Document d'Habilitation à Diriger des Recherches

Université Joseph Fourier

Systèmes Discrets, Temporisés et Hybrides

Oded Maler

May 30, 2000

Chapter 1

Prologue

Ce document fait le bilan de mes recherches de 1986 à ce jour. J'y présente l'esprit et le contexte de mes travaux et mes publications. J'ai essayé de faire un compromis entre la présentation chronologique (plus simple pour l'auteur) et la présentation thématique (qui fait sens pour le lecteur). Ainsi, chaque section décrit une direction de recherche qui correspond à un ou plusieurs articles, localement ordonnés dans le temps. Chacune comporte une liste de références importantes suivie par une liste de mes propres publications ; le texte de celles-ci se trouve en annexe. Comme la plupart de mes articles contiennent des introductions longues et assez détaillées, j'ai évité, dans la mesure du possible, de répéter ces détails techniques dans le texte et j'ai tenté d'en illustrer, à l'aide d'exemples, les idées principales.

This document summarizes most of my research work from 1986 until these days. In the text that follows I give the general flavor and context of the various results I obtained and the papers I have written. In this presentation I tried the find a compromise between presenting my work in a chronological evolutionary order (this is easier for the writer) or grouping it on a thematic basis (which makes more sense for the reader). Consequently each section discusses a research direction which corresponds to one or more papers, which are locally ordered in time. A list of major references appears at the end of each section, followed by a list of those of my own papers on the subject which appear at the appendix. Since most of my papers contain lengthy introductions and definitions, I tried to avoid as much as possible technical details in the text, and illustrate the main ideas using examples.

Chapter 2

State Identification for ω-Automata

When I started my thesis at the Weizmann Institute I was interested in the domain of *inductive inference* which is concerned with the following problem: you have a black box which realizes a function f, you are given elements of the graph of f, i.e. pairs of the form $(x, f(x))$ and the goal is to infer f. This problem is common to many disciplines and is treated under the names of *machine learning, system identification, statistical inference* and more. This process can serve as a naïve model of how science progresses by generalizing observations into theories and then modifying them according to new observations which refute them. In fact, my first thesis advisor, Udi Shapiro, was attracted to this domain by Popper's philosophy of science. My first introduction to the topic was through a survey by Angluin and Smith which epmphasized the computability point of view. If we apply standard computability criteria, the inductive inference problem is either unsolvable or solvable in a trivial sense: if the sample points are only a subset of the domain of f (which is always the case when the domain is infinite), any result produced by a learning algorithm based on such a sample can be wrong and can be refuted by a new example. If the domain is finite and all the points are given, then there is nothing to compute, just to memorize (there is, of course the question of a compact description of f).

One approach to give meaning to the problem was suggested by Gold under the name of *identification in the limit*. In this setting you have a class F of functions and an oracle which gives examples for some $f \in F$ in a sequential manner. An inductive inference algorithm reads these examples and produces a sequence $f_1, f_2, \ldots$ of candidate descriptions of the function. A class F of functions is learnable in this sense if there is an algorithm which eventually converges to f for every $f \in F$. Technically this means that there is some finite time instance k such that for every $i > k$, $f_i = f$. Gold suggested the following algorithm, *identification by enumeration*, for identifying in the limit a large class of functions. Let F be a class of functions from some domain into $\{0, 1\}$ which is recursively enumerable, i.e. it is possible to generate a sequence of function descriptions $\varphi_0, \varphi_1 \ldots$ such the for every $f \in F$ there exists j such that $f = \varphi_j$.

Algorithm 1 (Identification by Enumeration)

$i := j := 0$; $f_0 := \varphi_0$; $V := \emptyset$;
repeat
  $V := V \cup \{$*next-example*$\}$;
  while φ_j *contradicts* V **do**
    $j := j + 1$;
  end
  $f_i := \varphi_j$;
forever

Note that in order to evaluate the candidate functions on the sample, the class F must be contained in the class of recursive functions. The algorithm shows, for example, that regular languages can be identified in the limit: you enumerate automata or regular expressions, test membership of the sample points and you are guaranteed to converge eventually. Gold's results opened at the time a large field of research, which is still practiced in some parts of the world where recursion theorists are proving new results beyond what less-motivated persons like myself can understand in bounded time.

Another approach to the problem, which is common in statistical inference or approximation theory, is to define a *metric* on the function space and try to quantify how close a learning algorithm can get to the real function using a given number of examples. A paper by Valiant, published just few years before I started my investigations, suggested a complexity-oriented definition of approximate learnability, based on how many examples you need in order to decrease the expected probabilistic distance between your guesses and the function. Valiant showed that some classes of Boolean functions such as k-DNF are polynomially learnable in his sense. The paper was very influential and generated the *computational learning* community where complexity theoreticians and probabilists could find a new playground for their lower and upper bounds. For some time I was attracted to this direction, and even wrote a lengthy thesis proposal around these ideas, but after sometime (with a good advice of Adi Shamir) I realized that complexity will not be my domain.

Among the other directions in inductive inference, the one that attracted me the most was related to my favorite mathematical objects, finite automata. The history of the topic dates back to an old paper by Moore, published before I was born, which treats the following situation: suppose you find a black box with various buttons and lights and you start pushing buttons and observing the resulting lights until you construct a finite-state model of the unknown machine which allows you to predict the outcome of any further input sequence. This is something that we do implicitly when we learn how to operate a new technological product without reading the manual. Moore's work is closely-related to one of the fundamental theorems in the theory of automata, the Myhill-Nerode theorem, which I will explain below.

Formal languages are subsets of Σ^*, the set of all finite sequences over an alphabet Σ. There are various ways to describe a given language L. For example, one can write a logical formula satisfied exactly by the sequences belonging to L. Another possible description of L is to give a transition system $\mathcal{A}$ which *recognizes* it, i.e. $\mathcal{A}$

accepts exactly the elements of L. There are many different recognizers for the same language as there are many different programs which compute the same function. The Myhill-Nerode theorem shows that for regular languages there is a *canonical* characterization which is isomorphic to the minimal automaton recognizing L. This automaton is not only minimal in the *combinatorial* sense (has the least number of states) but is also minimal in the *algebraic* sense, i.e. it is homomoprhic to any other automaton recognizing the same language.

An equivalence relation $\sim$ on Σ^* is a *right-congruence* if $u \sim v$ implies $uw \sim vw$ for all $u, v, w \in \Sigma^*$. We will denote by $[u] = \{v : v \in \Sigma^* \text{ and } v \sim u\}$ the equivalence class containing the word u. With every language L we can associate the following relation, called the *syntactic right-congruence* of L:

$$u \sim_L v \text{ iff } \forall w \in \Sigma^* (uw \in L \iff vw \in L) \tag{2.1}$$

This definition declares two sequences as equivalent if after reading either one of them as a prefix, the same suffixes will be accepted. Hence every class $[u]$ represents a set of *past histories* which are indistinguishable by any future. This is, in fact, the notion of a *state* in the modern state-space approach to dynamical systems. From $\sim_L$ you can construct immediately an automaton with the initial state $[\varepsilon]$ (the right-congruence class of the empty word) and the transition function defined as $\delta([u], a) = [ua]$ for every $a \in \Sigma$. This is the minimal automaton which accepts L. The theorem then gives an alternative characterization of regular sets as those which can be written as unions of equivalence classes of a right-congruence relation of a finite index.

Based on this theorem, Gold suggested an algorithm for learning regular languages from examples. When some oracle tells us whether or not a sequence w is in L, it also tells us, for every factorization $uv = w$, whether or not the prefix u accepts the suffix v. The observations can then be arranged in a table where both rows and columns are indexed by elements of Σ^*, and every entry (u, v) in the table is marked by $+$ or $-$ according to whether $uv \in L$. According to Myhill-Nerode theorem, for every regular language the table will have only *finitely many* different rows, each corresponding to one right-congruence class. It is not hard to see that there will be also finitely many different columns corresponding to left-congruence classes. For illustration, consider the language accepted by the automaton of Figure 2.1. Using a set of negative examples $\{\varepsilon, a, b, aa, ba, bb, aaa, aab, abab, abb, abba\}$ and positive examples $\{ab, aba, abaa, abbb\}$ we can construct Table 2.1 which corresponds to the automaton. In fact, the algorithm finds a kind of *spanning tree* for the transition graph of the automaton.

If it is known a-priori that the automaton has at most n states, it suffices to put all sequences up to length n+1 as rows and columns of the table in order to identify all the states of the automaton. Unfortunately the number of such sequences is exponential in n (it is, by the way, questionable whether the number of states is an appropriate complexity measure for an automaton, but one should adopt some rules of some game, at least for some time).

A solution to the complexity problem was proposed by Dana Angluin in a paper published at that time, using the following setting: the learning algorithm can ask

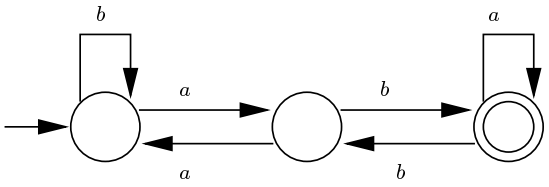

Figure 2.1: A 3-state automaton.

	ε	a	b
ε	$-$	$-$	$-$
a	$-$	$-$	$+$
ab	$+$	$+$	$-$
b	$-$	$-$	$-$
aa	$-$	$-$	$-$
aba	$+$	$+$	$-$
abb	$-$	$-$	$+$

Table 2.1: The observation table for the automaton. The states of the canonical automaton are $[\varepsilon]$, $[a]$ and $[ab]$ and the rest of the "candidate" are equivalent to those.

queries concerning the membership of sequences in the language. When the algorithm has a consistent table, it suggests the corresponding automaton as a conjecture. If this is correct, the algorithm terminates. Otherwise, another oracle supplies a *counter-example*, that is, a sequence which contradicts the conjecture. Angluin has showed that the algorithm terminates after polynomially many membership queries and counter-examples. This paper was the first (and probably the last) one I read and understood completely.

Around that time I deviated a lot from the original motivation of my first thesis supervisor, and he decided that he cannot continue to play this role – I don't blame him for that. Consequently I was looking for a topic and a supervisor, after more than a year of studentship. One day in a seminar I heard Amir Pnueli, whom I already tried to convince before to take me under his responsibility, give one of his talks about temporal logic and verification. Unlike other talks I heard from him, which were more based on the deductive approach, this one was about algorithmic verification and the specification formalism was not logic but some kind of ω-automata, automata accepting infinite sequences. After the talk I asked him whether he will be interested in using learning technology to extract such specifications from users who cannot express them formally but rather give examples of bad and good behaviors of the intended system (scenarios). Pnueli said politely that he will be very interested.

In our first meeting on the topic I raised an immediate problem: since ω-automata accept sets of *infinite* sequences, how can such examples be presented to the learning algorithm? Pnueli suggested to look at ultimately-periodic sequences,

	a^ω	b^ω	ba^ω	ab^ω	$(ab)^\omega$	...
ε	+	−	+	−	−	...
a	+	−	+	−	−	...
b	+	−	+	−	−	...
ab	+	−	+	−	−	...
...	+	−	+	−	−	...

Table 2.2: The observation table for the language $(a+b)^*a^\omega$. The corresponding 1-state trivial automaton cannot accept the language.

i.e. those of the form uv^ω where u and v are finite sequences. Such sequences play an important role in Σ^ω, somewhat analogous to the role of rational numbers among the reals. Sometime later, when I came with some more ideas about new theories to be invented and new theorems to be proved, I managed, for the first and last time, to extract from Pnueli an angry reaction: why do you keep on suggesting "this could be proved, that should be done" why don't you just *do* it? This was the most important lesson in my personal career. I sat down to prove my first, rather trivial, lemma (at the age of 30!): if two ω-regular languages are different from each other, then there is an ultimately-periodic word (whose length depends on the number of states) which distinguishes between them. After that, Pnueli accepted to be my supervisor based on a new research proposal [**P01**].

A month later Moshe Vardi came for a summer visit and gave a short course on ω-automata and their role in verification. The course influenced me and my friend Muli Safra to look closer at these creatures. For Safra this led later to his famous result concerning a new determinization construction for ω-automata. I started to work on extending Angluin's algorithm to learn ω-languages. In fact, the extension seemed to be rather straightforward: in the observation table you put ultimately-periodic words at the columns and try to identify states as before. After some experimentation I observed a strange phenomenon: for some languages the table construction terminated but the resulting automaton could not accept the language (see Table 2.2).

After some time I realized that in ω-languages the correspondence between right-congruence classes and states of a minimal accepting automaton does not hold (later I learned that this fact was observed already by Trakhtenbrot). States have two roles in such automata: one is to memorize finite prefixes and the other is to separate infinite suffixes by sending them to different limit cycles. Fortunately, I came across a paper by Ludwig Staiger who showed, using complicated topological arguments, that for some non-trivial sub-class of languages, an ω-variant of Myhill-Nerode theorem holds and all states can be characterized by the ω-words they accept. This way I could have my first result, a polynomial learning algorithm for this sub-class of ω-regular languages [**P02**]. As a byproduct of this work I learned some of the topology of infinite sequences (the Cantor set) – quite a funny set for learning topology on.

The next natural step, the generalization of the algorithm to the whole class of ω-regular sets, faced a major obstacle, the absence of a unique minimal automaton. I found a paper of André Arnold who gave a canonical characterization by means

of a two-sided congruence. The essence of his definition was to consider two words equivalent if they can replace each other not only inside a finite prefix but also in the period of an ultimately-periodic word:

$$u \cong_L v \text{ iff } \forall x, y, z \in \Sigma^* \begin{array}{l} (xuyz^\omega \in L \iff xvyz^\omega \in L) \wedge \\ (x(yuz)^\omega \in L \iff x(yvz)^\omega \in L) \end{array} \tag{2.2}$$

Two-sided congruences do not correspond to *states* of the automaton but rather to elements of its *transformation monoid*, a much larger object. After some time I found myself in a two-month visit to the church of the free monoid, the LITP laboratory in Paris. This was one of the many interesting trans-cultural experiences I had while observing many different scientific communities. Any group of talented people can do an elaborate body of work, with terminologies, seminars, papers and journals, and give its members a feeling of being in the center of the world, while, at the same time, be completely ignored elsewhere. Contrary to my naïve hopes, nobody there "solved" my problem, but I made some good friends, learned to appreciate rotten cheese and had the privilege to meet a person in the caliber of Marcel-Paul Schützenberger, whose advise on mathematics, science and philosophy I tried to follow whenever I could.

Coming back to Weizmann I suggested to Pnueli that I spend the remaining two years of my thesis attacking the problem of minimization of ω-automata. I was ready to bang my head against the wall, even at the price of not finishing my thesis. Pnueli objected and, in fact, only few years later, together with Ludwig Staiger, I found a somewhat satisfactory solution to that problem [**P03**]. Instead, Pnueli managed to convince me to look at the Krohn-Rhodes decomposition theorem and to try to apply it to translating automata into temporal logic.

References

D. Angluin, Learning Regular Sets from Queries and Counter-Examples, *Information and Computation* 75, 87-106, 1987.

D. Angluin and C.H. Smith, Inductive Inference: Theory and Methods, *ACM Computing Surveys* 15, 237-269, 1984.

A. Arnold, A Syntactic Congruence for Rational ω-Languages, *Theoretical Computer Science* 39, 333-335, 1985.

E.M. Gold, Language Identification in the Limit, *Information and Control* 10, 447-474, 1967.

E.M. Gold, System Identification via State Characterization, *Automatica* 8, 621-636, 1972.

D.E. Moore, Gedanken Experiments on Sequential Machines, in: Shannon and McCarthy (Eds.), *Automata Studies*, 129-153, Princeton University Press, Princeton, 1956.

S. Safra, On the complexity of ω-automata, *Proc. 29th FOCS*, 319-327, 1988.

L. Staiger, Finite-State ω-Languages, *J. of Computer and System Sciences* 27, 434-448, 1983.

W. Thomas, Automata on Infinite Objects, in J. Van Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, 133-191, Elsevier, Amsterdam, 1990.

B.A. Trakhtenbrot, Finite automata and monadic second order logic, *Siberian Math. J.* 3, 103–131, 1962.
L.G. Valiant, A Theory of the Learnable, *Comm. of the ACM* 27, 1134-1142, 1984.
M.Y. Vardi and P. Wolper, Automata-Theoretic Techniques in Modal Logics of Programs, *J. of Computer and Systems Science* 32, 183-221, 1986.

My Papers

[**P01**]
O. Maler, *Detecting Regularities in Computations*, Unpublished Ph.D. Proposal, Weizmann Institute, 1987.
[**P02**]
O. Maler and A. Pnueli, On the Learnability of Infinitary Regular Sets, *Information and Computation*, 118, 316-326, 1995. [results obtained in 1988]
[**P03**]
O. Maler and L. Staiger, On Syntactic Congruences for ω-Languages, *Theoretical Computer Science*, 183, 93-112, 1997. [results obtained in 1992]

Chapter 3

The Krohn-Rhodes Decomposition Theorem

The Krohn-Rhodes primary decomposition theorem was considered in the past as one of the cornerstones of automata theory. It is a theorem about constructing semigroups and monoids from simple building blocks. A semigroups $(S, \diamond)$ is algebraic structure consisting of a set S of elements closed under an associative operation $\diamond$. If you add an identity element e such that $a \diamond e = e \diamond a = a$ for every $a \in S$, you obtain a monoid. If you add the axiom of existence of an inverse a^{-1} for every element a, i.e. $a \diamond a^{-1} = e$, you obtain the much more rich, noble and famous structure, the group. Examples of monoids are:

- The non-negative numbers (integers, rationals or reals) under the operation of addition with 0 as identity.
- The set of all finite sequences over an alphabet A under the concatenation operation with the empty word as identity. This is called sometimes the *free monoid* generated by A and denoted by A^*.
- The set of all functions (or partial functions) of the from $f : Q \to Q$ for some set Q (transformations on Q) under the composition of functions. In particular any automaton defines a finite transformation monoid generated by the transformations induced by the letters of the alphabet.

