Timing Analysis of Asynchronous Circuits using Timed Automata*

Oded Maler[1] Amir Pnueli[2]

[1] SPECTRE – VERIMAG, Miniparc-ZIRST, 38330 Montbonnot, France,
Oded.Maler@imag.fr
[2] Dept. of Computer Science, Weizmann Inst. Rehovot 76100, Israel,
amir@wisdom.weizmann.ac.il

Abstract. In this paper we present a method for modeling asynchronous digital circuits by timed automata. The constructed timed automata serve as "mechanical" and verifiable objects for asynchronous sequential machines in the same sense that (untimed) automata do for synchronous machines. These results, combined with recent results concerning the analysis and synthesis of timed automata provide for the systematic treatment of a large class of problems that could be treated by conventional simulation methods only in an ad-hoc fashion. The problems that can be solved due to the results presented in this paper include: *the reachability analysis of a circuit with uncertainties in gate delays and input arrival times*, *inferring the necessary timing constraints on input signals that guarantee a proper functioning of a circuit* and *calculating the delay characteristics of the components required in order to meet some given behavioral specifications.*
Notwithstanding the existence of negative theoretical results concerning the worst-case complexity of timed automata analysis algorithms, initial experimentation with the KRONOS tool for timing analysis suggest that timed automata derived from circuits might not be so hard to analyze in practice.

1 Introduction

Digital circuits can be viewed at various levels of abstraction. This paper is concerned with the level situated between the transistor differential model and the purely-discrete model of synchronous sequential machines. At this intermediate level, the underlying continuous dynamics of the gates is not completely ignored, but rather encapsulated into real-time constraints that serve as an approximation of this dynamics. Unlike the synchronous modeling style where time is abstracted away into a sequence of points where nothing exists between them, time is viewed here as a continuous entity whose progress interferes with discrete state transitions.

* This research was supported in part by the European Community projects BRA-REACT(6021) and HYBRID EC-US-043. VERIMAG is a joint laboratory of CNRS, INPG, UJF and VERILOG SA. SPECTRE is a project of INRIA.

At this intermediate level of *timed Boolean functions*[3] the primary objects are *Boolean signals* defined over the *real* time axis, unlike *Boolean sequences* defined over the integers. In this model the output of a gate is a combinatorial function of the inputs, shifted in time. These delays could have been inferred from the differential dynamics of the components, but we will not be concerned with these low-level details (unlike [KM91]) in this paper and consider them as given.

We will present a fairly general model of asynchronous digital circuits consisting of Boolean gates and delay elements and show how this model translates naturally into the timed automata formalism of [AD94]. After this translation timed automata techniques can be applied to the analysis of the circuits (this was, in fact, the primary motivation for the introduction of timed automata in [D89]).

The main advantage of this formalism is that it allows automatic analysis of *all* the possible behaviors of the circuit.[4] The novel feature of these analysis methods compared to more conventional simulation techniques currently employed in timing analysis, is that it can capture uncertainties in the input arrival times, in the initial conditions or in the delay parameters of the gates, without any problem. This is because the "simulation" is global in the sense that instead of simulating *one* possible execution of the circuit, we simulate in one "step" an infinite (and even uncountable) number of executions (see also [BM83], [L89], [D89], [BD91], [AD94] for the origins of this "geometric" simulation method for timed systems, and [ACH+95] [AMP95-a] for the application of this approach in the more general setting of *hybrid systems*).

The core of this paper is a careful translation of circuits, defined via a system of delay equations into a network of interacting timed automata whose set of possible behaviors is exactly the set of signals satisfying the equations. The translation is done using two types of basic components and it reflects the structural properties of the circuit, including the functional and temporal dependencies between the state-variables. As such it can serve as a basis for further optimizations and algorithmic fine-tuning.

The rest of the paper is organized as follows: in section 2 we present signals, delay equations and circuits. Section 3 consists of a presentation of a modified version of timed automata communicating via shared variables, which we find to be the most suitable for circuit modeling. In section 4 we show how to translate between the two models and conclude in section 5 with the potential applications of these results.

2 Signals and Circuits

Let T denote the set of non-negative reals and let $\mathcal{Q}$ be any set.

[3] We adopt the term, but not the techniques, which are essentially deterministic, from [LB94]. In fact, our formalism could be called *timed Boolean relations*.

[4] Some recent simulation-oriented attempts to achieve this goal are reported in [ML93], [LL94].

Definition 1 (Piecewise-continuous Signals). *A $\mathcal{Q}$-valued piecewise continuous signal is a function $\alpha : T \rightarrow \mathcal{Q}$ admitting a (possibly finite) countable increasing sequence $\mathcal{L}(\alpha) = t_0, t_1, \ldots$ such that $t_0 = 0$ and α is continuous at $T - \mathcal{L}(\alpha)$ and discontinuous at $\mathcal{L}(\alpha)$.*

We use α_t to denote $\alpha(t)$ and let $I(\alpha) = I_0, I_1, \ldots = [t_0, t_1), [t_1, t_2), \ldots$ be the sequence of left-closed right-open intervals induced by the signal. We call $\mathcal{L}(\alpha)$ the *boundary points* of α. Continuous signals are obtained as a special case when $\mathcal{L}(\alpha) = 0, \infty$ and $I(\alpha) = [0, \infty)$.

