[CDFV88] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, March 1988.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Workshop on Logic of Programs*, Lecture Notes in Computer Science 131. Springer-Verlag, 1981.

[Cer92] K. Cerāns. *Algorithmic Problems in Analysis of Real-time System Specifications.* PhD thesis, University of Latvia, 1992.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[CGH+93] E.M. Clarke, O. Grümberg, H. Hiraishi, S. Jha, D.E. Long, K.L. McMillan, and L.A. Ness. Verification of the futurebus+ cache coherence protocol. In *Proceedings of the 11th Symposium on Computer Hardware Description Languages and their Applications*, 1993.

[Chu63] A. Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.

[CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.

[DGG93] D. Dams, O. Grumberg, and R. Gerth. Generation of reduced models for checking fragments of ctl. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 479–490. Springer-Verlag, 1993.

[DHW92] D. Dill, A. Hu, and H. Wong-Toi. Checking for language inclusion using simulation preorders. In *Proceedings of Third International Workshop on Computer-Aided Verification, July 1991*, volume 575, pages 255–265. Springer-Verlag, 1992. Lecture Notes in Computer Science Series.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 208–219. Springer-Verlag, 1996.

[EH85] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.

[EH86] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not never" revisited: on branching versus linear-time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[EL86] E.A. Emerson and C. Lei. Efficient model checking in fragments of the propositional μ-calculus. In *Proceedings of the First Annual Symposium on Logic in Computer Science*, pages 267–278. IEEE Computer Society Press, 1986.

[ES84] E.A. Emerson and A.P. Sistla. Deciding full branching time logic. *Information and Control*, 61:175–201, 1984.

[FS94] L. Fix. and F.B. Schneider. Hybrid verification by exploiting the environment. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 1–18. Springer-Verlag, 1994.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings of the 14th Annual Symposium on Theory of Computing*, pages 60–65. ACM Press, 1982.

[GKN94] X. Ge, W. Kohn, and A. Nerode. Algorithms for chattering approximations to relaxed optimal controls. Technical Report 94-55, Mathematical Sciences Institute at Cornell University, Ithaca, New York, 1994.

[GKP89] R. Graham, D. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, 1989.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the Seventh Annual Symposium on Principles of Programming Languages*, pages 163–173. ACM Press, 1980.

[GR88] C.H. Golaszewski and P.J. Ramadge. On the control of real-time discrete event systems. In *Proc. 1989 Conference on Informational Sciences and Systems*, pages 98–102, 1988.

[GSSL94] R. Gawlick, R. Segala, J.F. Sogaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. In S. Abiteboul and E. Shamir, editors, *ICALP 94: Automata, Languages, and Programming*, Lecture Notes in Computer Science 820, pages 166–177. Springer-Verlag, 1994.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[HH95] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 225–238. Springer-Verlag, 1995.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36rd Annual*

Symposium on Foundations of Computer Science, pages 453–462. IEEE Computer Society Press, 1995.

[HHW95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.

[HK94] T.A. Henzinger and P.W. Kopke. Undecidability results for hybrid systems. Technical Report CSD-TR-95-1483, Cornell University, 1994. Presented at the Workshop on Hybrid Systems and Autonomous Control (Ithaca, NY).

[HK96] T.A. Henzinger and P.W. Kopke. State equivalences for rectangular hybrid automata. In U. Montanari, editor, *CONCUR 96: Concurrency Theory*, Lecture Notes in Computer Science. Springer-Verlag, 1996.

[HKPV] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*. To appear.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the 27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.

[HMP92] T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *ICALP 92: Automata, Languages, and Programming*, Lecture Notes in Computer Science 623, pages 545–558. Springer-Verlag, 1992.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[Ho95] P.-H. Ho. *Automatic Analysis of Hybrid Systems*. PhD thesis, Cornell University, 1995.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–580, 1969.

[Hoo91] J. Hooman. *Specification and Compositional Verification of Real-time Systems*. PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 1991.

[Hoo94] J. Hooman. Correctness of real time systems by construction. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 19–40. Springer-Verlag, 1994.

[HPV94] T.A. Henzinger, A. Puri, and P. Varaiya. Clock transformation of hybrid automata with rectangular differential inclusions. Presented at the Workshop on Hybrid Systems and Autonomous Control (Ithaca, NY), 1994.

[HRP94a] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. In B. LeCharlier, editor, *SAS 94: Static Analysis Symposium*, Lecture Notes in Computer Science 864, pages 223–237. Springer-Verlag, 1994.