In the same way as every abstract group is isomorphic to a group of permutations (bijective transformations), every abstract monoid is isomorphic to a monoid of transformations. The Jordan-Hölder theorem states the every finite group can be written as a wreath product of simple groups (those simple groups have been all catalogued). The Krohn-Rhodes theorem extends this results to monoids and it shows that in addition to the simple groups you need only one more type of monoid, the monoid U_2 whose operation table is

$\diamond$	e	a	b
e	e	a	b
a	a	a	b
b	b	a	b

in order to construct all monoids.

The automata-theoretic version of the theorem is based on the notion of a *cascade product* of automata. A cascade $\mathcal{B} = \mathcal{A}_1 \circ \mathcal{A}_2$ of two automata can be defined in two equivalent ways. Using automata without output, you let $\mathcal{A}_1 = (\Sigma, Q_1, \delta_1)$ and $\mathcal{A}_2 = (\Sigma \times Q_1, Q_2, \delta_2)$. The second automaton reads both the input and the state of the first automaton and $\mathcal{B} = (\Sigma, Q_1 \times Q_2, \delta)$ where

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, (q_1, a))).$$

This notion is generalized naturally to a cascade $\mathcal{A}_1 \circ \ldots \circ \mathcal{A}_n$ where the input alphabet of every $\mathcal{A}_i$ is $\Sigma \times Q_1 \times \ldots \times Q_{i-1}$. If you work with transducers ("Mealy machines"), the first automaton has Σ as an input alphabet and X_1 as output alphabet and every automaton $\mathcal{A}_i$ has X_{i-1} as input and X_i as output. These two versions of the cascade product are depicted in Figure 3.1. The initial motivation for such cascades comes from hardware where you want to build automata from simple components with a limited interaction between them. The theorem says that you can realize any automaton as a cascade while using only the two types of automata, illustrated in Figure 3.2:

- Permutation Automata: every letter induces a permutation of the states. The transformation monoid of such an automaton is a group.
- Reset Automata: every letter induces either an identity or a reset, i.e. it drives the automaton from any state into a single state.

An important sub-class of automata are those which are *counter-free*, i.e. there is no word which induces a permutation on a non-singleton subset of the states. Algebraically speaking, the transformation monoid of such automaton is group-free (it contains no non-trivial sub-groups). The Krohn-Rhodes theorem states that such automata admit a decomposition which uses *only* reset automata. The languages accepted by counter-free automata are known as the *star-free languages*, due to a theorem by Schützenberger showing that these are exactly the languages which correspond to regular expressions without the Kleene star (but with intersection and complementation). Another property of these languages is that they are definable in the *first-order* theory of order (unlike the whole regular sets which need a monadic *second-order* logic). These languages and their characterization were the subject of a monograph by Minsky and Papert. Pnueli's interest in these language came from yet another characterization of these languages (due to a result by Kamp who showed that "tense logic" has the same expressive power as the first-order theory of order): they are exactly those languages definable by propositional linear time temporal logic. The only known procedure for translating counter-free automata into temporal logic formulae was indirect (via first-order logic) and its complexity was non-elementary. His idea was to try to find a better translation by going directly from the Krohn-Rhodes decomposition into temporal logic.

This final result by itself, took the least effort. By studying a construction by Albert Meyer for translating cascades of reset automata into star-free regular expressions, I could invent a similar translation into past temporal logic. The crucial observation is that in reset automata you can characterize the set of all sequences

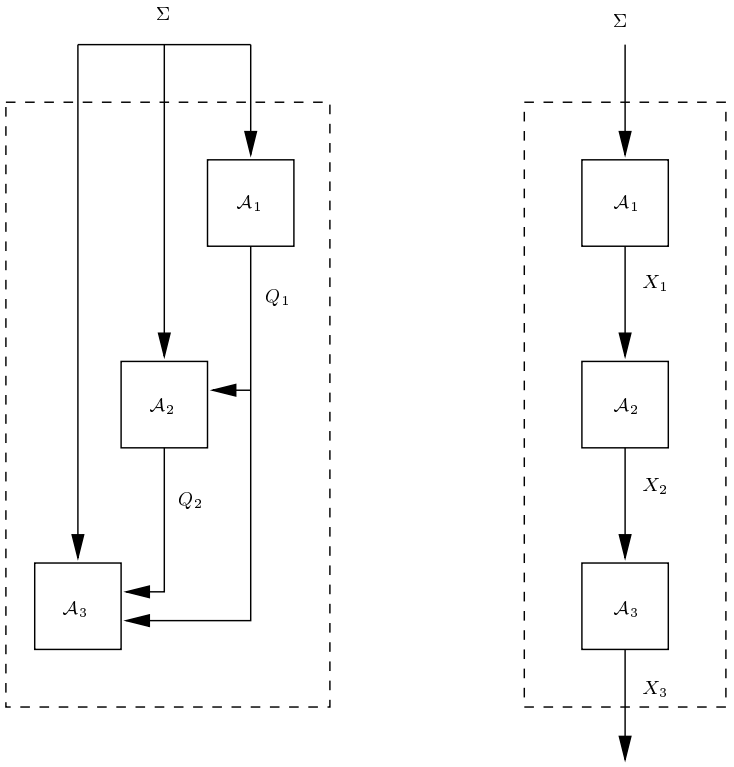

Figure 3.1: A cascade product of 3 automata using the state-as-input convention (left) and the input-output convention (right).

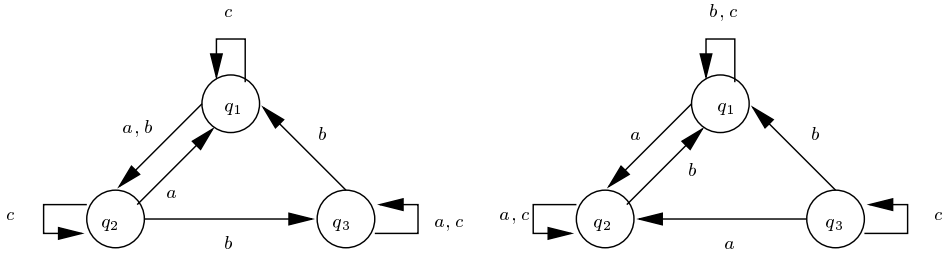

Figure 3.2: A permutation automaton (left) where the letters a, b and c induce the permutations (213), (231) and (123) respectively. A reset automaton where those letters induce the transformations (222), (111) and (123). Note that the generated monoid is isomorphic to the abstract monoid U_2.

which lead to a state by a simple formula involving the *Since* operator. For example, the sequences leading to q_1 in the reset automaton of Figure 3.2 can be described by the formula $\neg a \mathcal{S} b$, saying that we have not left state q_1 (by any letter which is a reset to another state) since we have entered it (by a letter which is a reset to q_1). In order to construct a formula describing words which go, say, to a state in the automaton $\mathcal{A}_2$ in the cascade, you do the same thing while replacing every letter in the input alphabet of the form $(a, q) \in \Sigma \times Q_1$ by the formula $a \wedge \ominus \varphi_q$ where φ_q is the since-formula for q and $\ominus$ is the *Previously* operator. This way states at the second level will be characterized by formulae of the form

$$\neg(a \wedge \ominus(\neg b \mathcal{S} c)) \mathcal{S} \neg(d \wedge \ominus(\neg e \mathcal{S} f)).$$

The maximal nesting of *Since* and *Previously* operators in the formula is the depth of the cascade.

The hard part was to understand how the decomposition goes on and determine its complexity, i.e. the relation between the size of the cascade and the size of the original automaton. Many of the papers and books on the subject were practically unreadable for me (a book by Ginzburg was an exception). I got an advice from Schützenberger: read about it in volume B of Eilenberg. The effort I invested in reading some chapters of this elegant and motivation-free[1] algebraic prose was enormous, having to translate every passage into my automata-theoretic world view. The Krohn-Rhodes theorem occupied 12 pages of the text under the title "the holonomy decomposition theorem". It took me several months to understand (and much more to *really* understand) but the effort was rewarding: I could convert the proof from monoids to automata (which is needed in order to do the translation to logic) and became one of the few persons who know how to take any automaton and decompose it. The algorithm, as described in my thesis, was later implemented by students from the group of Wolfgang Thomas. The construction was only exponential and consequently I could give an elementary algorithm to translate automata into past temporal logic. The result could be lifted almost immediately to translate ω-automata into formulae with future and past operators.

The last year of my thesis was devoted to proving my first and last lower-bound, a kind of a farewell to the complexity-dominated environment at Weizmann (no one is completely immune to social pressure). I wanted to prove that the decomposition algorithm was the best possible. At the end I found a family of automata (the "elevator", see Figure 3.3) for which the size of the decomposition grows exponentially with the size of the automaton. This tight bound bought me a ticket into FOCS, a major American "theory" conference where my paper was presented in the logic-and-automata ghetto – yet another anthropological experience.

In addition to getting acquainted with early papers on automata and with some of the algebraic culture, what I gained from working on the Krohn-Rhodes theory was an internalization of the conceptual scheme of *products of automata* as a useful metaphor for systems which interact with each other subject to some communication constraints. Since then I try to apply this scheme to many different problems. Looking back at that period I observe that although I was tuned during my thesis

[1] There is a mathematical motivation but no interest at all in the phenomenon of computation.

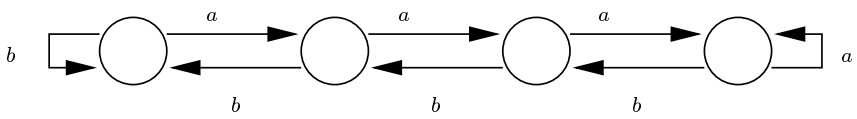

Figure 3.3: The elevator automaton, the hardest counter-free automaton to decompose.

to many new trends and fashions (see next section), and even got enthusiastic about some of them, I finally did my technical work around two topics, the Myhill-Nerode and Krohn-Rhodes theorems which were rather old (the second practically forgotten). I also noticed that I found computer science journal issues from the sixties much more interesting than those of the seventies and eighties. I worked most of the time in isolation, not taking part in existing communities and defining myself as an automata-theorist, not a very popular profession.

References

S. Eilenberg, *Automata, Languages and Machines, Vol. B*, Academic Press, New-York, 1976.
D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, On the Temporal Analysis of Fairness, *Proc. 7th POPL*, 163-173, 1980.
A. Ginzburg, *Algebraic Theory of Automata*, Academic Press, New-York, 1968.
J.A.W. Kamp, *Tense Logic and the Theory of Order*, UCLA, 1968.
K. Krohn and J.L. Rhodes, Algebraic Theory of Machines, I Principles of Finite Semigroups and Machines, *Transactions of the American Mathematical Society* 116, 450-464, 1965.
R. McNaughton and S. Papert, *Counter Free Automata*, MIT Press, Cambridge, MA, 1971.
A.R. Meyer, A Note on Star-Free Events, *Journal of the ACM* 16, 220-225, 1969.
A. Pnueli, The Temporal Logic of Programs, *Proc. 18th FOCS*, 46-57, 1977.
M.P. Schützenberger, On Finite Monoids Having only Trivial Subgroups, *Information and Control* 8, 190-194, 1965.

My Papers

[**P04**]
O. Maler and A. Pnueli, Tight Bounds on the Complexity of Cascaded Decomposition of Automata, *Proc. 31st FOCS*, 672-682, 1990.

Chapter 4

Hybrid Systems I

During the 80s I have seen several waves of enthusiasm concerning AI. The first wave was around "expert systems", a topic on which I worked during my masters thesis at the faculty of management of Tel-Aviv university, and also as a consultant. It was followed by a related big noise about logic programming and the "Japanese" fifth generation project, which was my initial (and false) motivation to come to Weizmann. After that there was the connectionist neural network movement which attracted also biologists, physicists and cognitive scientists. Toward the end of my thesis I stumbled upon some papers by Rod Brooks from MIT. It was yet another argument against *good old-fashioned symbolic AI* whose goal was to imitate the higher-level reasoning capabilities of humans. Brooks argued that reasoning should be constructed from the bottom up, starting with basic senso-motoric behaviors and combining them while creating higher and higher capabilities. The fact that one needs to take care of senso-motoric loops is far from being a new discovery for control theorists. Many other aspects of this "behavior-based" approach can be criticized from the points of view of software engineering or philosophy in general, yet I think, it was a positive development in the AI context.

What attracted me to this approach were the models used by Brooks in his "subsumption architecture": these were automata (finite-state machines, as engineers like to call them) augmented with sensors, effectors and counters. Some immediate questions came to my mind due to my mild exposure to semantics and verification: What is the appropriate model for describing the behavior of such a system? How can you prove anything about the behaviors of such a system in a given environment? For me at that time, differential equations and the rest of continuous mathematics were nothing but bad memories from my freshman years, but Pnueli, who started his career as an applied mathematician knew all about these horrible things and the idea to extend verification methodology beyond computers has crosed his mind in the past. Together we started to play with some toy examples such as models of mobile robots moving in corridors and even submitted a research proposal on verifying properties of robots. I went to a workshop of the behavior-based robotics community and tried to preach verification to them. The inter-cultural gaps were a good preparation for later contacts with control theorists (at least the AI guys had no favorite mathematics to defend).

Due to a strange combination of reasons and coincidences, I arrived at IRISA

Rennes for a post-doc. My agenda included the investigation of those discrete-continuous systems as well as other ambitious and dreamy projects. Some of the results I obtained there are described in Section 10, however the most impactful outcome was the paper "from timed to hybrid systems" [**P05**] written mostly by Pnueli and co-authored by Zohar Manna and myself. This was the first formal model of hybrid systems coming from the verification community. The model, called *phase transition systems*, contained both discrete and continuous variables subject to transition relations and differential equations (activities). In addition, the model contained all the Manna-Pnueli machinery (variables, primed variables, super-dense sequences, fairness and justice) which is perhaps unavoidable for practical verification of large discrete systems, but a bit of an overkill for studying a new phenomenon and for approaching other communities. This is why the simpler hybrid automata introduced later by Henzinger, Sifakis and others became more popular. It is fair to say that this paper, presented by Pnueli in 91 and published in 92, is among the main culprits for the computer science part of hybrid system research. This was the reason that Joseph Sifakis has offered me a second chance in the French system after I blew it in my unfocused post-doc in Rennes.

I moved to Grenoble few weeks before the first European meeting on hybrid systems in Lyngby. Since I have not really worked technically on hybrid systems, all I could produce in the remaining time was a sort of position paper, investigating the meaning and motivations for hybrid systems, the difference between classical "autistic" computability and real-time programming. The paper was not accepted to the proceedings (maybe someone wanted to protect my career) but looking at it today, after having learned few more things about control, real-time programming, and sociology of science, I wonder upon what knowledge did the author base his correct speculations [**P06**].

Coming back to Grenoble, I decided that in order to be fair to my employer, I need to have some technical results about hybrid systems (everybody and his sister seemed to be deciding, undeciding or model-checking linear hybrid automata). This led to the definition of PCD systems, dynamical systems with piecewise-constant derivatives (or if you want, deterministic linear hybrid automata without resets) which were closer in nature to continuous dynamical systems. Defining and studying these systems I had to backtrack to topics such as linear algebra and its relation to geometry which were taught at school while I was busy doing other things. I also had to resort to topological arguments (Jordan's theorem) and even reinvent (independently) a version of Poincaré maps. With the E-mail help of Pnueli and the temporal generosity of the Mediterranean chairman of CAV'93 program committee, I got a nice decidability result for hybrid systems with two continuous state variables and a first encounter with the verification community.

PCD systems consist of a partition of the space into a finite number of disjoint polyhedral sets (regions), in each of which the continuous variables have a fixed derivative. The trajectories of a PCD system consist of broken lines (see Figure 4.1). The reachablity problem is whether some point (or a set) is reachable from another point (or a set). The decidability of such problems on the plane is based on two observations:

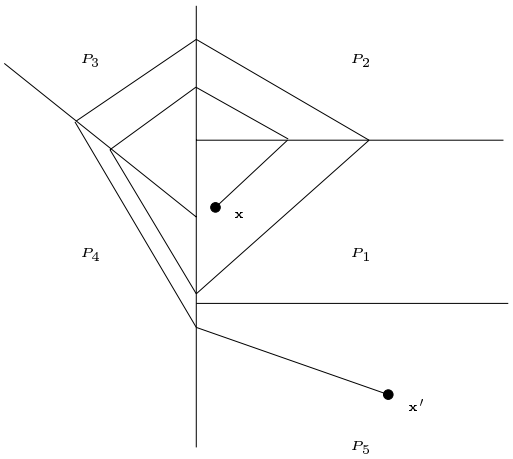

Figure 4.1: A PCD system with 5 regions and a trajectory from $\mathbf{x}$ to $\mathbf{x}'$ whose qualitative behavior can be written as $P_1P_2P_3P_4P_1P_2P_3P_4P_5$.