Let $\mathbb{B} = \{0, 1\}$. A Boolean signal is a $\mathbb{B}^k$-valued signal for some k. In this case the above properties of signals specialize into:

- $t_1, t_2 \in I_i \Rightarrow \alpha_{t_1} = \alpha_{t_2}$,
- $t_1 \in I_i \wedge t_2 \in I_{i+1} \Rightarrow \alpha_{t_1} \neq \alpha_{t_2}$

One can see that the conditions above prevent a non-countable number of discrete variations in the value of the signal as well as the so-called *Zeno phenomenon* in which infinitely many discrete transitions happen within a bounded real-time interval. We denote the set of all such Boolean signals by S^k.

A Boolean function is a function $f : \mathbb{B}^k \rightarrow \mathbb{B}$ for some $k \geq 0$. We will use the same notation for the temporal extension, $f : S^n \rightarrow S$, of f, defined as $\beta = f(\alpha)$ iff for every $t \in T$, $\beta_t = f(\alpha_t)$. We call this an instantaneous signal function.

Definition 2 (Ideal Deterministic Delay). *Let d be a non-negative number. The ideal delay associated with d is a function $\Delta_d : \mathbb{B} \times S \rightarrow S$ such that $\beta = \Delta_d(b, \alpha)$ iff for every $t \in T$:*

$$\beta_t = \begin{cases} b & \text{if } t < d \\ \alpha_{t-d} & \text{if } d \leq t \end{cases}$$

The value of β always mirrors the value of α as it was d time units before and b is a default value of β for the initial interval $[0, d)$ – see signals s_1 and s_3 in figure 1. Ideal delays are nice mathematically but are not comfortable for finite-state modeling (and are not physically realistic). This is because β has always to "remember" all the possible variations in the value of α that could have occurred in the last temporal window of length d. It is common to assume that every change in α has to persist for a minimal interval of time (*latency*)[5] in order to be "noticed" by the delay element. In order to simplify the presentation we unify these two constants and assume that the latency associated with Δ_d is equal to d. More generally it could be any number not greater than d (otherwise the function becomes non-causal as the value of β at time t might depend on the value of α at time greater than t).

Definition 3 (Deterministic Latency Delay). *Let d be a non-negative number. the latency delay associated with d is a function $\Delta_d : \mathbb{B} \times S \rightarrow S$ such that $\beta = \Delta_d(b, \alpha)$ iff*

[5] Also called "inertial delay", see, for example, the survey [BS91].

1. $\beta_t = b$ *for every* $t \in [0, d)$ *and*
2. *For every* $t \geq d$, $t \in \mathcal{L}(\beta)$ *iff* $t-d \in \mathcal{L}(\alpha)$, $(t-d, t) \cap \mathcal{L}(\alpha) = \emptyset$ *and* $\alpha_{t-d} = \beta_t$

The condition $(t-d, t) \cap \mathcal{L}(\alpha) = \emptyset$ ensures that the change that took place at $t-d$ persisted for d time units. This definition does not refer directly to the values of β at every t, but rather indirectly using $\mathcal{L}(\beta)$. From the definition it follows that if some t is not in $\mathcal{L}(\beta)$ then the value of β at t is the value of β at the latest boundary point (which could be as well the point $t = 0$ if no change in α persisted long enough since the beginning). Every non-ideal delay can be decomposed into a "d-filter" (an element that ignores variations that persist less than d), and an ideal delay – see signals s_1, s_2 and s_4 in figure 1.

Remark: In certain physical settings the effects on β of high-frequency variations in α is not predictable. Consequently the value of β in the corresponding instants can be any value and the delay operator is non-deterministic. We have chosen a "lazy" version of the latency delay such that no state-transition takes place unless it must. The suitability of this modeling decision is application-dependent and our theory could be developed under different assumptions.

Delay characteristics of real components cannot be known precisely. The most one can expect from a specification of such components is a delay interval $[l, u]$ expressing lower and upper-bounds on the time it takes for a change in the input to propagate to the output. This motivates the following definition:

Definition 4 (Non-Deterministic Delay). *Let* l *and* u *be two non-negative numbers such that* $l \leq u$. *The non-determinisitic delay associated with* l, u *is a function* $\Delta_{[l,u]} : \mathbb{B} \times S \rightarrow 2^S$ *defined as:* $\beta \in \Delta_{[l,u]}(b, \alpha)$ *iff*

1. $\beta_t = b$ *for every* $t \in [0, l)$,
2. *For every* $t \geq l$, $t \in \mathcal{L}(\beta) \Rightarrow \exists t' \in \mathcal{L}(\alpha) \cap [t-u, t-l]$ *such that* $\alpha_{t'} = \beta_t$ *and* $(t', t'+l) \cap \mathcal{L}(\alpha) = \emptyset$. (Every change in β must be preceded by an l-persistent change in α).
3. *For every* t', $t' \in \mathcal{L}(\alpha) \wedge (t', t+u) \cap \mathcal{L}(\alpha) = \emptyset \Rightarrow (t'+l, t'+u) \cap \mathcal{L}(\beta) \neq \emptyset$. (Every u-persistent change in α must be reflected in β).

All these notions are depicted in figure 1. Non-deterministic delays pose problems to traditional simulation methods as the next "event" in the simulation can take place anywhere within an interval.