[HRP94b] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. In B. LeCharlier, editor, *SAS 94: Static Analysis Symposium*, Lecture Notes in Computer Science 864, pages 223–237. Springer-Verlag, 1994.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, 1979.

[HW92a] G. Hoffmann and H. Wong-Toi. The input-output control of real-time discrete event systems. In *Proceedings of the 1992 IEEE Real-Time Systems Symposium*, pages 256–265, Phoenix, AZ, December 1992.

[HW92b] G. Hoffmann and H. Wong-Toi. Symbolic synthesis of supervisory controllers. In *Proceedings of the 1992 American Control Conference*, pages 2789–2793, Chicago, IL, June 1992.

[HW95] P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 381–394. Springer-Verlag, 1995.

[Isa65] R. Isaacs. *Differential Games; a mathematical theory with applications to warfare and pursuit, control and optimization.* Wiley, 1965.

[KdR85] R. Koymans and W.-P. de Roever. Examples of a real-time temporal specification. In B.D. Denvir, W.T. Harwood, M.I. Jackson, and M.J.

Wray, editors, *The Analysis of Concurrent Systems*, Lecture Notes in Computer Science 207, pages 231–252. Springer-Verlag, 1985.

[KNR96] W. Kohn, A. Nerode, and J. Remmel. Hybrid systems as Finsler manifolds: finite state control as approximation to connections. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066. Springer-Verlag, 1996.

[KNRY95] W. Kohn, A. Nerode, J. Remmel, and A. Yakhnis. Viability in hybrid systems. *Theoretical Computer Science*, 138(1):141–168, 1995.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 179–208. Springer-Verlag, 1993.

[Kup95] O. Kupferman. *Model Checking for Branching-time Temporal Logics*. PhD thesis, The Technion, 1995.

[KVdR83] R. Koymans, J. Vytopil, and W.-P. de Roever. Real-time programming and asynchronous message passing. In *Proceedings of the Second Annual Symposium on Principles of Distributed Computing*, pages 187–197. ACM Press, 1983.

[Lam80] L. Lamport. "Sometimes" is sometimes "not never"—on the temporal logic of programs. In *Proceedings of the 7th Annual Symposium on Principles of Programming Languages*, pages 174–185. ACM Press, 1980.

[Lam91] L. Lamport. The Temporal Logic of Actions. Technical Report 79, DEC Systems Research Center, Palo Alto, California, 1991.

[Lam93] L. Lamport. Hybrid systems in TLA+. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736. Springer-Verlag, 1993.

[LPY95] K.G. Larsen, P. Pettersson, and W. Yi. Compositional and symbolic model checking of real-time systems. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 76–87. IEEE Computer Society Press, 1995.

[LPZ85] O. Lichtenstein, A. Pnueli, and L.D. Zuck. The glory of the past. In R. Parikh, editor, *Logics of Programs*, Lecture Notes in Computer Science 193, pages 196–218. Springer-Verlag, 1985.

[LSVW96] N.A. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O Automata. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 496–510. Springer-Verlag, 1996.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the Sixth Annual Symposium on Principles of Distributed Computing*, pages 137–151. ACM Press, 1987.

[LV92] N.A. Lynch and F. Vaandrager. Forward and backward simulations for timing-based systems. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 397–446. Springer-Verlag, 1992.

[LV95] N. Lynch and F. Vaandrager. Forward and backward simulations part i: Untimed systems. *Information and Computation*, 121(2):214–233, 1995.

[McM93] K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem.* Kluwer Academic Publishers, 1993.

[McN93] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65:149–184, 1993.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Second International Joint Conference on Artificial Intelligence*, pages 481–489. The British Computer Society, 1971.

[Moo90] C. Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):346–358, May 1990.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, 1992.

[MP93a] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 194–209. Springer-Verlag, 1993.

[MP93b] Z. Manna and A. Pnueli. Models for reactivity. *Acta Informatica*, 30(2):609–678, 1993.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS 95: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 229–242. Springer-Verlag, 1995.

[MV94] J. McManis and P. Varaiya. Suspension automata: a decidable class of hybrid automata. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 105–117. Springer-Verlag, 1994.

[NK93a] A. Nerode and W. Kohn. Models for hybrid systems: automata, topologies, controllability, observability. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 317–356. Springer-Verlag, 1993.

[NK93b] A. Nerode and W. Kohn. Multiple-agent hybrid control architecture. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 297–316. Springer-Verlag, 1993.

[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 149–178. Springer-Verlag, 1993.

[OG76] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6(4):319–340, 1976.