1. Trajectories in the plane have a limited complexity due to topological properties. Hence a trajectory which visits a region and then abandons it, cannot visit it again (see Figure 4.2). Since there are finitely meny regions, this implies that all trajectories are ultimately-periodic (in the qualitative sense) and settle in an ultimate spiral.

2. If a trajectory leaves a point on some edge and then returns to it, the new point is a linear function of the first point. Consequently one can compute the limit of every seqeunce of intersection points of a trajectory with an edge. This result holds in any dimension (see Figure 4.3).

Combining the two together we obtain the decidability result for systems in 2 dimensions. Note that this result is *not* based on finite bisimulation (a kind of homomorphism from the system to a finite-state automaton). If you want to find similarity between 2-dimensional PCD systems to some transition system, you should look at infinite-state systems whose reachability problem is decidable, e.g. the push-down automaton.

I tried to generalize this result to higher dimensions and to systems with resets (which break the topological properties). However, my colleague Ahmed Bouajjani showed me how a two-counter machine can be simulated by a PCD system with 6 variables and I did not know in what direction to continue this work. Fortunately, Evgeni Asarin, a Russian mathematician and computer scientist whom I already met in Rennes, passed in Grenoble, looked at the CAV paper and liked it. After some time he produced an undecidability result in 4 dimensions and then in 3. The result is based on encoding stack contents as real numbers and doing Push and Pop operations using PCD elements. We published the result in ICALP'94 and the combined results in a special issue of TCS [**P06**]. Later he showed that these simple hybrid systems are even more undecidable than we thought before [**P07**].

Theoretical computer scientists enjoy the privilege of being able to publish nega-

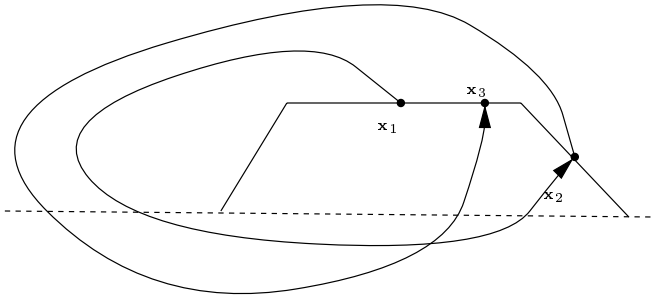

Figure 4.2: If a trajectory leaving a boundary of a region at point x_1 and later comes back at a point x_2 to the right of x_1, then it cannot visit anymore any point between x_1 and x_2.

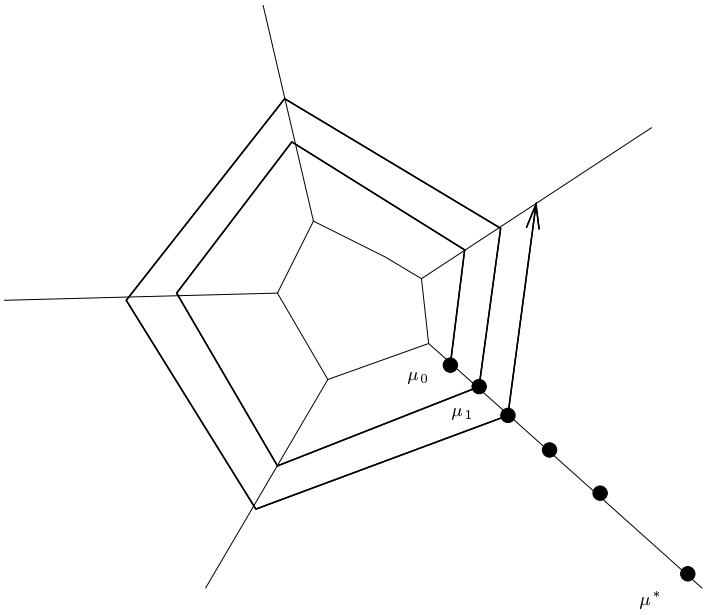

Figure 4.3: The sequence of intersection points of a trajectory with an edge can be written as $\mu_{i+1} = A\mu_i + B$ where A and B can be derived from the paramaters of the inequalites and the derivatives of the corresponding regions.

tive results about computability and complexity, but, nevertheless, I found the bottom line of these (and other) results rather depressing. If even for these piecewise-trivial systems we cannot solve basic verification problems, how will we face *real* continuous systems? The feedback from various control theorists and practitioners was not encouraging either. Consequently I became convinced that we are not asking the right questions and decided to put hybrid systems aside for a while, and concentrate in the more tractable class of timed systems.

References

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.

P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Eds.), *Hybrid Systems II*, LNCS 999, Springer, 1995.

R.A. Brooks, A Robust Layered Control System for Mobile Robots, *IEEE Journal of Robotics and Automation* RA-2, 14-23, 1986.

R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel, *Hybrid Systems*, LNCS 736, Springer, 1993.

My Papers

[**P05**]

O. Maler, Z. Manna, and A. Pnueli, From Timed to Hybrid Systems, In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg (Eds.), *Real-Time: Theory in Practice*, 447-484, LNCS 600, Springer, 1992.

[**P06**]

O. Maler: Hybrid Systems and Real-World Computations, *Workshop on Hybrid Systems*, Lyngby, 1992. [Unpublished]

[**P07**]

E. Asarin, O. Maler and A. Pnueli, Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives, *Theoretical Computer Science*, 138, 35-65, 1995. [Results obtained in 93-94]

[**P08**]

E. Asarin and O. Maler, Achilles and the Tortoise Climbing Up the Arithmetical Hierarchy, *Journal of Computers and Systems Science*, 57, 389-398, 1998. [Results obtained in 95]

Chapter 5

Timed Systems: Synthesis

In a very general form, the problem of synthesis can be formulated as follows. Given some relation $R \subseteq X \times Y$ find the set $X' = \{x : \exists y\ R(x,y)\}$ and a *function* $f : X' \to Y$ such that $R(x, f(x))$ for every $x \in X'$ (see Figure 5.1). In descriptive set theory this is called the *uniformization* problem. Of particular interest is the case where $X = A^\omega$ and $Y = B^\omega$ for some finite alphabets A and B and where the function $f : A^\omega \to B^\omega$ is sequential (causal, retrospective, non-anticipatory), i.e. there is a function $g : A^* \to B$ such that f can be defined as $f(\varepsilon) = \varepsilon$ and $f(ua) = f(u) \cdot g(ua)$. In this case we can interpret the whole setting as a two-person zero-sum game between players A and B, each makes its move by choosing from its action alphabet. Every instance of the game is an infinite sequence $a_1 b_1 a_2 b_2 \ldots$ in which B wins if the resulting sequence belongs to R. The function g can be seen as a *strategy* for B, telling it what to chose at every instant t based on the state of the game, which is identified by the finite prefix $a_1 b_1 a_2 b_2 \ldots a_t$.

Since our intuition is more used to the Eucildean space than to the set of infinite sequences, I will try to illustrate the relation to set theory using the more familiar sets $[0,1)$ and $[0,1)^2$. An instance of an infinite game between two players over binary alphabets is an infinite binary string $a_1 b_1 a_2 b_2 \ldots$ which can be transformed via positional encoding

$$r = a_1 2^{-1} + b_1 2^{-2} + a_2 2^{-3} + \cdots$$

into a real number. Under this interpretation the first player decides whether the resulting number will be in $[0, 1/2)$ or $[1/2, 1)$. Depending on this choice the second player determines the second bit of the number, choosing either between $[0, 1/4)$ and $[1/4, 1/2)$ or between $[1/2, 3/4)$ and $[3/4, 1)$, etc. At infinity the game converges to a point r and the winner is determined according to whether $r \in R$, where R is a subset of $[0,1)$, the *winning condition* of the game (see Figure 5.2).

Another useful geometric metaphor is obtained by decomposing $a_1 b_1 a_2 b_2 \ldots$ into a pair of strings $(a_1 a_2, \ldots, b_1 b_2, \ldots)$ which corresponds to a pair of real numbers (r_a, r_b). Under this interpretation the two players go down a quad-tree toward a two-dimensional board where one player makes horizontal choices, i.e. between the left or right parts of the board, and the other makes vertical decisions between the upper and lower parts, as can be seen in Figure 5.3.

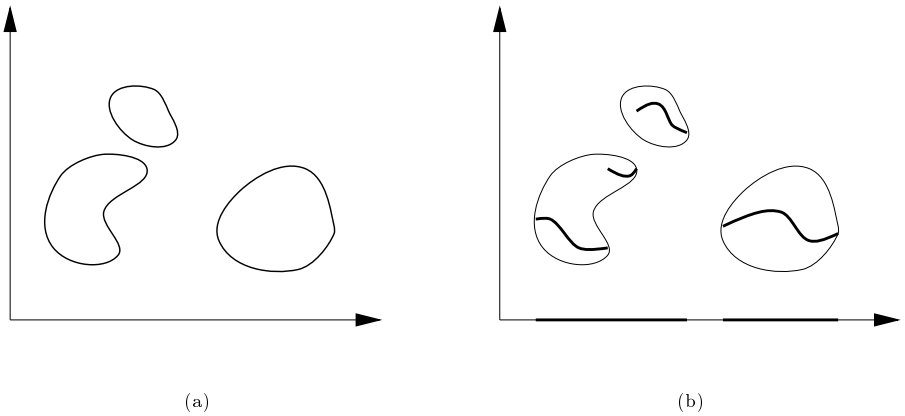

Figure 5.1: The uniformization problem: a relation $R \subseteq X \times Y$ (a) and a function $f : X' \to Y$.

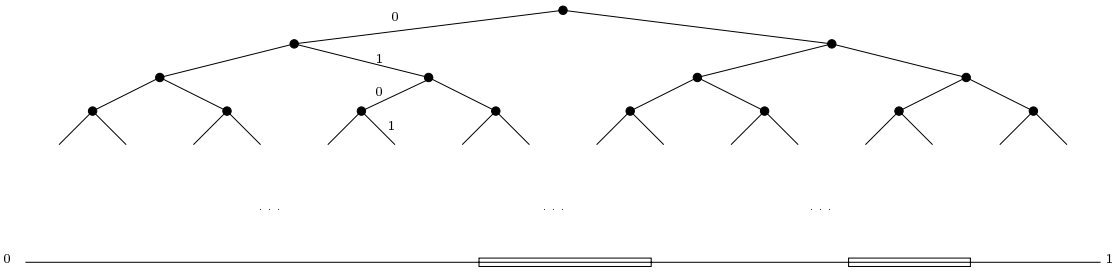

Figure 5.2: The game as a tree with branches interpreted as numbers in $[0, 1]$. The set of winning behaviors is a union of two intervals.

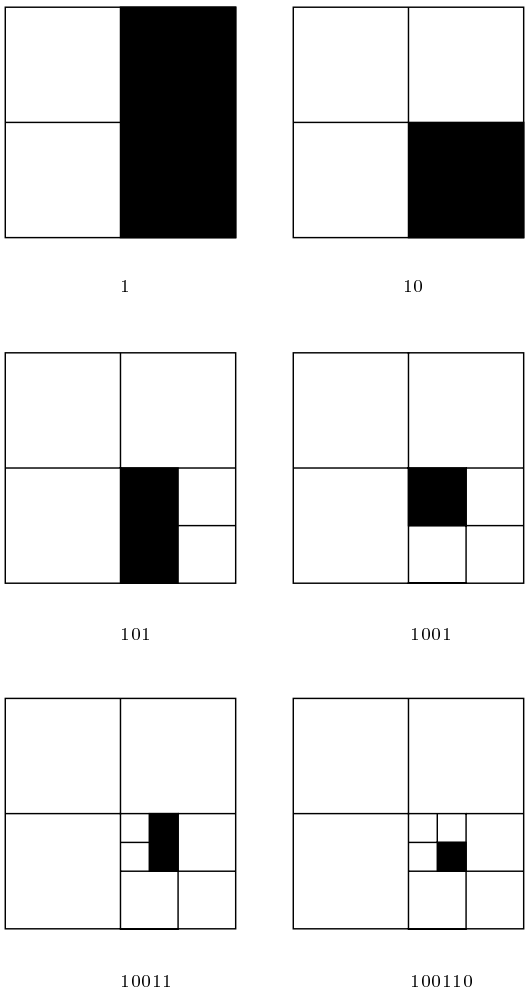

Figure 5.3: The three first steps in a game.

Such infinite games were introduced by Gale and Stewart in the 50s as an extension of the classical finite games of von-Neumann and Morgenstern. The questions posed by mathematicians concerning these games where *existential* in nature and related to the foundations of mathematics and set theory. In particular they asked whether such games are *determined*, i.e. is it true that for every choice of a winning condition, there is a winning strategy for one of the players. The question is not unrelated to the complexity of the winning conditions: over the mathematical "real" numbers (or the set of infinite sequences) one can think of sets of varying degress of complexity, starting from closed or open sets, countable unions or intersections of those, climbing up the Borel hierarchy, then using projections to construct higher classes, and more weird sets. Gale and Stewart have shown that if the winning condition is a closed set, the game is determined. A series of subsequent papers pushed further the limits of determinacy, culminating in a paper by Martin showing that all Borel sets are determined. According to these results, a game where the winning condition is the set of rational numbers (or, equivalently the ultimately-periodic sequences), which is a countable union of closed sets, and hence Borel, is determined. The distance between these games, played in Cantor's paradise of fools, and practical models of real games is, of course, very large.

In parallel with this activity, there developed another, more effective thread of research initiated by Church, which dealt with the question of actually *computing* those strategies. The seminal result in this domain, achieved by Büchi and Lanweber, can be rephrased as follows. If the set of winning behaviors, $R \subseteq (A \times B)^\omega$ is ω-regular, that is, defined by an ω-automaton which accepts it, there is an algorithm to construct a finite-state strategy for one of the players, i.e. an automton (transducer) which observes the evolution of the game and makes the choice for the winning player. Note that ω-regular sets occupy a very modest place in the Borel hierarchy (all are included in $F_{\sigma\delta} \cap G_{\delta\sigma}$), but unlike Martin's results, the results are constructive! Starting from an effective description of the winning conditions of the game, a terminating algorithm generates an effective description of a strategy. In Büchi's last paper (an interesting document in the sociology of science) he poses the challenge to find a constructive version for Martin's theorem, restricted to Borel sets admitting an effective description.

The result of Büchi and Landweber, which was formulated initially in terms of the logic S1S, was rephrased in automata-theoretic (and readable) terms in the book by Trakhtenbrot and Barzdin. This formulation is very intuitive from a practical point of view: suppose the other player is an external environment and the player for whom we want to extract the strategy is a "controller", a hardware or software system we want to design. The winning condition for the game can be viewed as the specification, e.g. "every *request* is followed by a *grant*", and the winning strategy as a correct implementation of the specification. Unfortunately, these results were put in the shade by a stronger result, the seminal Rabin tree automaton theorem. This result shows the decidability of the logic S2S by finding strategies for games defined on *tree automata*. These results subsume BL results (games on automata can be seen as a special case where the tree alphabet consists of a single letter and hence there is only one tree) but they lack the practical intuition that every non-logician or engineer can understand. In retrospect, I think that Rabin's theorem,

which is justly considered as one of the gems of logic, had a negative side-effect in preventing the practical development in synthesis for many years. I recall the thesis of Roni Rosner, also a student of Pnueli, who attacked the synthesis problem using the heavy machinery of tree automata and I believe that their work, which can be viewed as the beginning of the modern practical treatment of the synthesis problem, could have much more impact had it been worked out using a simpler formulation.

Around the same time, similar ideas were re-discovered and re-formulated by M. Wonham (a prominent control theorist) and P. Ramadge, in an attempt to give a sound theoretical basis for what is called discrete-event dynamical systems (DEDS, or poor man's computer science, if you want). The RW model consists of a "plant" which is roughly the external environment to be controlled, and a controller which can interfere with the plant dynamics by disabling certain transitions at various states (this corresponds to the game-theoretic notion of strategy). The nice thing about the RW framework is that it had a very natural feedback control flavor, and that, not being computer scientists, they did not start with complex infinitary acceptance conditions (these were treated later in Thistle's thesis) but with simple reachability (safety) conditions. What I find a less fortunate design choice, was the framing of the problem in language-theoretic terms (the "supremal controllable sub-language" of the plant language). Admittedly, there are some advantages for formulating things in terms of languages, but the computational essence of controller synthesis (and verification) is automata-theoretic — at least from my subjective point of view. In addition, this language-theoretic formulation hides from many in the control community the fact that, if one abstracts away certain technicalities, DEDS control and continuous control treat essentially the *same* problem: influencing the trajectories of some kind of dynamical systems using state-based feedback.