Definition 5 (Circuit). *A* k*-wire digital circuit is a tuple* $\mathcal{N} = (X, \mathcal{F}, D)$ *where* $X = \{x_1, \ldots, x_k\}$ *is a set of wires,* $\mathcal{F} = \{f_1, \ldots, f_k\}$ *is a set of functions of the form* $f_i : \mathbb{B}^k \rightarrow \mathbb{B}$ *and* $D = \{(l_1, u_1), \ldots, (l_k, u_k)\}$ *is a set of positive pairs of integers such that* $l_i \leq u_i$. *A behavior of the circuit starting from an initial state* $\mathbf{b} = (b_1, \ldots, b_k)$ *is a* $\mathbb{B}^k$*-valued signal* $x = \langle x_1, \ldots, x_k \rangle$ *satisfying the system of simultaneous inclusions:*

$$\begin{array}{c} x_1 \in \Delta_{[l_1,u_1]}(b_1, f_1(x_1, x_2, \ldots, x_k)) \\ x_2 \in \Delta_{[l_2,u_2]}(b_2, f_2(x_1, x_2, \ldots, x_k)) \\ \ldots \\ x_k \in \Delta_{[l_k,u_k]}(b_k, f_k(x_1, x_2, \ldots, x_k)) \end{array} \quad (1)$$

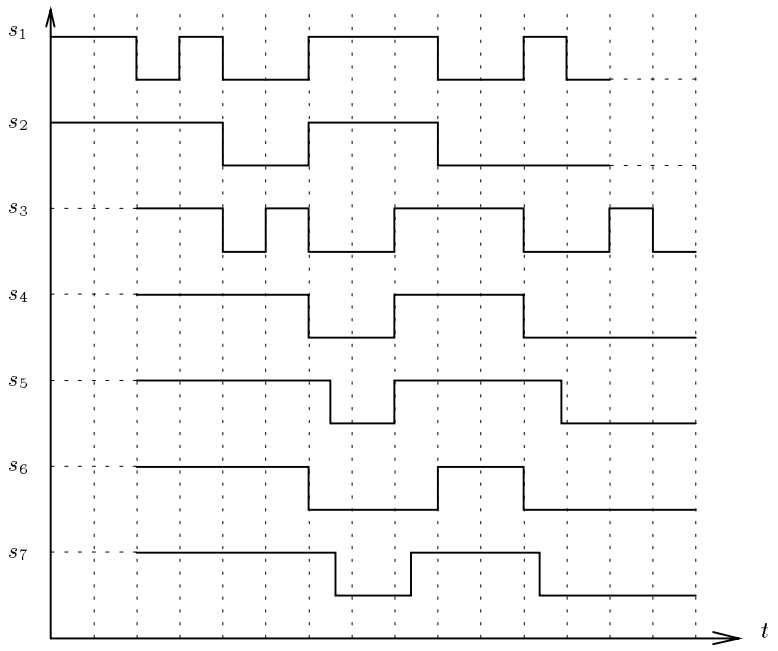

Fig. 1. A signal s_1, its 2-filtered version s_2, $s_3 = \Delta_2(s_1)$ (ideal), $s_4 = \Delta_2(s_1)$ (latency) and $\{s_4, s_5, s_6, s_7\} \subseteq \Delta_{[2,3]}(s_1)$.

A circuit appears in figure 2. Such a decomposition of gates into the combinatorial and the delay part is common (e.g., [LB94]). The correspondence between a circuit and the system of equations (1) is straightforward and we will refer to (1) as the description of the circuit. In practice gates have a limited fan-in and each f_i refers only to a small subset of the wires. However for proving our results it is simpler to assume that all functions are k-ary. The syntactic structure of $\mathcal{F}$ reflects the topology of the circuit and it will certainly play a role in any efficient implementation of analysis algorithms. In the sequel, in order not to drag with us too much notation, we will omit the reference to the initial value from the delay equations and their corresponding automata, and use equations of the form

$$x_i \in \Delta_{[l_i,u_i]}(f_i(x_1, x_2, \ldots, x_k)).$$

Needless to say, the system of equations (1) need not have unique solution. For the readers who wonder where have the *input* signals disappear in our model, the answer is that in a non-deterministic framework inputs can be treated as any other signal, having the property of being independent of other signals. For example, the assumption that the rising and falling of an input signal x_i must be separated by at least d time units, can be expressed as $x_i \in \Delta_{[d,\infty]}(\neg x_i)$. Similarly, $x_i \in \Delta_{[l,u]}(\neg x_i)$ specifies all signals the distance between their two

consecutive alternations is between l and u. An unconstrained input signal does not appear in the left-hand side of any equation.

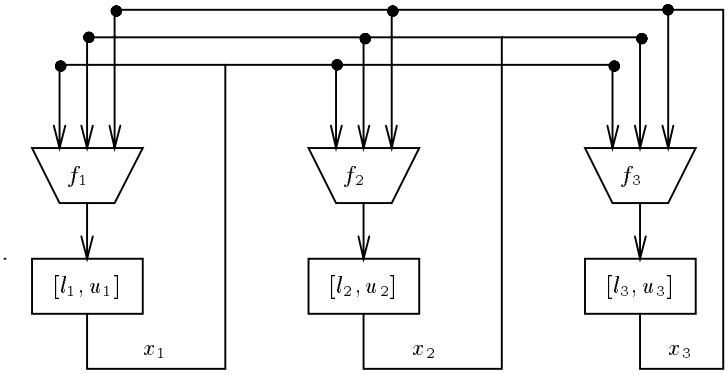

Fig. 2. A 3-wire circuit.