[OSY94] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 81–94. Springer-Verlag, 1994.

[OW89] J.S. Ostroff and W.M. Wonham. A framework for real-time discrete event control. *IEEE Transactions on Automatic Control*, 35(4):386–397, April 1989.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *5th GI-Conference on Theoretical Computer Science*, Lecture Notes in Computer Science 104, pages 167–183. Springer-Verlag, 1981.

[PBGM62] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mischenko. *The Mathematical Theory of Optimal Processes*. Wiley, 1962.

[PBV96] A. Puri, V. Borkar, and P. Varaiya. ε-approximation of differential inclusions. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 362–376. Springer-Verlag, 1996.

[PdR82] A. Pnueli and W.-P. de Roever. Rendez-vous with Ada: a proof-theoretical view. In *Proceedings of the SIGPLAN AdaTEC Conference on Ada*, pages 129–137. ACM Press, 1982.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13(1):45–60, 1981.

[PR89a] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th Annual Symposium on Principles of Programming Languages*, pages 179–190. ACM Press, 1989.

[PR89b] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *ICALP 89: Automata, Languages, and Programming*, Lecture Notes in Computer Science 372, pages 652–671. Springer-Verlag, 1989.

[PR90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1990.

[Pra81] V. Pratt. A decidable μ-calculus. In *Proceedings of the 22th Annual Symposium on Foundations of Computer Science*, pages 421–427. IEEE Computer Society Press, 1981.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 95–104. Springer-Verlag, 1994.

[Rab72] M. Rabin. *Automata on Infinite Objects and Church's Problem.* Regional Conference Series in Mathematics 13. American Mathematical Society, 1972.

[Ram89] P. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by büchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, January 1989.

[RW87a] P. Ramadge and W. Wonham. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization*, 25(5):1202–1218, September 1987.

[RW87b] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, January 1987.

[RW89] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[SBM92] F.B. Schneider, B. Bloom, and K. Marzullo. Putting time into proof outlines. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 618–639. Springer-Verlag, 1992.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[SE89] R.S. Streett and E.A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81:249–264, 1989.

[Sie95] M. Siegel. A refinement theory that supports both "Decrease of Nondeterminism" and "Increase of Parallelism". In I. Lee and S. Smolka, editors, *CONCUR 95: Concurrency Theory*, Lecture Notes in Computer Science 962, pages 378–392. Springer-Verlag, 1995.

[Son81] E. Sontag. Nonlinear regulation: the piecewise linear approach. *IEEE Transactions on Automatic Control*, 26:346–358, 1981.

[Thi95] J.G. Thistle. On control of systems modelled as deterministic rabin automata. *Discrete Event Dynamical Systems: Theory and Application*, 5(4):357–81, September 1995.

[TW94a] J.G. Thistle and W.M. Wonham. Control of infinite behavior of finite automata. *SIAM Journal on Control and Optimization*, 32(4):1057–97, July 1994.

[TW94b] J.G. Thistle and W.M. Wonham. Supervision of infinite behavior of discrete-event systems. *SIAM Journal on Control and Optimization*, 32(4):1098–1113, July 1994.

[Var95] M. Vardi. An automata-theoretic approach to fair realizability and synthesis. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939. Springer-Verlag, 1995.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Annual Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society Press, 1986.

[WD90] H. Wong-Toi and D.L. Dill. Synthesizing processes and schedulers from temporal specifications. In R.P. Kurshan and E.M. Clarke, editors, *CAV 90: Computer-aided Verification*, Lecture Notes in Computer Science 531, pages 272–281. Springer-Verlag, 1990.

[WH91] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems (extended abstract). In *Proceedings of 30th IEEE Conference on Decision and Control*, pages 1527–1528, Brighton, England, December 1991.

[Wil94] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 694–715. Springer-Verlag, 1994.

[Wol82] P. Wolper. *Synthesis of Communicating Processes from Temporal-Logic Specifications*. PhD thesis, Stanford University, 1982.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.

[Won94] Howard Wong-Toi. *Symbolic Approximations for Verifying Real-Time Systems*. PhD thesis, Department of Computer Science, Stanford University, CA, December 1994.

[WR87] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, May 1987.

[WZ92] Henri B. Weinberg and Lenore D. Zuck. Timed ethernet: Real-time formal specification of ethernet. In *Proceedings of Third International Conference on Concurrency Theory, CONCUR '92*, pages 370–385, Stony Brook, NY, August 1992. Springer-Verlag. Lecture Notes in Computer Science 630.

[ZHR91] C. Zhou, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.