I was looking at these things during my post-doc, discussed some related issues with M. Vardi (I argued, wrongly, that for all ω-regular games you can find the strategy by simple algorithms on the transition graph of the game), thought about proving effective Borel determinacy but have done nothing substantial until the visit of Amir Pnueli in 95. At that time I was sure that research in HS is blocked and I was open to look at various problems related to Timed Automata (TA). These automata, augmented with clock variables, which where introduced by Alur and Dill, turned out to be the most popular among the various models proposed by the verification community in the late 80s for modeling real-time constraints, and they were the subject of an ongoing research and development effort at Verimag (the thesis of Sergio Yovine and the tool Kronos). Pnueli proposed to look at the problem of synthesis for these automata. Automata over trees defined over dense time are even more complicated than discrete time tree automata, so I suggested to backtrack and re-formulate the untimed synthesis problem using simple games on automata with two action alphabets. On these models we have defined the *controllable predecessors* operator which associates with each set F of states, the set $\pi(F)$ consisting of the states from which the controller can *force* the game into F regardless of the actions of the environment:

$$\pi(F) = \{q \in Q : \exists a \in A \ \ \forall b \in B \ \ \delta(q, a, b) \in F\}.$$

Using this operator, the winning states under various winning conditions can be

computed. For example, for the safety condition "always stay in F", the set F^* of winning states can be characterized as the maximal solution of the equation $F^* = F \cap \pi(F^*)$, and can be computed using the following iterative algorithm:

$F_0 := F$
for $i = 1, 2, \ldots,$ **repeat**
$\quad F_i := F_{i-1} \cap \pi(F_{i-1})$
until $F_i = F_{i-1}$
$F^* := F_i$

After having re-established the basic facts concerning untimed games we turned to timed automata. I consider the main achievement of this work, not in the results but in the conceptual framework for analyzing real-time games.[1] The main observations concerning these games were:

1. In these games there is no use speaking about turns — the players can act at any time.

2. Doing nothing and waiting until something better can be done is an important ingredient in real-time strategies. However you need to consider what happens when the adversary does not wait for your actions.

3. Applying the standard definitions uncautiously to timed games may lead to Zeno controllers, controllers which can formally win in hopeless situations by switching infinitely many times between states and "blocking time" (like driving in a car without brakes toward a wall and avoiding a clash by turning the lights on and off in an increasing frequency).

Previous work in these domain has been done by Wong-Toi and Hoffmann, who used the fact that timed automata admit a finite bisimulation (the region graph) and then showed that RW synthesis can be applied to this graph. We preferred a more direct method which works on the TA itself, and synthesizes the controller by restricting the guards and the staying conditions (invariants). Consequently we have defined the predecessor operator over sets of clock valuations and have shown that zones (those polyhedra defining bisimilar regions in the clock space) are closed under this operator. This guarantess the convergence of the algorithm.

The first paper on the topic was rejected from STOC and later accepted to STACS [**P09**]. In a later paper [**P10**] some small bugs have been fixed, some examples of applying synthesis to scheduling problems were given along with a discussion of the use of symbolic methods to fight the state-explosion problem. Unfortunately,

[1] Some mathematicians and pseudo-mathematicians often work within formal models created by others a long time ago, and are used to admire complex technical results *within* the model (something which they themselves know how to do) but not the creative act of building a new model suitable for a new class of phenomena. Personally I will always prefer a paper with wrong technical results but with new concepts and ideas over a mathematically-correct paper, doing some dead truth-preserving reductions in an irrelevant formal world, adding some more epsilon results to the garbage can of history. Of course, this opinion is yet another example of human's ability to invent evaluation criteria which suit themelves.

the journal version is still in the making, so I offer [**P11**] as the mathematically cleanest presentation of the basic results. Later, with E. Asarin we have shown that the results can be extended to find *time optimal* controllers, those which make the automaton reach F as fast as possible [**P12**]. This result, which uses an extension of the region graph concept to deal with a certain class of cost functions may lead to other interesting connections between timed automata and other timed models such as the Max-Plus algebra.

The models used in these papers have been used as a framework for describing scheduling problems in real-time and multi-media applications. They also inspired some work on controller synthesis for hybrid systems, once the control theorists grasped the difference between discrete and continuous predecessors. Our recent work in this direction is described in Section 11.

References

M. Abadi, L. Lamport, and P. Wolper, Realizable and Unrealizable Concurrent Program Specifications. In *Proc. 16th ICALP*, 1-17, LNCS 372, Springer, 1989.

K. Altisen, G. Goessler, A. Pnueli, J. Sifakis, S. Tripakis and S. Yovine, A Framework for Scheduler Synthesis, *IEEE Real-Time Systems Symposium*, 1999.

R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.

J.R. Büchi and L.H. Landweber, Solving Sequential Conditions by Finite-state Operators, *Trans. of the AMS* 138, 295-311, 1969.

J.R. Büchi, State-strategies for Games in $F_{\sigma\delta} \cap G_{\delta\sigma}$, *J. of Symbolic Logic* 48, 1171-1198, 1983.

A. Church, Logic, Arithmetic and Automata, in *Proc. of the Int. Cong. of Mathematicians 1962*, 23-35, 1963.

D. Gale and F.M. Stewart, Infinite Games with Perfect Information, in H.W. Kuhn and A.W. Tucker (Eds.), *Contribution to the Theory of Games* II, 245-266, Princeton University Press, 1953.

D.A. Martin, Borel determinacy, *Annals of Mathematics(2)*, 102, 363-371, 1975.

J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module, In *Proc. 16th ACM Symp. Princ. of Prog. Lang.*, pages 179–190, 1989.

A. Pnueli and R. Rosner. On the Synthesis of an Asynchronous Reactive Module, In *Proc. 16th ICALP*, 653-671, LNCS 372, Springer, 1989.

M.O. Rabin, Automata on Infinite Objects and Church's Problem, AMS, 1972.

P.J. Ramadge and W.M. Wonham, Supervisory Control of a Class of Discrete Event Processes, *SIAM J. of Control and Optimization* 25, 206-230, 1987.

P.J. Ramadge and W.M. Wonham, The Control of Discrete Event Systems, *Proc. of the IEEE* 77, 81-98, 1989.

J.G. Thistle and W.M. Wonham, Control of Infinite Behavior of Finite Automata, *SIAM J. of Control and Optimization* 32, 1075-1097, 1994.

W. Thomas, On the Synthesis of Strategies in Infinite Games, In E.W. Mayr and C. Puech (Eds.), *Proc. STACS '95*, 1-13, LNCS 900, Springer, 1995.

B.A. Trakhtenbrot and Y.M. Barzdin, *Finite Automata: Behavior and Synthesis*, North-Holland, Amsterdam, 1973.
H. Wong-Toi and G. Hoffmann, The Control of Dense Real-Time Discrete Event Systems, Technical report STAN-CS-92-1411, Stanford University, 1992.

My Papers

[**P09**]
O. Maler, A. Pnueli and J. Sifakis, On the Synthesis of Discrete Controllers for Timed Systems, In E.W. Mayr and C. Puech (Eds.), Proc. STACS '95, 229-242, LNCS 900, Springer, 1995.
[**P10**]
E. Asarin, O. Maler and A. Pnueli, Symbolic Controller Synthesis for Discrete and Timed Systems, in P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Eds.), *Hybrid Systems II*, 1-20, LNCS 999, Springer, 1995.
[**P11**]
E. Asarin, O. Maler, A. Pnueli and J. Sifakis, Controller Synthesis for Timed Automata, in *Proc. IFAC Symposium on System Structure and Control*, 469-474, Elsevier, 1998.
[**P12**]
E. Asarin and O. Maler, As Soon as Possible: Time Optimal Control for Timed Automata, in F. Vaandrager and J. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, 19-30, LNCS 1569, Springer, 1999.

Chapter 6

Circuits and Timed Automata

One of the initial motivations for timed automata was the modeling of circuits with uncertain but bounded propagation delays. The fact that timed systems can be analyzed by reasoning on polyhedra in the clock space had been discovered already in 83 by Berthomieu and Menasche in the context of Timed Petri nets, but only the work of Alur and Dill put the interaction between logic and timing in the center of attention of the verification community.

In the untimed world, the relation between circuits with memory (sequential circuits, as called in electrical engineering) and finite automata is fundamental. The facts that every circuit consisting of Boolean gates and flip-flops can be modeled as an automaton, and that any automaton can be realized, via state-encoding, using combinational and memory elements is part of the professional background of every engineer. This folk theorem can be phrased in terms of sequential operators (functions from sequences to sequences):

Folk Theorem: *Every finite-state operator*[1] *can be realized by a network of interconnected components taken from a class of operators containing:*

1. *a sufficient basis for the Boolean functions (e.g.* $\{\wedge, \vee, \neg\}$*), and*

2. *the* unit delay *operator,* $D : X^\omega \to X^\omega$*, defined as* $\beta = D(\alpha)$ *if for every* $i > 1$*,* $\beta[i] = \alpha[i-1]$*.*

Every network of such operators realizes a finite-state operator.

The delay operator is equivalent to a special automaton, the 1-place shift-register (Figure 6.1) whose state records the last input read. A k-step delay D_k can be defined by letting $D_1 = D$ and $D_{i+1} = D(D_i)$, and the corresponding automaton is the k-place shift register with 2^k states used to memorize all possible input histories of length k. The transition graph of the shift register is also known as the de-Bruijn graph.

The semantics of a circuit constructed this way can be given in terms of solutions of systems of equations over sequences. For example, the behavior of the circuit in Figure 6.2 is captured by the equation $x = y \oplus D(x)$. Similarly the behavior of any synchronous sequential circuit consisting of gates and latches, as in figure 6.3, can

[1]The number of "states" of an operator is exactly the index of its Myhill-Nerode congruence.

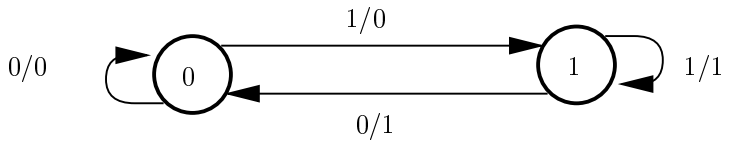

Figure 6.1: The automaton realizing the D operator in discrete qualitative time.

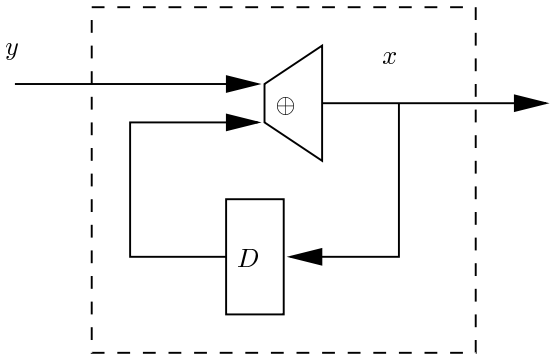

Figure 6.2: A simple sequential circuit whose semantics is defined by $x = y \oplus D(x)$.

be written as a solution of a system of equations of the form

$$\begin{aligned} x_1 &\in D(f_1(x_1, \ldots, x_n, y_1, \ldots, y_m)) \\ &\cdots \\ x_n &\in D(f_n(x_1, \ldots, x_n, y_1, \ldots, y_m)) \end{aligned}$$

where the x_i are the state variables, y_i are the input variables and every f_i is a memoryless function computing the next value of x_i. Programs in the synchronous data-flow languages Signal and Lustre are, in fact, written as such equations.

After having understood how these principles work in the untimed case, I could move to their extension to metric time. Here the natural objects for charaterizing the semantics are not sequences but rather Boolean *signals*, functions defined over the real time axis. With every positive number d we can associate a delay operator D_d such that $\beta = D_d(\alpha)$ if for every $t \geq d$, $\beta[t] = \alpha[t - d]$. However, this type of operator cannot be realized by any automaton having a finite number of states and clocks because a-priori there is no bound on the number of changes in α which may occur in an interval $[t, t + d]$ and hence no bound on the number of distinct input histories that the automaton might need to remember. There are two approaches to alleviate this problem: one is to restrict the domain of the operator to "well-behaving" input signals with bounded variability, e.g. signals which stay stable for at least d time after every change. The alternative approach is to use an intertial (latency) delay model, where fast changes in the input are filtered away and not propagated to the output.

The most important contribution of timed automata to the analysis of delay phenomena is in expressing delay *uncertainty*, which from the point of view of verification methodology, is yet another type of environmental non-determinism. To

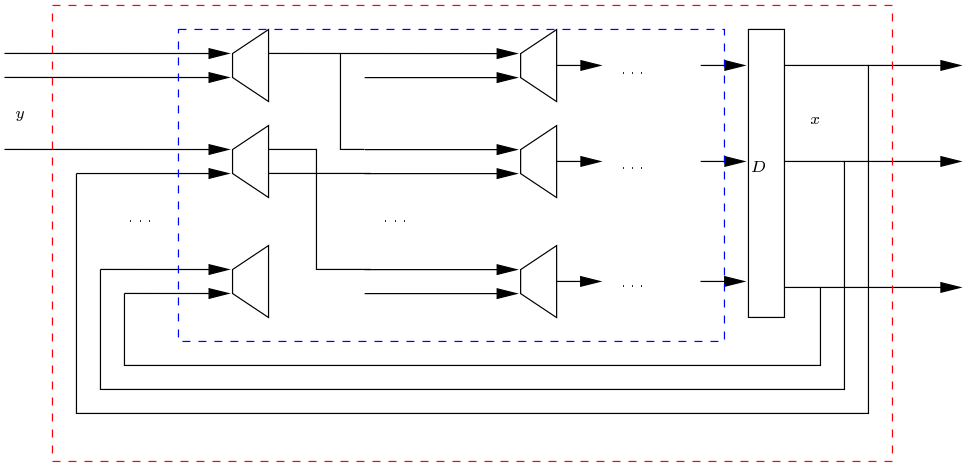

Figure 6.3: A synchronous sequential circuit.

this end we have defined a non-deterministic delay operator $D_{[l,u]}$ parametrized by lower- and upper-bound on the propagation delay. Intuitively, $\beta \in D_{[l,u]}(\alpha)$ if

- Every change in β at t is preceded by a similar change in α which happened at least l time before t and persisted until t.
- Every change in α at t which persists for u time must be propagated to β.

Figure 6.4 shows a signal α and some elements of the set $\Delta_{[1,3]}(\alpha)$. The semantics of a circuit consisting of interconnected gates with such bi-bounded delays is the set of solutions of a system of simultaneous signal inclusions of the form

$$\begin{aligned} x_1 &= D_{[l_1,u_1]}(f_1(x_1,\ldots,x_n,y_1,\ldots,y_m)) \\ &\cdots \\ x_n &= D_{[l_1,u_1]}(f_n(x_1,\ldots,x_n,y_1,\ldots,y_m)) \end{aligned}$$

The next step is to associate a timed automaton with each delay inclusion such that the semantics of the automaton is exactly the set of solutions. For convenience we introduce auxiliary variables z_i such that any equation is split into

$$z_i = f_i(x_1,\ldots,x_n,y_1,\ldots,y_m)$$

and

$$x_i \in D_{[l_i,u_i]}(z_i)$$

The first equation is just a constraint on the values of the variables at each time instant, hence it can be transformed into a one-state automaton. The automaton for the delay equation appears in Figure 6.5. At state 0 the input and the output values are the same and the state is stable. Once the input changes to 1 the automaton moves to the unstable state $0'$, the output remains 0 but a clock is reset to zero to record the time of the transition. The automaton can stay at $0'$ until the clock reaches the upper-bound, but it can move to 1 (and change the output) once

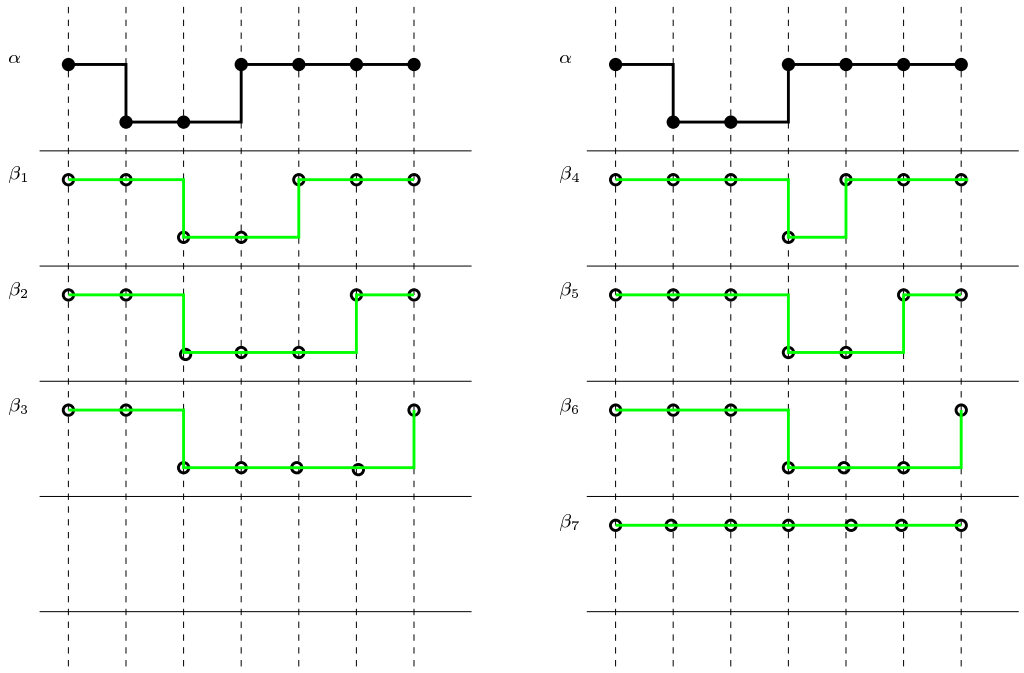

Figure 6.4: $\{\beta_1, \ldots, \beta_7\} \subseteq \Delta_{[1,3]}(\alpha)$.

the lower-bound has been attained. The transition back from $0'$ to 0 models the possibility for the input to "regret" before the expiration of the upper bound, and it can be avoided if the inputs are well-behaving. The transitions from 1 to $1'$ and 0 are symmetric. This automaton is a rather generic mathematical object, the metric extension of the shift register. Composing these automata together we get:

Theorem: *Every k-variable circuit with bi-bounded delays can be translated into an equivalent timed automaton with* 2^k *states and k clocks.*

The significance of this result, published in [**P13**], is that it allows to verify all the behaviors of a circuit under *all* choices of firing times of the delay elements and *all* choices of input arrival times. This is in contrast with current practice in hardware timing analysis which is based either on non-exhaustive simulations or on approximate analysis based on decoupling logic from timing. In [**P14**] we have applied this approach to model MOS circuits as timed automata and verified them using Kronos. However, like in other application domains of timed automata, there is still a performance bottleneck limiting the size of problems which can be treated, and hence practitioners still stick to less ambitious but more efficient methods. Every now and then I try to convince the hardware community (both academic and industrial) to adopt timed automata as the basic modeling formalism for delay phenomena, and although we are not yet there, I believe the gap is being slowly bridged. Some attempts to improve performance are described in the next section.