Before introducing timed automata let us try to explain intuitively how we model the functioning of a circuit. We use a variable $\mathbf{v}_i$ to represent the observable value of x_i, that is, the value at the output port of the delay element (see figure 2). We associate two additional *fictitious* variables v_i and C_i that reflect what happens between f_i and the delay: v_i is the current value of the output of f_i while C_i is a clock that measures the time since v_i obtained this value. Using hardware terminology, wire x_i is *excited* when $\mathbf{v}_i \neq v_i$. When the time is ripe, i.e. $C_i \geq l_i$, and $\mathbf{v}_i$ lags behind v_i, $\mathbf{v}_i$ can catch up and update its value. Since x_i might be connected via a feed-back loop to f_i, the change in $\mathbf{v}_i$ may change f_i and v_i and possibly invalidate the condition that enabled this very change in $\mathbf{v}_i$. Hence in order to avoid this instability we[6] must break the simultaneity and assume that every change in a $\mathbf{v}_i$ precedes (by an "infinitesimal") the changes it triggers in the various v_i's. This will become hopefully clearer in the sequel.

3 Timed Automata

Let $\mathcal{V} = \mathbf{V} \cup V$ be a set of Boolean variables and let $\mathcal{C}$ be a set of clock variables ranging over T. The variables in $V \cup \mathcal{C}$ are called *hidden* variables while those in $\mathbf{V}$ are called *observable* variables.

Definition 6 (Formulae). *A $(\mathcal{V}, \mathcal{C})$-formula is a Boolean combination of conditions of the form $v = 0$, $v = 1$, $C \leq c$, and $C < c$ for $v \in \mathcal{V}$, $C \in \mathcal{C}$ and*

[6] And anyone else attempting to model feed-back loops using sequential mathematics.

$c \in \{0, \ldots, h\}$ *for some integer* h (which we assume to be fixed throughout the paper).

We denote the set of all such formulae by $F(\mathcal{V}, \mathcal{C})$. Clearly, if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{C}' \subseteq \mathcal{C}$ then every $(\mathcal{V}', \mathcal{C}')$-formula is also a $(\mathcal{V}, \mathcal{C})$-formula.

Definition 7 (States and Transitions). *A* $(\mathcal{V}', \mathcal{C}')$*-state for* $\mathcal{V}' \subseteq \mathcal{V}$ *and* $\mathcal{C}' \subseteq \mathcal{C}$ *is a pair* $s = \langle q, r \rangle$ *of functions* $q : \mathcal{V}' \to \mathbb{B}$ *and* $r : \mathcal{C}' \to T$*, assigning actual values to every* $v \in \mathcal{V}'$*,* $C \in \mathcal{C}'$*. A* $(\mathcal{V}', \mathcal{C}')$*-transition is a pair* (s, s') *of* $(\mathcal{V}', \mathcal{C}')$*-states.*

Let $s = \langle q, r \rangle$, and $s' = \langle q', r' \rangle$ be two $(\mathcal{V}', \mathcal{C}')$-state. For every $v \in \mathcal{V}'$, $C \in \mathcal{C}'$ we use $s[v] = q(v)$ and $s[C] = r(C)$ to denote the interpretations that s gives to its variables. For a transition (s, s') we will use $(s, s')[v]$ to denote $s'[v]$ if $v \in \mathbf{V}$ and $s[v]$ otherwise. As we see the observable variables are interpreted according to their value *"after"* the transition while the hidden variables are interpreted *"before"* the transition. This can be viewed as giving a priority to the observed variables.

We will use $\mathcal{S} = \mathcal{Q} \times \mathcal{H}$ where $\mathcal{Q} = \mathbb{B}^{|\mathcal{V}|}$ and $\mathcal{H} = T^{|\mathcal{C}|}$ to denote the set of all global states, i.e. all $(\mathcal{V}, \mathcal{C})$-state. All the signals we will use henceforth are $\mathcal{S}$-valued. Such a signal $x : T \to \mathcal{S}$ induces for every $v \in \mathcal{V}$ (resp. $C \in \mathcal{C}$) an interpreted signal $x[v] : T \to \{0, 1\}$ (resp. $x[C] : T \to T$) which is almost the projection of x on v. It is defined for every t as

$$x[v]_t = \begin{cases} x_t[v] & \text{if } v \in \mathbf{V} \\ (\lim_{t' \to t, t' < t} x_{t'})[v] & \text{otherwise} \end{cases}$$

Note that the above two expressions are equal whenever x is continuous at t, and the distinction concerns only the points in $\mathcal{L}(x)$, where the hidden variables are interpreted according to their value "before" the discrete jump.

Definition 8 (Interpretation of Formulae). *Let* $\mathcal{R}$ *be a formula,* s, s' *be two* $(\mathcal{V}', \mathcal{C}')$*-state and* $x : T \to \mathcal{S}$ *be a signal. Then*

1. *The interpretation of* $\mathcal{R}$ *at* s*, denoted by* $s[\mathcal{R}]$*, is the formula obtained by substituting* $s[v]$ *in* v *for every* $v \in \mathcal{V}' \cup \mathcal{C}'$*.*
2. *The interpretation of* $\mathcal{R}$ *at* (s, s')*, denoted by* $(s, s')[\mathcal{R}]$ *is the formula obtained by substituting* $(s, s')[v]$ *in* v *for every* $v \in \mathcal{V}' \cup \mathcal{C}'$*.*
3. *The interpretation of* $\mathcal{R}$ *at* x*, denoted by* $x[\mathcal{R}]$*, is a signal* $x[\mathcal{R}] : T \to \{\mathbf{true}, \mathbf{false}\}$ *such that for every* t*,* $x[\mathcal{R}]_t$ *is obtained by substituting* $x[v]_t$ *in* $\mathcal{R}$ *for every* $v \in \mathcal{V} \cup \mathcal{C}$*.*

A $(\mathcal{V}', \mathcal{C}')$-state s transforms a $(\mathcal{V}, \mathcal{C})$-formula $\mathcal{R}$ into a $(\mathcal{V} - \mathcal{V}', \mathcal{C} - \mathcal{C}')$-formula $\mathcal{R}' = s[\mathcal{R}]$ and if all the variables mentioned in $\mathcal{R}$ are included in $\mathcal{V}' \cup \mathcal{C}'$, then $s[\mathcal{R}]$ is either **true** or **false** (which is always the case when s is global). We use a similar notation for the interpretation at a state s or a signal x of a subset of variables, e.g. $s[\mathcal{V}']$, $x[\mathcal{V}']$. If $\mathcal{V}' \subseteq \mathbf{V}$ then $x[\mathcal{V}']$ is exactly the projection of x on $\mathcal{V}'$. For a set L of signals, $L[\mathcal{V}']$ is the set of corresponding projected signals.

Example: Let $\mathcal{V} = \{v_1, v_2\}$, $\mathcal{C} = \{C\}$, and consider the states

$$s : \langle v_1 = 0, v_2 = 1, C = 5 \rangle \quad s' : \langle v_1 = 0 \rangle$$

and formulae

$$\mathcal{R}_1 : v_1 = 1 \vee C < 4 \quad \mathcal{R}_2 : v_1 = 0 \wedge C > 4.$$

Then $s[v_1] = 0$, $s[C] = 5$, $s[\mathcal{V}] = \langle 0, 1 \rangle$, $s[\mathcal{R}_1] = \mathbf{false}$, $s[\mathcal{R}_2] = \mathbf{true}$, $s'[\mathcal{R}_1] = C < 4$ and $s'[\mathcal{R}_2] = C > 4$.

Our model of timed automata, defined below, slightly differs from the original model of [AD94] and others in the following features:

1. The distinction we make between hidden and observable variables.
2. We allow communication between the automaton and the external world by means of continuously-present shared variables, instead of by "message-passing".
3. We employ a "dense" semantics (signals) instead of "sampled" semantics (steps).

It can be shown that these models are essentially equivalent.

Definition9 (Timed Automaton). *A timed automaton is a tuple* $\mathcal{A} = (\mathcal{V}_\mathcal{A}, \mathcal{C}_\mathcal{A}, \mathcal{R}, \mathcal{O})$ *where:*

- $\mathcal{V}_\mathcal{A} \subseteq \mathcal{V}$ *is a set of* m *Boolean variables.* We denote the set $\mathbb{B}^m$ of all their possible valuations by $\mathcal{Q}_\mathcal{A}$ and call it the *state-space* of $\mathcal{A}$.
- $\mathcal{C}_\mathcal{A} \subseteq \mathcal{C}$ *is a set of* n *clock variables.* We denote the set T^n of all their possible valuations by $\mathcal{H}_\mathcal{A}$ and call it the *clock space* of $\mathcal{A}$. The *configuration space* of $\mathcal{A}$ is $\mathcal{Q}_\mathcal{A} \times \mathcal{H}_\mathcal{A}$.
- $\mathcal{R} : Q_\mathcal{A} \times Q_\mathcal{A} \rightarrow F(\mathcal{V} - \mathcal{V}_\mathcal{A}, \mathcal{C})$ *is a function that assigns a formula (over the clocks and the variables which are "external" to* $\mathcal{A}$*) to each pair of states,*
- $\mathcal{O} : Q_\mathcal{A} \times Q_\mathcal{A} \rightarrow 2^{\mathcal{C}_\mathcal{A}}$ *is a function that assigns to every pair of states a subset of the clocks of* $\mathcal{A}$ (we require that $\mathcal{O}(q, q) = \emptyset$).

The intuitive meaning of this definition is as follows: the configuration-space of $\mathcal{A}$ consists of all possible valuations to its own variables and clocks – the rest of the clocks and variables are considered *external* to the automaton and thir values can take the form of arbitrary *signals*.[7] The internal clocks grow uniformly with time. The automaton can stay at some state q as long as the evaluation of $\mathcal{R}(q, q)$ on the clocks and the external variables remains **true**. Similarly, a transition from q to q' can be taken when $\mathcal{R}(q, q')$ evaluates to **true**. In this case, $\mathcal{A}$ resets all the internal clocks in $\mathcal{O}(q, q')$. The external clocks and variables can, thus, influence the behavior of $\mathcal{A}$, but only the values of clocks in $\mathcal{C}_\mathcal{A}$ and the variables in $\mathcal{V}_\mathcal{A}$ can be *modified* by $\mathcal{A}$.

[7] The notion of external/internal with respect to an automaton should not be confused with the notion of observable and hidden variables. An automaton can "own" both hidden and observable variables and can employ (other) variables of both sorts in its associated formulae.