In addition to the practical motivation, the study of circuits and their relation to automata is fundamental from a theoretical point of view (some recent studies of Trakhtenbrot are aimed explicitly at lifting classical automata theory to deal

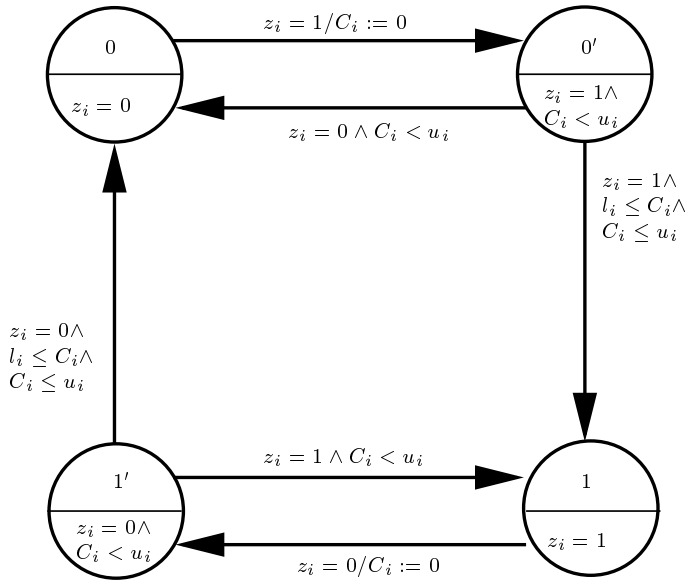

Figure 6.5: The automaton for $x_i \in D_{l_i,u_i}(z_i)$.

with metric time) and since circuits constitute a well-behaving sub-class of timed systems, they may hint toward the (still missing) "good" class of timed automata. It is interesting to note that the automata obtained via our translation from circuits have a structure similar to the "event-clock" clock automata of Henzinger et al.

References

R. Alur, L. Fix and T.A. Henzinger, A determinizable class of timed automata, in D. Dill (Ed.) *Proc. CAV '94*, 1–13, LNCS 818, Springer, 1994.

R. Alur and D.L. Dill, A theory of timed automata, *Theoretical Computer Science* 126, 183–235, 1994.

A. Benveniste and P. LeGuernic, Hybrid dynamical Systems Theory and the Signal language, *IEEE Trans. on Automatic Control* 35, 535-546, 1990.

B. Berthomieu and M. Menasche, An Enumerative Approach for Analyzing Time Petri Nets, in R.E.A. Mason (Ed.) *Information Processing 83*, North-Holland, 1983.

J.A. Brzozowski and C-J.H. Seger, Advances in Asynchronous circuit theory Part II: Bounded Inertial Delay Model, MOS Circuits, Design Techniques, *EATCS Bulletin* 43, 199-263, 1991.

D. Dill, Timing Assumptions and Verification of Finite-State Concurrent Systems, in J. Sifakis (Ed.), *Automatic Verification Methods for Finite State Systems*, LNCS 407, Springer, 1989.

N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, The Synchronous Data-flow Programming Language LUSTRE, *Proc. of the IEEE* 79, 1305-1320, 1991.

W.K.C. Lam and R.K. Brayton, *Timed Boolean Functions*, Kluwer, 1994.

H.R. Lewis, Finite-state Analysis of Asynchronous Circuits with Bounded Temporal Uncertainty, TR15-89, Harvard University, 1989.

A. Rabinovich and B.A. Trakhtenbrot, From finite automata toward hybrid systems, *Proc. FCT'97*, 1997.
S. Tasiran, Y. Kukimoto and R.K. Brayton, Computing Delay with Coupling using Timed Automata, *Proc. TAU'97*, 1997.
S. Tasiran, S.P. Khatri, S. Yovine, R.K. Brayton and A. Sangiovanni-Vincentelli, A Timed Automaton-Based Method for Accurate Computation of Circuit Delay in the Presence of Cross-Talk, *Proc. FMCAD'98*, 1998.

My Papers

[**P13**]
O. Maler and A. Pnueli, Timing Analysis of Asynchronous Circuits using Timed Automata, in P.E. Camurati, H. Eveking (Eds.), *Proc. CHARME'95*, 189-205, LNCS 987, Springer, 1995.
[**P14**]
O. Maler and S. Yovine, Hardware Timing Verification using KRONOS, In *Proc. 7th Israeli Conference on Computer Systems and Software Engineering*, Herzliya, Israel, June 1996.

Chapter 7

Timed Systems: BDDs and Discretization

Verification of timed automata consists in manipulating certain types of polyhedra which deserve to be called *timed polyhedra*. Roughly these are the sets which can be written as the Boolean closure of half-spaces of the form $x_i - x_j < c$ or $x_i < c$. When these polyhedra are convex, they admit an efficient canonical representation, however during the execution of verification algorithms, the set of reachable configuration usually becomes non-convex and is represented in Kronos as a list of convex sets. Calculating successors and checking convergence of reachability algorithms becomes a very complex and time-consuming task.

During Pnueli's visit in '95 we started to look for a canonical represenation for non-convex timed polyhedra. Our source of inspiration were the ordered BDDs (Binary Decision Diagrams) a well-known canonical data-structure for representing subsets of $\{0,1\}^n$, which underlies the success of symbolic model-checking in treating systems with a huge state-space. Our first attempt was to assign a Boolean variable to every inequality of the from $x_i - x_j < c$ but there are too many such variables and they are not mutually independent. Moreover, when doing operations on sets you might need to refine such an inequality into $x_i - x_j < c'$ and $c' < x_i - x_j < c$, and hence split a Boolean variable into two. In the worst case you might end up having a Boolean variable for every pair of variables and every unit interval. While this might work in practice, it lacks the mathematical elegance of BDDs. Contemplating on a minimal set of comparisons which is sufficient to identify any subset, we came to the following observation: if you look at discrete time interpretation of clock variables, clocks are just bounded integer variables, their values can be encoded in binary, and sets of values of n clocks ranging in $\{0,\ldots,k-1\}$ are just Boolean functions with $n \log k$ variables, which inherit the canonical properties of BDDs (see Figure 7.1).

For two years these ideas remained unimplemented and the question whether this representation can improve performance remained open. Fortunately, when I presented this result in a VERIMAG seminar, Marius Bozga saw similarities with some version of BDDs he already implemented, and agreed to implement our BDDs as well. The empirical results (reported in [**P15**]) were mixed. On one hand, many examples behaved better than the classical version of Kronos, especially when there

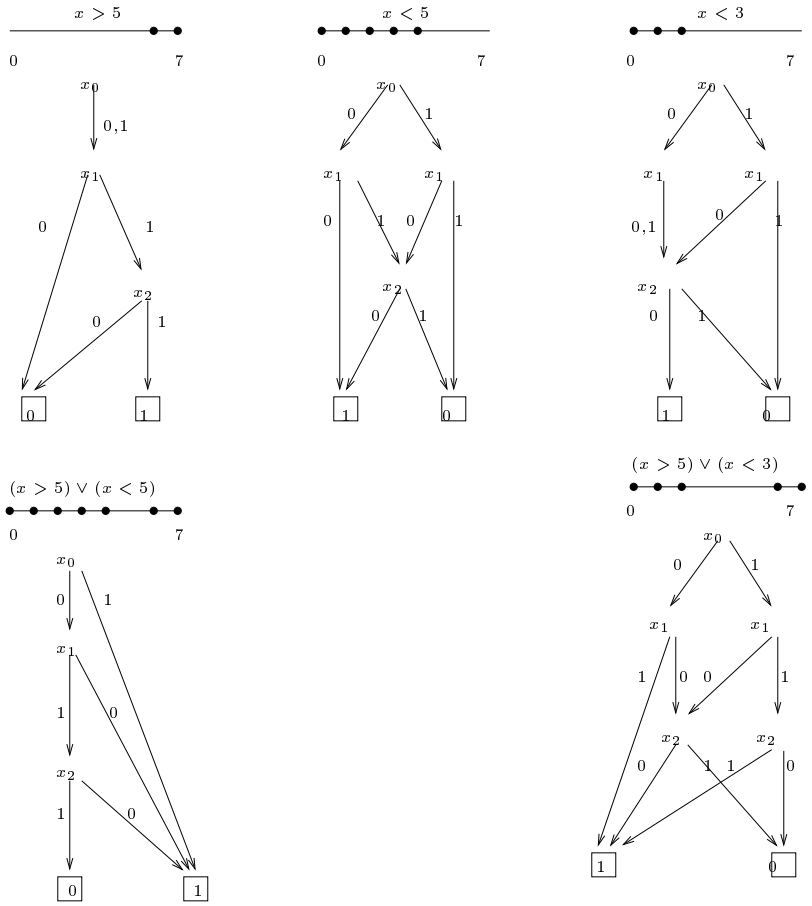

Figure 7.1: Some timed polyhedra in one dimension and their BDD representation.

was an explosion in the discrete state-space, handled enumeratively by Kronos. On the other hand, due to the binary encoding of the clock space, the results were sensitive to the size of the constants in the system, and we were not able to go as far as we hoped. The best results obtained so far by the BDD version were concerned with an asynchronous ciruit, STARI, designed by M. Greenstreet, where we were able to verify a system with 54 clocks ([**P16**]). So the bottom line is that we added one more representation technique to the arsenal of Kronos, a technique which in certain cases performs better than the others.

Since the BDD version works on discrete time, it remained to show under what conditions the verification results hold also for the dense time interpretation. Clearly the set of discrete time behaviors is a subset of the dense semantics, but the interesting question is whether every qualitative behavior realizable in dense time has a similar discrete time representative. Using some geometrical reasoning we have understood the correspondence between the discrete and dense time semantics, and in particular that:

1. Every acyclic automaton can be discretized without loss of semantics using a time step $1/m$ where m is the longest path in the automaton.

2. Some cyclic automata may exhibit infinitary behaviors which are not possible under *any* discretization.

3. All automata where the timing constraints are closed (no strict inequalities) can be discretized using time step 1.

The last of these facts, all published in [**P17**], validates the results obtained using the BDD version. While looking at this problem I realized that the density of time is not the main issue in timed automata, but rather the fact that the state-space consists of domains which admit order and metric, and that you can use Boolean combinations of inequalities to represent subsets of the clock space (also in discrete time). The fact that in dense time enumeration is impossible and symbolic representation is mandatory led many to the misconception that discrete time is inherently enumerative.

References

R.E. Bryant, Graph-based Algorithms for Boolean Function Manipulation, *IEEE Trans. on Computers* C-35, 677-691, 1986.

J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang, Symbolic Model-Checking: 10^{20} States and Beyond, *Proc. LICS'90*, Philadelphia, 1990.

C. Daws, A. Olivero, S. Tripakis, and S. Yovine, The Tool KRONOS, in R. Alur, T.A. Henzinger and E. Sontag (Eds.), *Hybrid Systems III*, LNCS 1066, 208-219, Springer, 1996.

A. Göllü, A. Puri and P. Varaiya, Discretization of Timed Automata, *Proc. 33rd CDC*, 1994.

M. R. Greenstreet, STARI: Skew Tolerant Communication, to appear in *IEEE Transactions on Computers*, 1997.

T. Henzinger, Z. Manna, and A. Pnueli. What Good are Digital Clocks?, in W. Kuich (Ed.), *Proc. ICALP'92*, LNCS 623, 545-558, Springer, 1992.
K.L. McMillan, *Symbolic Model-Checking: an Approach to the State-Explosion problem*, Kluwer, 1993.
S. Tasiran and R.K. Brayton, STARI: A Case Study in Compositional and Hierarchical Timing Verification, in O. Grumberg (Ed.) *Proc. CAV'97*, 191-201, LNCS 1254, Springer, 1997.

My Papers

[**P15**]
M. Bozga, O. Maler, A. Pnueli, S. Yovine, Some Progress in the Symbolic Verification of Timed Automata, in O. Grumberg (Ed.) *Proc. CAV'97*, 179-190, LNCS 1254, Springer, 1997.
[**P16**]
M. Bozga, O. Maler and S. Tripakis, Efficient Verification of Timed Automata using Dense and Discrete Time Semantics, in L. Pierre and T. Kropf (Eds.), Proc. CHARME'99 , 125-141, LNCS 1703, Springer, 1999.
[**P17**]
E. Asarin, O. Maler and A. Pnueli, On the Discretization of Delays in Timed Automata and Digital Circuits, in R. de Simone and D. Sangiorgi (Eds), *Proc. Concur'98*, LNCS 1466, 470-484, Springer, 1998.

Chapter 8

Discrete Infinite-State Systems

The notion of a finite-state automaton augmented with variables of various data-types is now commonly accepted as a useful model of computation, capable of expressing theoretical constructs such as Turing machines, or two-counter machines, as well as operational semantic models of programs in high-level languages where the automaton states correspond to program locations (as in the SPL language used in Manna and Pnueli books). Timed and hybrid systems can also be seen as automata having auxiliary variables and although they pose additional problems due to the continuous time domain, they share with discrete infinite-state systems the property of potential infinitude of the sets of reachable states. Hence these sets cannot be represented enumeratively and a symbolic representation technique is required. In an attempt to re-utilize the insights gained in treating timed and hybrid systems, I started looking in '94 at one of the simplest infinite-state systems known to computer scientists, the push-down automaton (PDA) which is an automaton augmented with one auxiliary variable, the push-down stack, which can store values belonging to Γ^* for some alphebet Γ. Note that we are not looking at the PDA as an acceptor of some language over an *input* alphabet Σ, as is common in formal language theory, but rather as a kind of dynamical system over the state-space $Q \times \Gamma^*$.

The Push and Pop operations on a stack can be captured by the rewrite rules $\gamma \to w$ and $\gamma \to \varepsilon$ where the first rule means that if the top element of the stack is γ you replace it by the string w, i.e. you transform any string of the form γu into wu. The second rule just pops γ out of the stack. In order to calculate the set of successors or predecessors of a set $L \subseteq \Gamma^*$ via such rules, I started to look at the rewriting literature and came across a theorem in a book by Book and Otto. This theorem states that the set of successors of a regular set L via a *monadic string rewriting system* is also regular (a monadic string rewriting system can be seen as a PDA having only one control state). The proof of this result is based on transforming an automaton representation of L into an automaton representation of $Post(L)$, and I'll illustrate how this technique can be used to compute the predecessors of L, $Pre(L)$, under the rules $\gamma \to w$ and $\gamma \to \varepsilon$.

Let $\mathcal{A}$ be an automaton accepting L. A new automaton $\mathcal{A}'$ is created by adding transitions as follows: for every state q reachable from the initial state upon reading the word w you add a transition (q_0, γ, q). If a word of the form wu led to an accepting state in $\mathcal{A}$, the word γu, its predecessor via the rewrite rule, is accepted

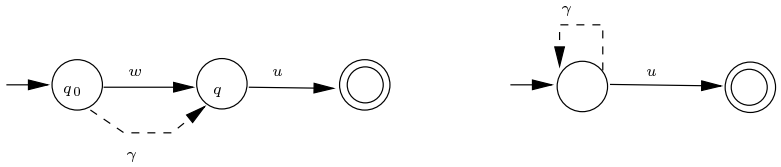

Figure 8.1: Adding predecessors via the rules $\gamma \to w$ (left) and $\gamma \to \varepsilon$ (right).

by $\mathcal{A}'$. For the rule $\gamma \to \varepsilon$ you just add a self-loop labeled by γ to the initial state (see Figure 8.1). This procedure is repeated until the automaton stabilizes (since no states are added, this is guaranteed to happen).