Definition 10 (Semantics of Timed Automata). *Let $\mathcal{A} = (\mathcal{V}_\mathcal{A}, \mathcal{C}_\mathcal{A}, \mathcal{R}, \mathcal{O})$ be a timed automaton. The semantics of $\mathcal{A}$ consists of all signals $x : T \to \mathcal{S}$ such that:*

1. *T is partitioned into a sequence of intervals $[t_0, t_1), [t_1, t_2), \ldots$ such that $x_t[\mathcal{V}_\mathcal{A}]$ is constant inside every interval,*
2. *For every $i \geq 0$:*
   (a) $x[\mathcal{V}_\mathcal{A}]_{t_i} = q$ and $x[\mathcal{V}_\mathcal{A}]_{t_{i+1}} = q'$ implies
      - *$x[\mathcal{R}(q, q')]_{t_{i+1}} = \mathbf{true}$,*
      - *$x[C]_{t_{i+1}} = x[C]_{t_i} + (t_{i+1} - t_i)$ for every $C \in \mathcal{C}_\mathcal{A} - \mathcal{O}(q, q')$,*
      - *$x[C]_{t_{i+1}} = 0$ for every $C \in \mathcal{O}(q, q')$.*
   (b) If $x[\mathcal{V}_\mathcal{A}]_{t_i} = q$ then for every $t \in [t_i, t_{i+1})$:
      - *$x[\mathcal{R}(q, q)]_t = \mathbf{true}$,*
      - *$x[C]_t = x[C]_{t_i} + (t - t_i)$ for every $C \in \mathcal{C}_\mathcal{A}$.*

The set of all such signals is denoted by $L(\mathcal{A})$.

Condition (2-a) says that $\mathcal{A}$ makes a discrete transition from q to q' at time t only if the transition condition $\mathcal{R}(q, q')$ is true at t – in this case it resets the corresponding internal clocks. Condition (2-b) says that in order to stay at state q (and let the internal clocks grow) $\mathcal{R}(q, q)$ must be satisfied.
Note that the semantics is defined as a set of $\mathcal{S}$-valued signals including dimensions that correspond to variables in $\mathcal{V} - \mathcal{V}_\mathcal{A}$. This facilitates the following definition of composition.

Definition 11 (Composition of Timed Automata). *Let $\mathcal{A}_1 = (\mathcal{V}_{\mathcal{A}_1}, \mathcal{C}_{\mathcal{A}_1}, \mathcal{R}_1, \mathcal{O}_1)$ and $\mathcal{A}_2 = (\mathcal{V}_{\mathcal{A}_2}, \mathcal{C}_{\mathcal{A}_2}, \mathcal{R}_2, \mathcal{O}_2)$ be two timed automata such that $(\mathcal{V}_{\mathcal{A}_1} \cup \mathcal{C}_{\mathcal{A}_1}) \cap (\mathcal{V}_{\mathcal{A}_2} \cup \mathcal{C}_{\mathcal{A}_2}) = \emptyset$.[8] Their composition is $\mathcal{A}_1 \otimes \mathcal{A}_2 = \mathcal{A} = (\mathcal{V}_\mathcal{A}, \mathcal{C}_\mathcal{A}, \mathcal{R}, \mathcal{O})$ such that:*

1. *$\mathcal{V}_\mathcal{A} = \mathcal{V}_{\mathcal{A}_1} \cup \mathcal{V}_{\mathcal{A}_2}$ (and the state-space is $\mathcal{Q}_\mathcal{A} = \mathcal{Q}_{\mathcal{A}_1} \times \mathcal{Q}_{\mathcal{A}_2}$),*
2. *$\mathcal{C}_\mathcal{A} = \mathcal{C}_{\mathcal{A}_1} \cup \mathcal{C}_{\mathcal{A}_2}$, (and the clock-space is $\mathcal{H}_\mathcal{A} = \mathcal{H}_{\mathcal{A}_1} \times \mathcal{H}_{\mathcal{A}_2}$),*
3. *$\mathcal{R} : \mathcal{Q}_\mathcal{A} \times \mathcal{Q}_\mathcal{A} \to F(\mathcal{V} - \mathcal{V}_\mathcal{A}, \mathcal{C})$ is defined for every $\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle \in Q_{\mathcal{A}_1} \times Q_{\mathcal{A}_2}$ as*

   $$\mathcal{R}(\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle) = (\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle)[\mathcal{R}_1(q_1, q_1') \wedge \mathcal{R}_2(q_2, q_2')].$$

   In other words, we make a conjunction of the two formulae and substitute the values of the hidden and observable variables induced by the transition from $\langle q_1, q_2\rangle$ to $\langle q_1', q_2'\rangle$.
4. *$\mathcal{O}(\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle) = \mathcal{O}_1(q_1, q_1') \cup \mathcal{O}_2(q_2, q_2')$.*

Claim 1 (Compositionality of $\otimes$). *$L(A_1 \otimes \mathcal{A}_2) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.*

Sketch of Proof: By induction on the number of discontinuities. □

[8] This means that each automaton cannot change the values of the clocks and variables of the other.

4 Translating Equations into Automata

For every k-wire circuit described by a system of equations (1) we let $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, $V = \{v_1, \ldots, v_k\}$ and $\mathcal{C} = \{C_1, \ldots, C_k\}$. With every equation $x_i \in \Delta_{[l_i,u_i]}(f_i(x_1, \ldots, x_k))$ we associate two timed automata A_i and $\mathbf{A}_i$

$$A_i = (\{v_i\}, \{C_i\}, R_i, O_i),$$

$$\mathbf{A}_i = (\{\mathbf{v}_i\}, \emptyset, \mathbf{R}_i, \mathbf{O}_i).$$

The intended meaning of these two automata is as follows. $\mathbf{A}_i$ represents the *observed* value on x_i while A_i represents the "hidden" value on x_i, namely the value x_i is about to obtain given that its input is stable for a sufficiently long period. When a change in f_i occurs, it is A_i that changes its state and resets the clock C_i. This way the clock represents the time elapsed since the occurrence of the change. Changes that last long enough can trigger an observable transition in $\mathbf{A}_i$ (which in turn may change the value of some f_j and v_j). More specifically:

$$\begin{aligned} R_i(0,0) = R_i(1,0) &= f_i(\mathbf{v}_1, \ldots, \mathbf{v}_k) = 0 \\ R_i(0,1) = R_i(1,1) &= f_i(\mathbf{v}_1, \ldots, \mathbf{v}_k) = 1 \\ O_i(0,1) = O_i(1,0) &= \{C_i\} \end{aligned}$$

$$\begin{aligned} \mathbf{R}_i(0,0) &= v_i = 0 \vee C_i < u_i \\ \mathbf{R}_i(0,1) &= v_i = 1 \wedge C_i \geq l_i \\ \mathbf{R}_i(1,0) &= v_i = 0 \wedge C_i \geq l_i \\ \mathbf{R}_i(1,1) &= v_i = 1 \vee C_i < u_i \end{aligned}$$

These two timed automata are depicted in figure 3.

Claim 2 (Properties of $L(A_i)$). *For every i, $L(A_i)$ consists of all the signals x such that for every t*

- $x[v_i]_t = f_i(x[\mathbf{v}_1]_t, \ldots, x[\mathbf{v}_k]_t)$
- $x[C_i]_t = \max\{d : \forall t' \in [t-d, t](x[v_i]_{t'} = x[v_i]_t)\}$, namely, the time elapsed since v_i obtained its current value.

Proof: By definition. □

Claim 3 (Properties of $L(\mathbf{A}_i)$). *For every i, $L(\mathbf{A}_i)$ consists of all the signals x such that for every t:*

- $x[\mathbf{v}_i]_t \neq x[v_i]_t \Rightarrow x[C_i]_t < u_i$
- $t \in \mathcal{L}(x[\mathbf{v}_i]) \Rightarrow l_i \leq x[C_i]_t$

Proof: By definition. □

Claim 4 Properties of $L(\mathbf{A}_i \otimes A_i)$. *The set $L(\mathbf{A}_i \otimes A_i)[\mathbf{V}]$ is exactly the set of signals satisfying the delay inclusion $x_i \in \Delta_{[l_i,u_i]}(f_i(x_1, \ldots, x_k))$.*

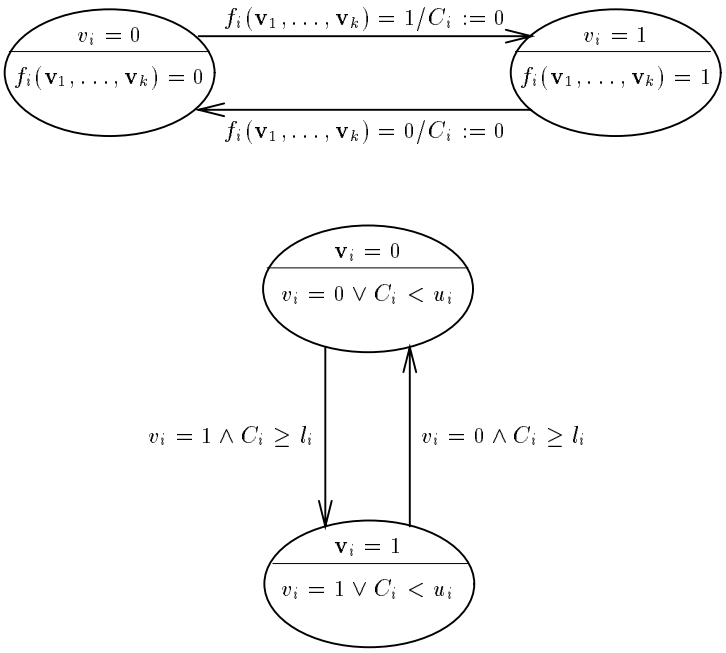

Fig. 3. The two automata A_i and $\mathbf{A}_i$ associated with each equation

.

Sketch of Proof: By combining what claim 2 tells us about C_i and v_i, what claim 3 tells us about their influence on $\mathbf{v}_i$, with the compositionality of $\otimes$ (claim 1) we obtain a characterization equivalent to definition 4. □

Theorem 5 (Main Result). *Every k-wire circuit can be transformed into an equivalent timed automaton with $2k$ variables and k clocks.*

Proof: Given a system of delay inclusions, we create the corresponding system of timed automata and compose them, obtaining

$$\mathcal{A} = \bigotimes_{i=1}^{k} (A_i \otimes \mathbf{A}_i) = \mathcal{A} = (V \cup \mathbf{V}, \mathcal{C}, \mathcal{R}, \mathcal{O})$$

as described above, whose set of observable behaviors $L(\mathcal{A})[\mathbf{V}]$ is exactly the set of solutions. □

Example: Consider the simple circuit in figure 4. We model x_1 as an oscillator with parameters l_1 and u_1. Initially we obtain the four 2-state automata $\mathbf{A}_1$, A_1, $\mathbf{A}_2$ and A_2 whose $\mathcal{R}$ and $\mathcal{O}$ are written in table 1. After composing $\mathbf{A}_1$ with A_1 and $\mathbf{A}_2$ with A_2 (and removing states and transitions whose $\mathcal{R}$ is **false** – these indicate unstable states in which the automaton cannot stay for a non-zero

duration) we obtain the two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ of table 2. Composing those two we end up with the four-state timed automaton of table 3 which is depicted (with the hidden variables removed) in figure 5. *This automaton generates all the observable signals of the circuit.*

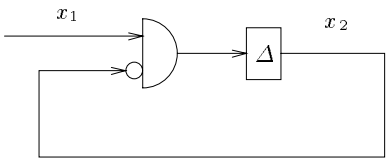

Fig. 4. A circuit.