The adaptation of this technique to PDAs was quite straightforward: a subset of $Q \times \Gamma^*$ can be written as a finite union of sets of the form $\{q_i\} \times L_i$, and instead of an automaton for L, you use a "multi-automaton" which accepts the set of languages $L_1, \ldots, L_n$. At that time I was motivated to extend the idea to alternating PDAs (having both existential and universal non-determinism) in order to find strategies for push-down games. I had some ideas for using alternating automata to represent sets of states but I could not prove anything. I discussed the matter with Ahmed Bouajjani during a flight to the US in '95 – I still remember us arguing about the proof on the moving belt of Newark airport. Ahmed, who worked before on push-downs and fixed-points, explained to me that when we add the self-loop to the initial state, we do what is called *acceleration*: adding in one iteration step *all* the predecessors of L under an arbitrary number of applications of $\gamma \to \varepsilon$. We kept working on this problem on and off, until finally Ahmed submitted it to a workshop on inifnite-state systems. Later, J. Esparza showed that this procedure can improve the complexity(!) of model-checking for certain logics, and hence this technique, published in [**P18**], received the appreciation of those who admire such things. Since then, several authors have used these techniques to reason about systems with other type of string-like variables (e.g. FIFO buffers) and about the control skeletons of programs with recursive procedure calls.

In parallel to this activity, Pnueli made me look at the problem of verifying properties of networks consisting of an arbitrary number of processes, and for which some language-theoretic approaches have already been tried by by Clarke and Grumberg. It is not hard to see that if you have an arbitrary number of processes in state a and one process in state b (say, the process who has the token), you can express this set as a string a^*ba^*. This expression represents all the configuration of this type for *any* number of processes. Local transition of one process from state a to state c can be expressed as a rewriting rule $a \to c$. If we assume that processes are oredered linearily and communicate with their neighbors, the act of token passing between neighbors can be expressed using the rewrite rule $ab \to ba$. If the number of processes always remains fixed, all such systems can be captured using length-preserving rewrite rules. Pnueli generalized the technique of calculating successors and predecessors by building a special automaton for the rewrite rules (a "transducer") and gave a general verification procedure for verifying such process networks [**P19**]. Of course, this procedure need not terminate, because unlike PDA, the reachability

problem is undecidable for these systems.

References

R.V. Book and F. Otto, String-Rewriting Systems, Springer, 93.
A. Bouajjani and P. Habermehl, Symbolic Reachability Analysis of Fifo-Channel Systems with Nonregular Sets of Configurations, *Theoretical Computer Science* 221, 211-250, 1999.
E.M. Clarke, O. Grumberg and S. Jha, Verifying Parametrized Networks using Abstractions and Regular Languages, *proc. CONCUR 95*, 1995.
Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*, Springer, 1992.
Z. Shtadler and O. Grumberg, Network Grammars, Communication Behaviors and Automatic Verification, *Proc. CAV'89*, LNCS 407, Springer, 1989.

My Papers

[**P18**]
A. Bouajjani, J. Esparza and O. Maler, Reachability Analysis of Pushdown Automata: Application to Model-Checking, in A. Mazurkiewicz and J. Winkowski (Eds.), *Proc. CONCUR'97*, 135-150, LNCS 1243, Springer, 1997.
[**P19**]
Y. Kesten, O. Maler, M. Marcus, A. Pnueli and E. Shahar, Symbolic Model Checking with Rich Assertional Languages, in O. Grumberg (Ed.) *Proc. CAV'97*, 424-435, LNCS 1254, Springer, 1997.

Chapter 9

Timed Regular Expressions

Automata theory developed slowly at the 50's and 60's, at least so it looks to someone coming to it few decades later. The developments of timed and hybrid systems took place in a much more hectic atmosphere where new variants of machines, real-time logics or process algebras are published in high frequency, so that it is hard sometimes to tell the forest from the trees. Toward the end of '95 I started to look at the possibility of extending Kleene theorem concerning the relation between automata and regular expressions to treat timed systems. The proof of Kleene theorem, which is part of the "theory" that almost every computer scientist learns at school,[1] has two parts, the first (and useful) part is almost trivial: construct inductively a non-deterministic automaton which corresponds to a given expression. The harder part of the theorem which builds a regular expression from any given automaton can be proved in several ways. One is to associate with every automaton a system of language equations, where each variable L_i corresponds to the language accepted from state q_i. For example, the equations which correspond to the automaton of Figure 9.1 are:

$$L_1 = a \cdot L_2 + b \cdot L_1$$
$$L_2 = a \cdot L_1 + b \cdot L_2$$
$$L_3 = (a + b) \cdot L_3 + \varepsilon$$

Using some kind of Gaussian elimination, and the fact that the solution of an equation of the form $X = A + B \cdot X$ is $X = B^* \cdot A$, the regular expressions can be obtained.

[1] Kleene's original paper proved the result using a model that we will call today a neural network, which is in fact, a network of threshold gates and delays.

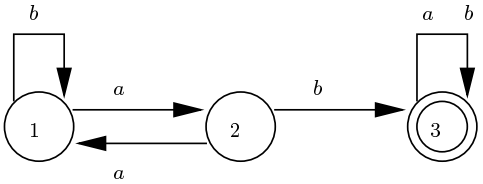

Figure 9.1: An automaton.

The other proof, due to McNaughton and Yamada, is based on defining a family of functions $\{R^i : Q \times Q \to \Sigma^*\}_{i=1}^n$ where $R^k(i,j)$ denote the set of sequences leading from q_i to q_j without passing at states in $\{q_{k+1}, \ldots, q_n\}$. Clearly $R^n(i,j)$ consists of *all* the sequences leading from q_i to q_j. The functions are defined inductively starting from $R^0(i,j)$, which is simply the set of letters that lead *directly* from q_i to q_j, and proceeding via the formula

$$R^{k+1}(i,j) = R^k(i,j) \cup R^k(i,k+1) \cdot (R^k(k+1,k+1))^* \cdot R^k(k+1,j).$$

The first term stands for sequences leading from q_i to q_j without ever visiting q_{k+1}. The second term denotes words that lead from q_i to q_{k+1} (without visiting q_{k+1} in the way), followed by an arbitrary number of words that induce a cycle from q_{k+1} to itself and ending with a word leading from q_{k+1} to q_j. This procedure derives the regular expression $R^n(i,j)$ for every pair of states and hence solves the problem.

Timed languages were defined previously in terms of *timed traces*, namely sequences of the form $(a_1,t_1),(a_2,t_2),\ldots$ where every pair (a_i,t_i) denotes the occurrence of the event a_i at time t_i. Following the work on circuits, I preferred to work with signals rather than with timed traces which, by their nature, do not admit a nice monoidal concetenation intuition.[2] In untimed words, the basis for defining regular expressions are the alphabet letters, where a denoted the occurrence of a for *one* time instant. For indivisdual signals a natural choice is to use expressions of the form a^3b^5 for denoting signals which are a for an interval of time of length 3 and then b for 5 time. Concatenation, finite union and star are completely non-problematic operations on signlas, but the major question is how to express infinite dense unions, such as the set of signals which are a for *some* time in the interval $[l,u]$ and then become b. The first direction which I pursued was to use *first-order* regular expressions with a^x denoting that the duration of a is x for some free variable, which can be restricted later. This approach considered three classes of syntactical objects: the *symbol expressions*, each denoting some subset of A, the *time expressions*, restricted arithmetic expressions constructed from finitely many integer constants and variables and the regular expressions constructed from symbol and time expressions. Formally:

- $\mathcal{E}_S$: A symbol expression is either a symbol $a \in A$ or $\alpha_1 \vee \alpha_2$ where $\alpha_1, \alpha_2 \in \mathcal{E}_S$.
- $\mathcal{E}_T$: A time expression is either a constant $k \in K$, a variable $x \in X$ or $\tau_1 + \tau_2$, $\tau_1 - \tau_2$, where $\tau_1, \tau_2 \in \mathcal{E}_T$.
- $\mathcal{E}$: A first-order regular expression is either α^τ, $\varphi_1 \cdot \varphi_2$, $\varphi_1 \vee \varphi_2$, φ^* or

$$\bigvee_{k_1 < x \leq k_2} \varphi$$

  where α is a symbol expression, τ is a time expression, x is a variable, k_1, k_2 are constants and $\varphi, \varphi_1, \varphi_2 \in \mathcal{E}$.

[2] Now we know also how to build a decent monoid for an event-based representation, namely the shuffle monoid of Σ^* and $\mathbb{R}_+$. This way a timed trace such as $(a,5)(b,5)(c,7)$ can be written as $5ab2c$.

Examples for expressions: $a_1^{x_1}$, $(a_1+a_2)^{x_1}$, $a_1^{x_1}\cdot a_2^{3-x_1}$, $\bigvee_{2<x_1\leq 6} a_1^{x_1}$, $\bigvee_{2<x_1\leq 8}(a_1^{x_1}\cdot a_2^{8-x_1})$.

Note that the expression $a_1^{x_1}\cdot a_2^{3-x_1}$ is the kind of expression you obtain in the language equation associated with TA transitions that do not reset the clock.

The natural automata for accepting sets of signals are automata where input letters are associated with *states* rather than with transitions. I discussed these expressions with Paul Caspi who helped me to give a context-based semantics to such objects. Then we sat down to prove the correspondence between these expressions and timed automata but failed to match the expressive power of the regular expressions with "classical" timed automata, so we had to invent another variant of automata. We sent a paper to some Israeli theory conference, but then found a bug and retracted the paper (we got rejected nevertheless). The situation in this front looked rather gloomy, until E. Asarin came to the rescue. During his winter visit, he inspected the first-order expressions, located their place on the border of diophantine equations, and finally came out with another syntax, which was in one sense weaker (no complicated time expressions) and in others stronger (including intersection).

Syntax and Semantics of Timed Regular Expressions: *A timed regular expressions over an alphabet Σ, is defined recursively as either a, $\varphi_1\cdot\varphi_2$, $\varphi_1\vee\varphi_2$, $\varphi_1\wedge\varphi_2$, φ^* or $\langle\varphi\rangle_I$ where $a\in\Sigma$, $\varphi,\varphi_1,\varphi_2\in\mathcal{E}(\Sigma)$ and I is an integer-bounded interval. The semantics of timed regular expressions, is given by:*

$$\begin{array}{lcl}
[\![a]\!] & = & \{a^r : r\in\mathbb{R}_+\}\\
[\![\varphi_1\vee\varphi_2]\!] & = & [\![\varphi_1]\!]\cup[\![\varphi_2]\!]\\
[\![\varphi_1\wedge\varphi_2]\!] & = & [\![\varphi_1]\!]\cap[\![\varphi_2]\!]\\
[\![\varphi_1\cdot\varphi_2]\!] & = & [\![\varphi_1]\!]\circ[\![\varphi_2]\!]\\
[\![\varphi^*]\!] & = & \bigcup_{i=0}^{\infty}([\![\varphi^i]\!])\\
[\![\langle\varphi\rangle_I]\!] & = & [\![\varphi]\!]\cap\{\xi : |\xi|\in I\}
\end{array}$$

Here the basis a denotes a signals of arbitrary length, and the timing information is expressed via the $\langle\varphi\rangle_I$ operator which restricts the metric length of the signals in $[\![\varphi]\!]$ to be in I. Some examples of expressions and their semantics are given below:

φ	$[\![\varphi]\!]$
$\langle a\rangle_{(0,3]}$	$\{a^x : x\in(0,3]\}$
$\langle(a\cdot b)^*\rangle_{(0,3]}$	$\bigcup_k\{a^{r_1}b^{s_1}\cdots a^{r_k}b^{s_k} : r_i,s_i\in\mathbb{R}_+\wedge\sum_{i=1}^k(r_i+s_i)\in(0,3]\}$
$\langle a\cdot b\rangle_3\cdot c\wedge a\cdot\langle b\cdot c\rangle_3$	$\{a^xb^yc^z : (x+y=3)\wedge(y+z=3)\}$

The easy part, the construction of automata from expressions, is illustrated in Figure 9.2. The construction for $\langle\varphi\rangle_I$ is done be adding a new clock c, and adding the condition $c\in I$ to any transition leading to an accepting state. For the difficult direction, Asarin manged to give, after a long journey of clock separation and other tricks, a McNaughton-Yamada-style proof of the fact that every AD automaton is equivalent (modulo renaming) to a timed regular expression [**P20**]. The use of intersection is unavoidable: the language denoted by the expression $\langle a\cdot b\rangle_3\cdot c\wedge a\cdot\langle b\cdot c\rangle_3$, which is recognized by a simple TA, cannot be expressed without intersection. The open problem concerning the necessity of renaming was solved later by Ph.

Herrmann. In a subsequent paper Asarin defined another concatenation operator, which corresponds to a transition without a reset, and used it to give an alternative proof using language equations.

References

E. Asarin, Equations on Timed Languages, in T. Henzinger and S. Sastry (Eds.), *Hybrid Systems: Computation and Control*, 1-12, LNCS 1386, Springer, 1998.

Z. Chaochen, C.A.R. Hoare, and A.P. Ravn, A Calculus of Durations, *Information Processing Letters* 40, 269–276, 1991.

J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

Ph. Herrmann, Renaming is Necessary in Timed Regular Expressions, *proc. FST&TCS'99*, 47-59, LNCS 1738, Springer, 1999.

S.C. Kleene, Representations of Events in Nerve Nets and Finite Automata, in C.E. Shannon and J. McCarthy (Eds.), *Automata Studies*, 3–42, Princeton University Press, 1956.

R. McNaughton and H. Yamada, Regular Expressions and State Graphs for Automata, *IRE Trans. Electronic Computers* EC-9, 39–47, 1960.

Th. Wilke, Specifying State Sequences in Powerful Decidable Logics and Timed Automata, in H. Langmaack et al (Eds.), *Proc. FTRTFT'94*, 694-715, LNCS 863, Springer, 1994.

My Papers

[**P20**]

E. Asarin, O. Maler and P. Caspi, A Kleene Theorem for Timed Automata, in G. Winskel (Ed.), *Proc. LICS'97*, 160-171, 1997.

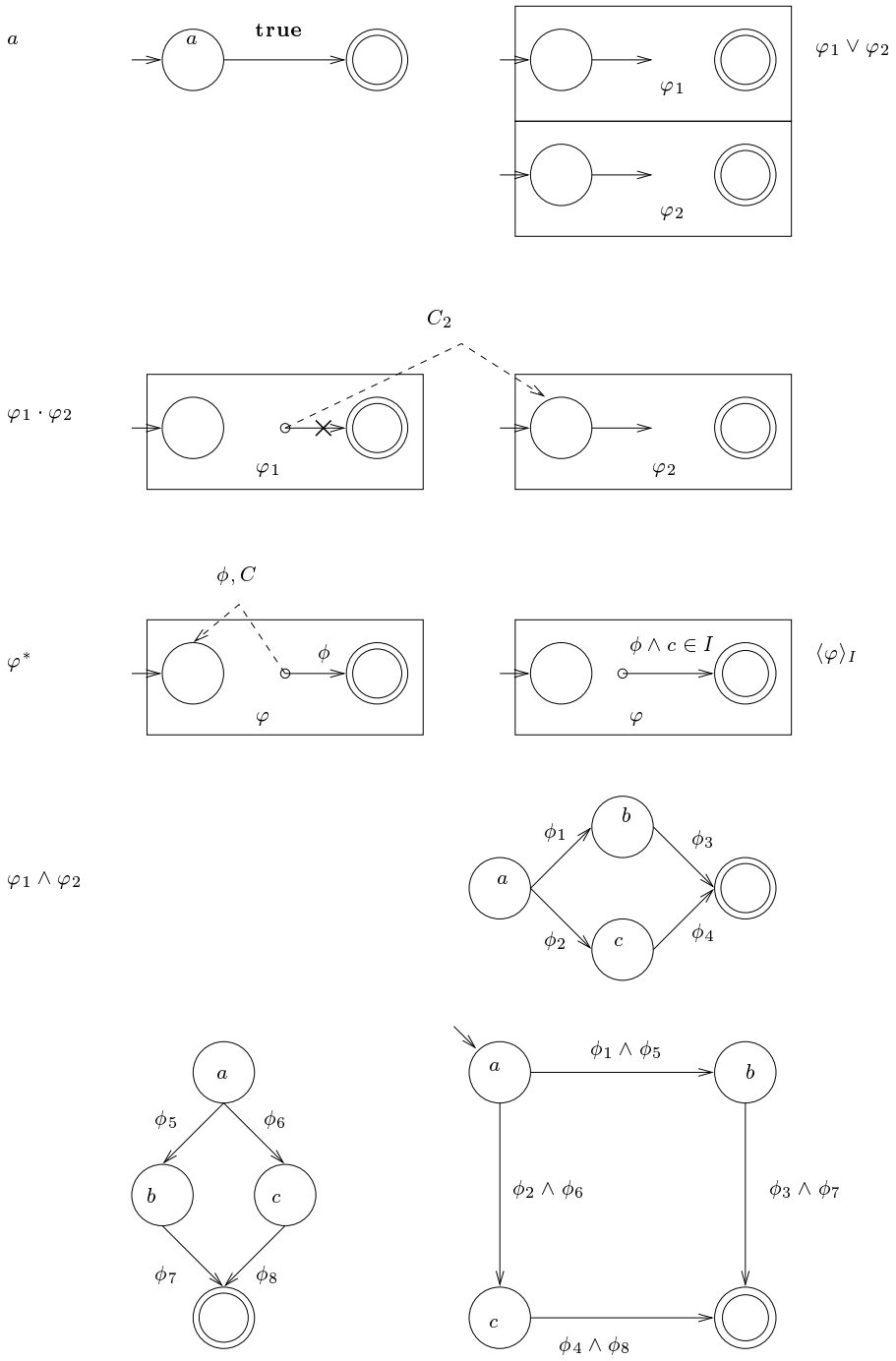

Figure 9.2: Constructing automata from expressions. Transitions annotated by C indicate that the set C of clocks is reset to zero.

Chapter 10

Probabilistic Systems

During my post-doc at Rennes, I took advantage of the proximity of experts in stochastic systems and learned about Markov chains (probabilistic automata with a single-letter input alphabet). My first reaction to works on the formal approach to real-time systems was to investigate what happens to an automaton which is too slow to cope with the inputs, and which might miss some events from time to time. This led to a model of an "ideal" deterministic automaton $\mathcal{A}$ compared with a noisy version $\mathcal{A}'$ of itself, which might in some probability take the wrong transition. The joint behavior of the two automata is captured by their product and the "error" states are the non-diagonal states of the product where $\mathcal{A}$ and $\mathcal{A}'$ disagree (see Figure 10.1 for a simplified example without an input alphabet). If you assume a probability over the inputs letters, you can transform the product into a Markov chain and compute the stationary probability of the states. I coded some examples in MATLAB, raised them to high powers and noticed that in some automata the stationary probability of error states goes to zero while in others it is very high. After some inspection, together with B. Delyon, we proved that automata having at least one reset transformation are much more immune to noise, while those which consist of permutations become very noisy. My personal problem with such domains (complexity, combinatorics) is that I do not find much connection between the intuition behind the results and the techniques used in proving them (i.e. manipulation of inequalities). Later this turned out to be my first journal paper [**P21**].

In another application of my automata-theoretic point of view, I realized that every Markov chain can be decomposed into a cascade product of a Bernoulli (memoryless) process and a *deterministic* automaton which reads the output of the process. In other words, memory and probability can be separated (see Figure 10.2). This result is really trivial if you use the vacabulary of communicating automata but can be rather surprising for probabilists. I used it to show that the Krohn-Rhodes decomposition can be applied to Markov chains [**P22**].

I have not touched probabilistic systems for several years until the visit of my friend Moshe Tennenholz in '98. During this visit we have updated each other in the respective developments in AI and verification, which took place since we were students. I told him of BDDs and symbolic model-checking and heard about the popularity of Markov Decision Processes (MDPs, games played over probabilistic

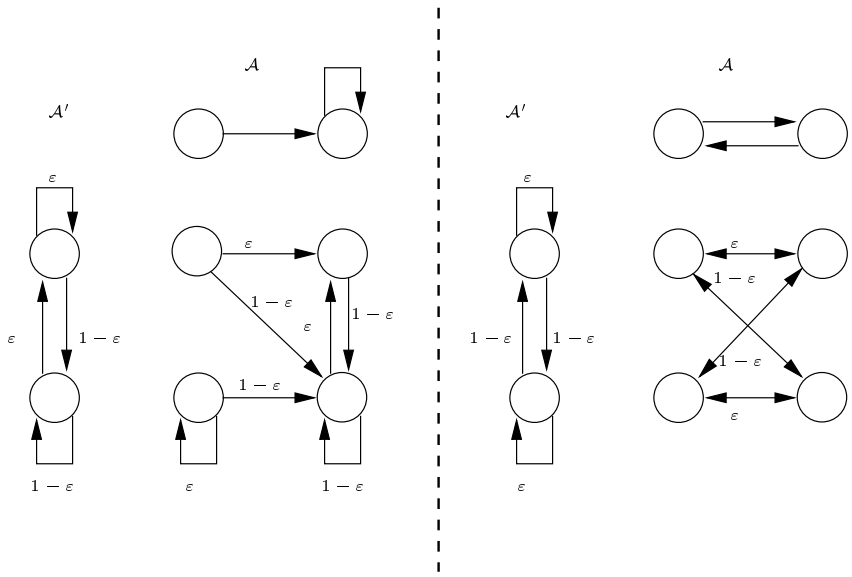

Figure 10.1: Left: a reset automaton $\mathcal{A}$, its noisy version $\mathcal{A}'$ and their product. In the long run the product automaton will be in high probability at a diagonal state. Right: the same model when $\mathcal{A}$ is a permutation automaton: in the long run the probability over the states is uniform, so $\mathcal{A}'$ will disagree with $\mathcal{A}$ half of the time.

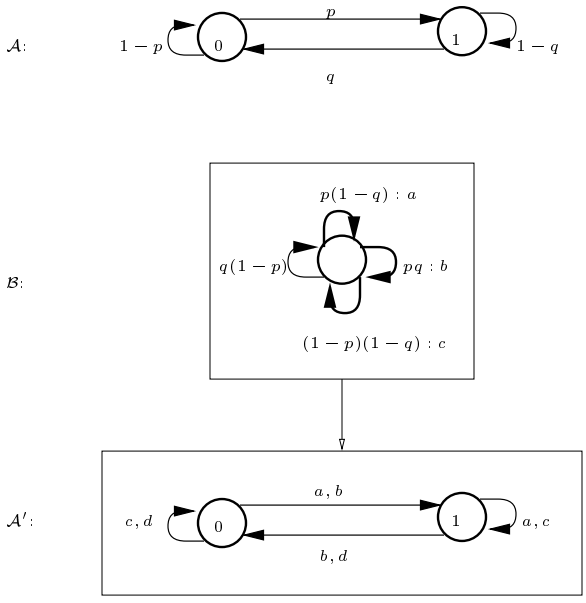

Figure 10.2: The Markov chain $\mathcal{A}$ is decomposed into the cascade $\mathcal{B} \circ \mathcal{A}'$.

automata) in modeling and solving problems of planning under uncertainty, and about attempts to represent them succintly using Bayesian networks (by the way, Bayesian networks were the subject of my Master's thesis during my pre-scientific days at the Business School). As a result of these discussions I started to think about what is needed in order to solve large-scale MDPs and export the BDD technology to probabilistic systems.

BDDs can represent Boolean functions of the form $f : \{0,1\}^n \to \{0,1\}$, and when there is little inter-dependence between the variables, the corresponding BDD is usually small. Several attempts have been made to extend BDD to represent probability functions of the form $p : \{0,1\}^n \to [0,1]$ using extensions of BDDs (known as ADD or MTBDD) which allow to put numbers on the leaves. However, one can construct examples where all the variables are independent yet the size of the MTBDD is exponential. Looking closer at the problem I realized that the right thing to do is to put probabilities on *all* the nodes (not only on the leaves) such that the probability of a Boolean vector is obtained by multiplying all the numbers in its corresponding path in the BDD. This amounts to using the *chain rule* for probabilities, phrased as:

$$p(x_1 x_2 \cdots x_n) = p(x_1) \cdot p(x_2|x_1) \cdots p(x_n|x_1 \cdots x_{n-1})$$

where $p(r|s)$ is the *conditional probability* of r given s. I called this structure Probabilsitic Decision Graphs (PDG). An example of a probability function over $\{0,1\}^3$ and its PDG appears in Figure 10.3.

The next thing to do was to find a representation for the transition matrix of Markov chains so that the basic "forward iteration" step $\mathbf{x}' := A\mathbf{x}$ for updating a PDG-represented probability vector can be efficiently performed. By looking closely at the way BDDs realize Boolean matrix multiplication, it was possible to define the appropriate data-structure, conditional PDG, and use it to perform a two-phase probabilistic matrix by vector multiplication. In general, I believe, similar techniques should be applied to other problems of analyzing the behavior of large-scale systems, currently dominated by matrix technology. Matrices, by their two-dimensional nature, force us to "flatten" structured domains, and only later, techniques based on sparseness try to exploit the (already deformed) structure. In verification of discrete systems, symbolic structured techniques allow us to treat systems whose explicit matrix representation is too large to be even written. Matrices and arrays are part of a pre-algebraic, or, at least pre-logical world-view which still dominates large parts of applied mathematics. I think one of the missions of computer scientists is to bring the gospel of logical terms, trees and data-structure to this A_{ijk} world, which has not yet completed the transition from Fortran to Algol and onwards.

During the long period between conception and submission of the paper [**P23**], I collaborated with Marius Bozga whose intimate knowledge of BDDs helped in concretizing the ideas, and who also implemented a prototype. We had some preliminary empirical results but there is still a lot to be done, especially in adapting the tool to specific classes of systems such as queues, noisy circuits, etc. Regardless of whether PDGs will prove useful, I like that paper very much.

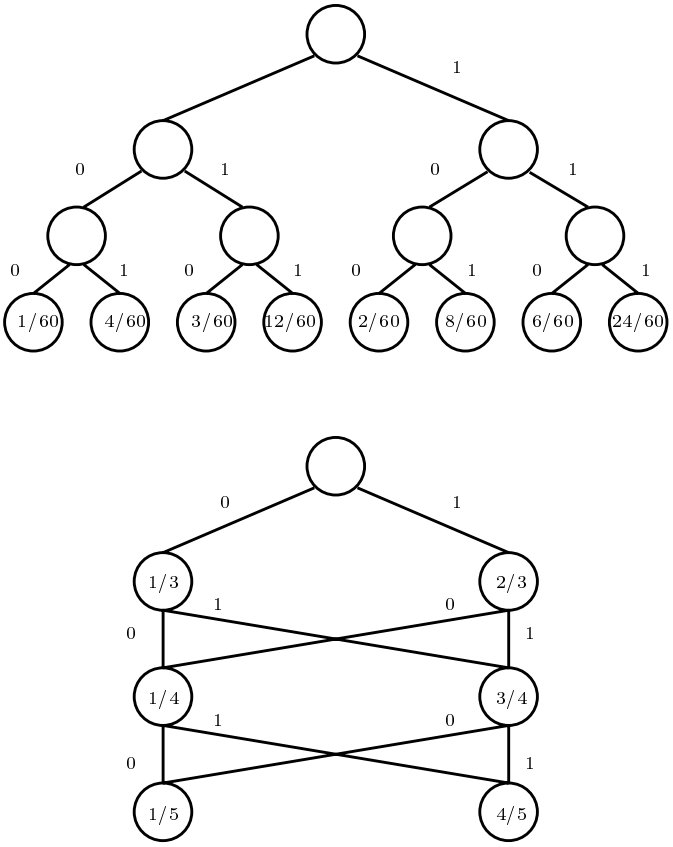

Figure 10.3: A probability function over $\{0,1\}^3$ represented as a tree and its corresponding PDG (whose size is linear because the variables are independent).

References

R.E. Bryant, Graph-based Algorithms for Boolean Function Manipulation, *IEEE Trans. on Computers* C-35, 677-691, 1986.

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, 142-70, 1992.

R.I. Bahar, E.A. Frohm, C.M. Ganoa, G.D. Hachtel, E. Macii, A. Pardo and F. Somenzi, Algebraic Decision Diagrams and their Applications, *Proc. ICCAD'93*, 188-191, 1993.

C. Baier, E. Clarke, V. Garmhausen-Hartonas, M. Kwiatkowska and M. Ryan, Symbolic Model Checking for Probabilistic Processes, in P. Degano, R. Gorrieri and A. Marchetti-Spaccamela (Eds.), *Proc. ICALP'97*, 430-440, LNCS 1256, Springer, 1997.

E. M. Clarke, M. Fujita, P. C. McGeer, K. L. Mcmillan and J. C.-Y. Yang, Multi-terminal Binary decision Diagrams: An Efficient Data-structure for Matrix Representation, *Proc. ILWS'93*, 1-15, 1993.

J.G. Kemeny and J.L. Snell, *Finite Markov Chains*, Van Nostrand, New York, 1960.

K.L. McMillan, *Symbolic Model-Checking: an Approach to the State-Explosion problem*, Kluwer, 1993.

C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*, Springer, 1998.

A. Paz, *Introduction to Probabilistic Automata*, Academic Press, New York, 1970.

J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.

M.L. Puterman, *Markov Decision Processes*, Wiley, 1994.

P. Tafertshofer and M. Pedram, Factored Edge-Valued Binary Decision Diagrams, *Formal Methods in system Design* 10, 137-164, 1997.

S.B.K. Vrudhula, M. Pedram and Y.-T. Lai, Edge-valued Binary Decision Diagrams, in T. Sasao and M. Fujita (Eds.), *Representations of Discrete Functions*, 109-132, Kluwer, 1996.

My Papers

[**P21**]

B. Delyon and O. Maler, On the Effects of Noise and Speed on Computations, *Theoretical Computer Science*, 129, 279-291, 1994. [results obtained in 1991]

[**P22**]

O. Maler, A Decomposition Theorem for Probabilistic Transition Systems, *Theoretical Computer Science*, 145, 391-396, 1995. [results obtained in 1992]

[**P23**]

M. Bozga and O. Maler, On the Representation of Probabilities over Structured Domains, in N. Halbwachs and D. Peled (Eds.), *Proc. CAV'99*, 261-273, LNCS 1633, Springer, 1999.

Chapter 11

Hybrid systems II

Meanwhile the international and inter-disciplinary activity around hybrid systems continued and several workshops took place, for some of which I share the blame [**P24**]. Mark Greenstreet visited us in the spring of '97 and told us about his method to approximate reachable sets of systems defined by differential equations. Consider a dynamical system defined by $\dot{\mathbf{x}} = f(\mathbf{x})$ and the problem of computing the states reachable from an initial polyhedron F. Due to continuity it is sufficient to look at boundary points (faces of F) and, more precisely, at faces from which f points outward. For every such face e, if you calculate the maximum of the normal of f, you can push e outward by a certain amount and obtain a new face e' such that all trajectories leaving e do not cross e' in time smaller than r. This way an over-approximation of all states reachable from F within the time interval $[0, r]$ is obtained (see Figure 11.1).

I liked the idea because it suggested a way to verify "real" continuous systems, not the simplified fixed-slope systems that we have tried to verify exactly in the past. My student Thao Dang started to work on this topic and as in the verification of timed automata, the problem of treating high-dimensional non-convex polyhedra had to be solved. We have chosen to represent reachable states using orthogonal ("griddy") polyhedra which are unions of hyper-cubes. Thao implemented the method (which we called "face-lifting") and we reported first results in the hybrid systems workshop at Berkeley [**P25**]. I think this paper, in which the role of reachability-based techniques in continuous verification was explained from first principles, contributed to the defintion of a concrete and feasible direction for hybrid systems research. Later I wrote a short pamphlet which showed the commonalities and differences between discrete and continuous systems and their simulation, verification and synthesis problems [**P26**].

The face-lifting approach turned out to be very computation-intensive and following a suggestion of E. Asarin, we started to look at methods specialized for linear systems of the form $\dot{\mathbf{x}} = A\mathbf{x}$. Our method is illustrated using Figure 11.2. Suppose we want to calculate the successors, via continuous evolution, of a convex set $F = conv(\mathbf{V})$ given by a set of vertices $\mathbf{V}$ (in the two-dimensional example $\mathbf{V} = \{\mathbf{x}_1, \mathbf{x}_2\}$). Let $\delta_r(F)$ denote the successors of F after exactly r time. Due to the fact that in linear systems $\delta_r(conv(\mathbf{V})) = conv(\delta_r(\mathbf{V}))$, one can compute $\delta_r(F)$ from $\mathbf{V}$, either by a finite number of numerical integration, or by matrix exponen-

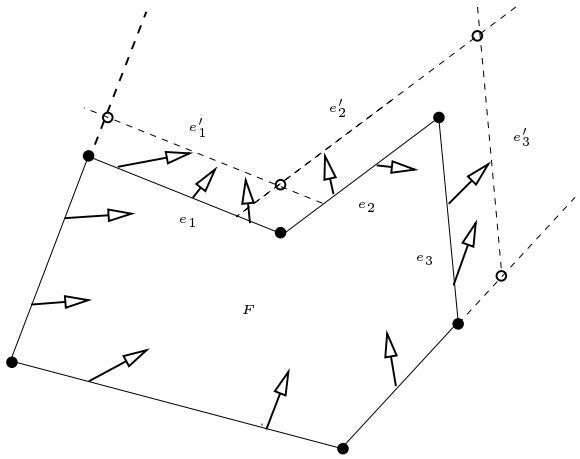

Figure 11.1: One step of the face-lifting procedure. The arrows illustrate the direction of f. Only edges e_1, e_2 and e_3 have a positive outward component of f and they are pushed into e'_1, e'_2 and e'_3.

tiation (Figure 11.2-a). In order to approximate the successors during the whole interval $[0, r]$ we take the convex hull of $\mathbf{V} \cup \delta_r(\mathbf{V})$ (Figure 11.2-b) and then push the faces outward in order to obtain an over-approximation (Figure 11.2-c). This polyhedron is then over-approximated by an orthogonal polyhedron (Figure 11.2-d). The computation of successors in the interval $[r, 2r]$ is done in a similar way starting from the points $\delta_r(\mathbf{V})$ (Figure 11.2-e) and the obtained set is merged into the set of states reachable so far. The advantage of this approach is that over-approximation errors in the sets do not accumulate (the next step is started from the real successors at time r) and that the set of reachable states is maintained as a single canonical orthogonal polyhedron, and not as a union of polyhedra. This faclitates termination detection and extension to hybrid systems where you need to compute intersections with guards and invariants.

This algorithm, along with various variations and extensions, such as backward reachability, under-approximation, treatment of hybrid systems, convex differntial inclusion and controller synthesis, has been implemented by Thao and tested over several examples [**P27**]. Of particular practical interest is the problem of synthesizing switching controllers, that is, devices that switch betwen several continuous modes in order to satisfy some reachability properties. This problem has been treated in [**P28**], where we took the "classical" controller synthesis algorithm, and used this technique for implelenting an approximate π operator. The description of the approach was preceded by some contemplations on the limitations of continuous mathematics, not always to the taste of some orthodox mathematicians, whitnessed by this passage from an anonymous referee report: *"The rest of the paper concerns the philosophy of continuous mathematics and control. Given that these philosophical remarks deserve to be exposed in a French cafe at best, but not in a world class journal, I cannot recommend this paper for publication ..."*

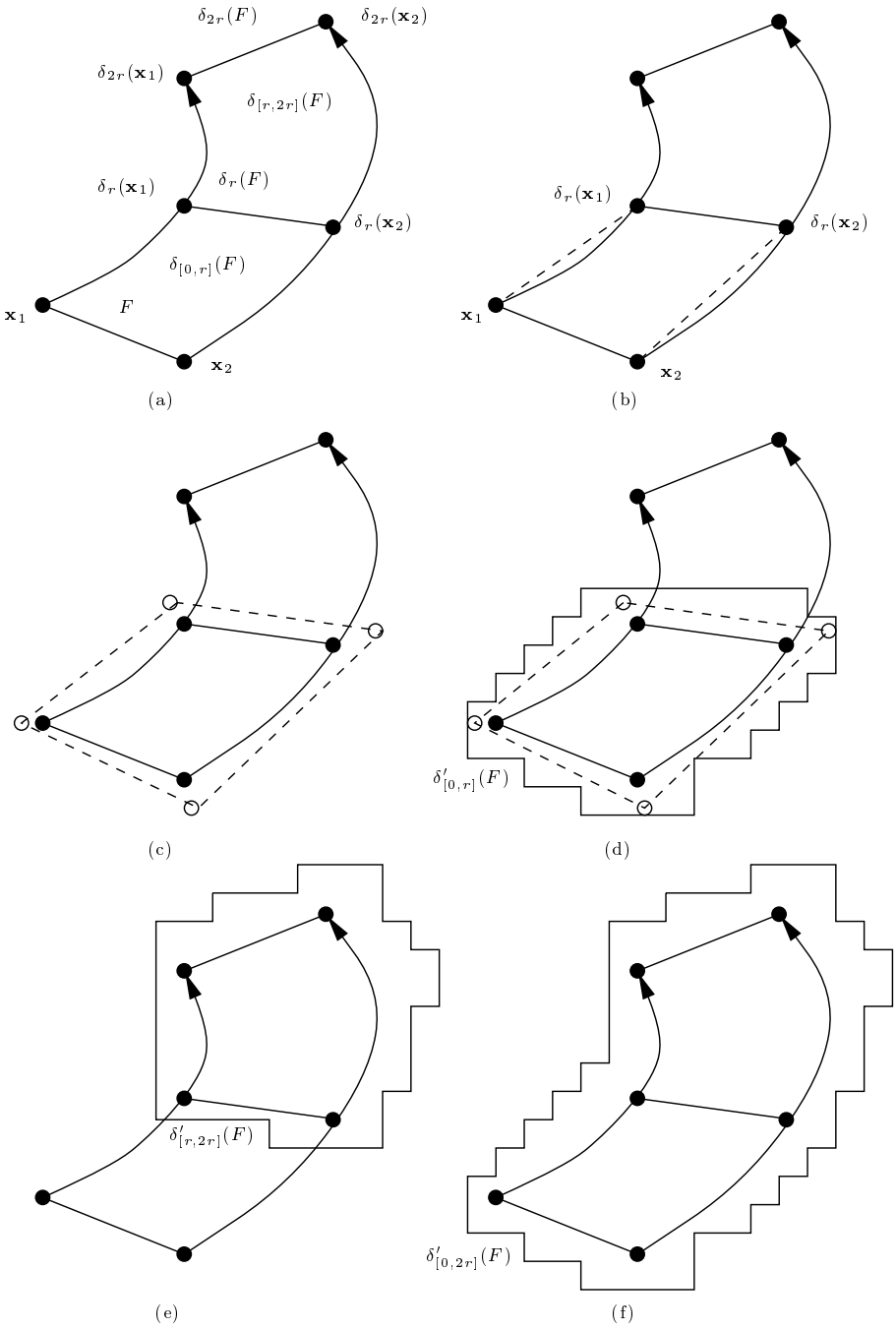

Figure 11.2: Iterative approximate computation of reachable states for a continuous two-dimansional linear system.

References

A. Chutinan and B.H. Krogh, Verification of Polyhedral Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations, in F. Vaandrager and J. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, 76-90, LNCS 1569, Springer, 1999.

M.R. Greenstreet, Verifying Safety Properties of Differential Equations, in R. Alur and T.A. Henzinger (Eds.), *Proc. CAV'96*, 277-287, LNCS 1102, 1996.

R. Isaacs, *Differential Games : A Mathematical Theory With Applications to Warfare and Pursuit, Control and Optimization*, Wiley, 1965.

C. Tomlin, J. Lygeros and S. Sastry, Controllers for Reachability Specifications for Hybrid Systems, *Automatica* 35, 349-370, 1999.

My Papers

[**P24**]
O. Maler (Ed.), *Hybrid and Real-Time Systems, International Workshop HART'97*, LNCS 1201, Springer, 1997.

[**P25**]
T. Dang, O. Maler, Reachability Analysis via Face Lifting, in T.A. Henzinger and S. Sastry (Eds), *Hybrid Systems: Computation and Control*, 96-109, LNCS 1386, Springer, 1998.

[**P26**]
O. Maler, A Unified Approach for Studying Discrete and Continuous Dynamical Systems, *Proc. CDC'98*, IEEE, 1998.

[**P27**]
E. Asarin, O. Bournez, T. Dang and O. Maler, Reachability Analysis of Piecewise-Linear Dynamical Systems, in B. Krogh and N. Lynch (Eds.), *Hybrid Systems: Computation and Control*, 20-31, LNCS 1790, Springer, 2000.

[**P28**]
E. Asarin, O. Bournez, T. Dang, O. Maler and A. Pnueli Effective Synthesis of Switching Controllers for Linear Systems, *Proc. of the IEEE*, to appear, 2000.

Chapter 12

Representation of Polyhedra

The representation of non-convex polyhedra turned out to be a recurring problem in the analysis of timed and hybrid systems. It is well-known that convex polyhedra can be represented by their vertices but no such representation is known for non-convex sets. Motivated by our work on reachability analysis of continuous systems, I started to look at "griddy" polyhedra (orthogonal, isothetic), those that can be written as unions of full-dimensional hyper-cubes taken from a fixed grid. Orthogonality and full-dimensionality already buy you some peace of mind by removing some complications which make the analysis of arbitrary polyhedra (even convex ones) hard. For simplicity of presentation we assume that we work with an integer grid in a bounded domain.

One way to represent griddy polyhedra is to use Boolean combinations of inequalities of the form $x_i \leq c$. An alternative approach is to use an n-dimensional Boolean array over $\{0, 1\}$ where each entry indicates whether the corresponding unit cube belongs to the set. The size of this array grows exponentially with the dimension, a fact usually not appreciated by computer scientists. I recalled one of the few excercises I tried to follow in a "data-structures" course (taken 20 years ago) concerning the representation of sparse matrices using linked lists. Playing a bit with this idea, I realized that in analyzing geometrical forms it is important to represent places where the value (in this case, the membership in the set) changes. This principle is used in various data compression schemes. To avoid confusion I fixed the correspondence between grid points and cubes by associating with every cube its *leftmost* corner. Trivial as it is, this decision helped me to see things in a more orderly fashion as every polyhedron could be viewed as a (color) function from grid points to $\{0, 1\}$ (see Figure 12.1).

After returning from a summer visit at Berkeley, where these ideas developed (partly due to discussions about PDEs and their numerical solutions) it occurred to me that the set of vertices together with their colors might be sufficient for defining *any* griddy polyhedron, that is, a set of black and white points as in Figure 12.2 might provide *all* the relevant information concerning the polyhedron in Figure 12.3. In order to show that this is indeed the case, one needs to find a decision procedure for the membership problem of a point/cube in a polyhedron based on this representation. By specializing the definitions of facets and vertices to orthogonal polyhedra, I could show that if a point $\mathbf{x} = (x_1, \ldots, x_n)$ is not a vertex, its

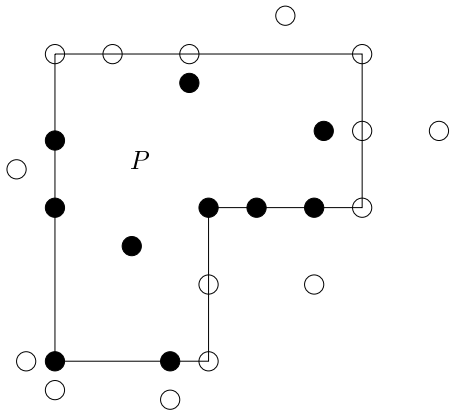

Figure 12.1: A polyhedron and a sample of the color function it induces over the grid points.

color can be computed from the colors of the points in its back-neighborhood, i.e. $\{x_1 - 1, x1\} \times \ldots \times \{x_n - 1, x_n\} - \{\mathbf{x}\}$. Assuming that the domain is bounded from below, this gives a recursive and terminating procedure for computing the color of any point.

I told my findings to Asarin ("I don't believe you" was his first reaction) and to Pnueli, who immediately had suggestions to improve the algorithm. The development accelerated significantly when Olivier Bournez came to pass his last year of thesis at VERIMAG. He managed to design and implement more efficient algorithms for membership, Boolean operations, projection, etc., to give proofs of correctness in any dimension as well as complexity bounds. Later, inspired by results of Aguilera and Ayala on three-dimensions, he proved that only a subset of the vertices (the "extreme" vertices, those that have an odd number of black points in their neighborhood) are needed to represent any griddy polyhedron in any dimension [**P29**]. Recently we have understood that this corresponds to the representation of the polyhedron as a XOR of rectangular *cones*.

Later we started to look at the extension of this technique to timed polyhedra which are unions of a special type of simplices associated with grid points and permutations. It took me some time to become the first person (I believe) in the TA research community who bothered drawing the equivalence classes of the region graph in more than 2 dimensions (Figure 12.4). For sets which are unions of such simplices,[1] Olivier managed to prove in his thesis that a canonical representation based on extreme vertex-permutation pairs exists as well. These results will be published in [**P30**] and are currently under implementation.

References

A. Aguilera and D. Ayala, Orthogonal Polyhedra as Geometric Bounds, in Constructive Solid Geometry, *Proc. Solid Modeling 97*, 1997

[1]Timed simplices are an instance of what is known as the Kuhn triangulation of the unit cube.

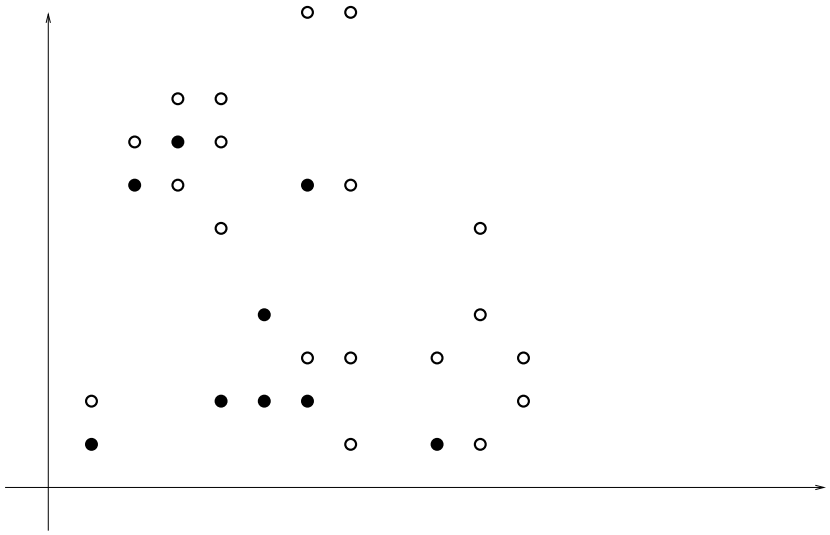

Figure 12.2: A set of colored vertices.

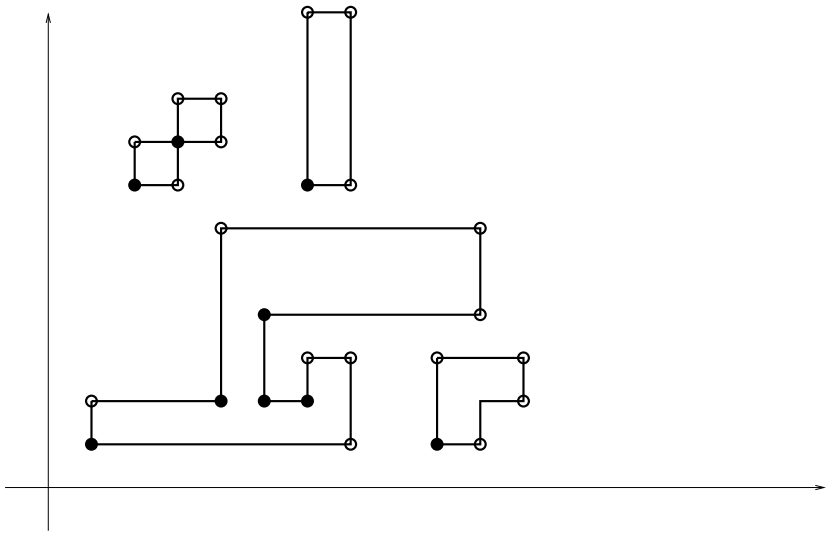

Figure 12.3: The polyhedron represented by the colored vertices of Figure 12.2.

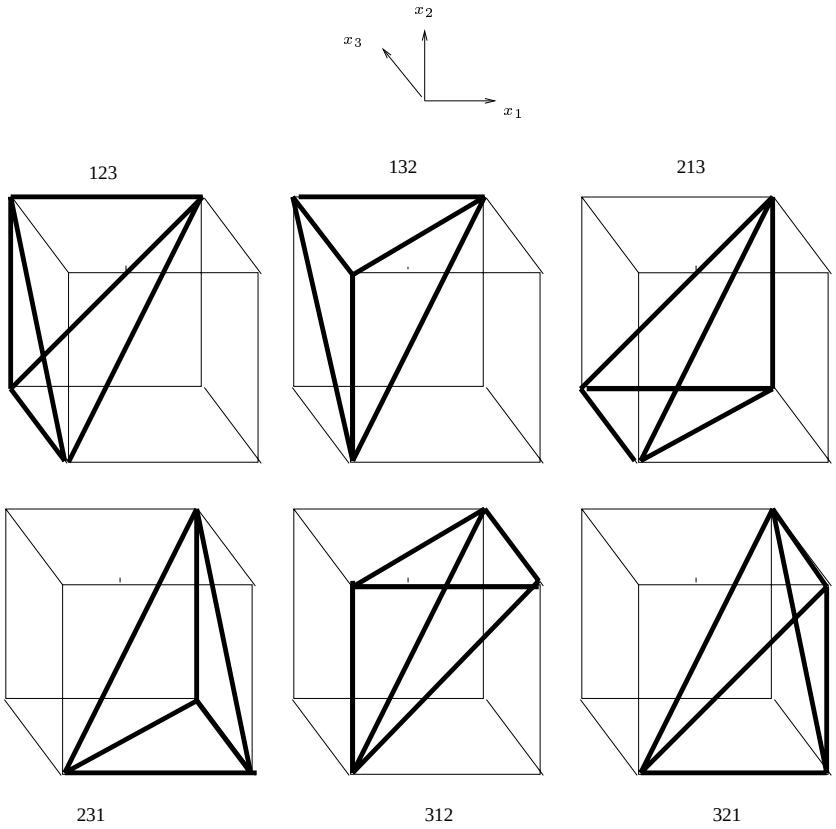

Figure 12.4: A decomposition of the unit hyper-cube into 6 simplices, each assocaited with a permutation (ijk) and an inequality $x_i \leq x_j \leq x_k$.

A. Aguilera and D. Ayala, Domain Extension for the Extreme Vertices Model (EVM) and Set-membership Classification, *Proc. Constructive Solid Geometry 98*, 1998.

My Papers

[**P29**]
O. Bournez, O. Maler and A. Pnueli, Orthogonal Polyhedra: Representation and Computation, in F. Vaandrager and J. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, 46-60, LNCS 1569, Springer, 1999.
[**P30**]
O. Bournez and O. Maler, On the Representation of Timed Polyhedra, *Proc. ICALP 2000*, to appear.