$\mathbf{v}_1$	0	1
0	$v_1 = 0 \vee C_1 < u_1$	$v_1 = 1 \wedge C_1 \geq l_1$
1	$v_1 = 1 \wedge C_1 \geq l_1$	$v_1 = 1 \vee C_1 < u_1$

$\mathbf{A}_1$

v_1	0	1
0	$\mathbf{v}_1 = 0$	$\mathbf{v}_1 = 1$ $\{C_1\}$
1	$\mathbf{v}_1 = 1$ $\{C_1\}$	$\mathbf{v}_1 = 1$

A_1

$\mathbf{v}_2$	0	1
0	$v_2 = 0 \vee C_2 < u_2$	$v_2 = 1 \wedge C_2 \geq l_2$
1	$v_2 = 0 \wedge C_2 \geq l_2$	$v_2 = 1 \vee C_2 < u_2$

$\mathbf{A}_2$

v_2	0	1
0	$\mathbf{v}_1 = 0 \vee \mathbf{v}_2 = 1$	$\mathbf{v}_1 = 1 \wedge \mathbf{v}_2 = 0$ $\{C_2\}$
1	$\mathbf{v}_1 = 0 \vee \mathbf{v}_2 = 1$ $\{C_2\}$	$\mathbf{v}_1 = 1 \wedge \mathbf{v}_2 = 0$

A_2

Table 1. The four initial automata.

5 Applications

Once a circuit has been translated into an automaton, we can reason about its behavior using (timed) automata-theoretic methods. The main applicable results are those concerning analysis/model-checking ([AD94], [ACD93], [HNSY94]) and

$\mathbf{v}_1, v_1$	$\mathbf{10}$	$\mathbf{01}$
$\mathbf{10}$	$C_1 < u_1$	$C_1 \geq l_1$ $\{C_1\}$
$\mathbf{01}$	$C_1 \geq l_1$ $\{C_1\}$	$C_1 < u_1$

$\mathcal{A}_1$

$\mathbf{v}_2, v_2$	$\mathbf{00}$	$\mathbf{01}$	$\mathbf{10}$
$\mathbf{00}$	$\mathbf{v}_1 = 0$	$\mathbf{v}_1 = 1$ $\{C_2\}$	
$\mathbf{01}$	$\mathbf{v}_1 = 0 \wedge C_2 < u_2$ $\{C_2\}$	$\mathbf{v}_1 = 1 \wedge C_2 < u_2$	$C_2 \geq l_2$ $\{C_2\}$
$\mathbf{10}$	$\mathbf{v}_1 = 0 \wedge C_2 \geq l_2$ $\{C_2\}$	$\mathbf{v}_1 = 1 \wedge C_2 \geq l_2$ $\{C_2\}$	$C_2 < u_2$

$\mathcal{A}_2$

Table 2. $\mathcal{A}_1 = \mathbf{A}_1 \otimes A_1$ and $\mathcal{A}_2 = \mathbf{A}_2 \otimes A_2$. An empty entry in the table indicates **false**.

$\mathbf{v}_1, v_1$ $\mathbf{v}_2, v_2$	$\mathbf{1001}$	$\mathbf{1010}$	$\mathbf{0100}$	$\mathbf{0110}$
$\mathbf{1001}$	$C_1 < u_1 \wedge C_2 < u_2$	$C_1 < u_1 \wedge C_2 \geq l_2$ $\{C_2\}$	$C_1 \geq l_1 \wedge C_2 < u_2$ $\{C_1, C_2\}$	$C_1 \geq l_1 \wedge C_2 \geq l_2$ $\{C_1, C_2\}$
$\mathbf{1010}$	$C_1 < u_1 \wedge C_2 \geq l_2$ $\{C_2\}$	$C_1 < u_1 \wedge C_2 < u_2$	$C_1 \geq l_1 \wedge C_2 \geq l_2$ $\{C_1, C_2\}$	$C_1 \geq l_1 \wedge C_2 < u_2$ $\{C_2\}$
$\mathbf{0100}$	$C_1 \geq l_1$ $\{C_1, C_2\}$		$C_1 < u_1$	
$\mathbf{0110}$	$C_1 \geq l_1 \wedge C_2 \geq l_2$ $\{C_1, C_2\}$	$C_1 \geq l_1 \wedge C_2 < u$ $\{C_1\}$	$C_1 < u_1 \wedge C_2 \geq l_2$ $\{C_2\}$	$C_1 < u_1 \wedge C_2 < u_2$

$\mathcal{A}$

Table 3. $\mathcal{A} = \mathcal{A}_1 \otimes \mathcal{A}_2$.

synthesis[9] ([HWT92], [MPS95] [AMP95-b]). We will briefly present these results and discuss their usefulness.

5.1 Analysis

Given a timed automaton $\mathcal{A}$, one can decide whether a state q' is reachable from a state q. This is done by an algorithm that calculates, using simple linear-algebra techniques, the set of successors of a given configuration. More generally, the satisfaction of properties expressed in various real-time temporal logics can be verified as well. Such properties go beyond simple reachability and allow one

[9] The word "synthesis" is used in the hardware community for a kind of "compilation" between an abstract representation into a more concrete one. In the software verification community the meaning is like in control theory, namely, constructing a system from its specifications. This is the sense in which we use the word.

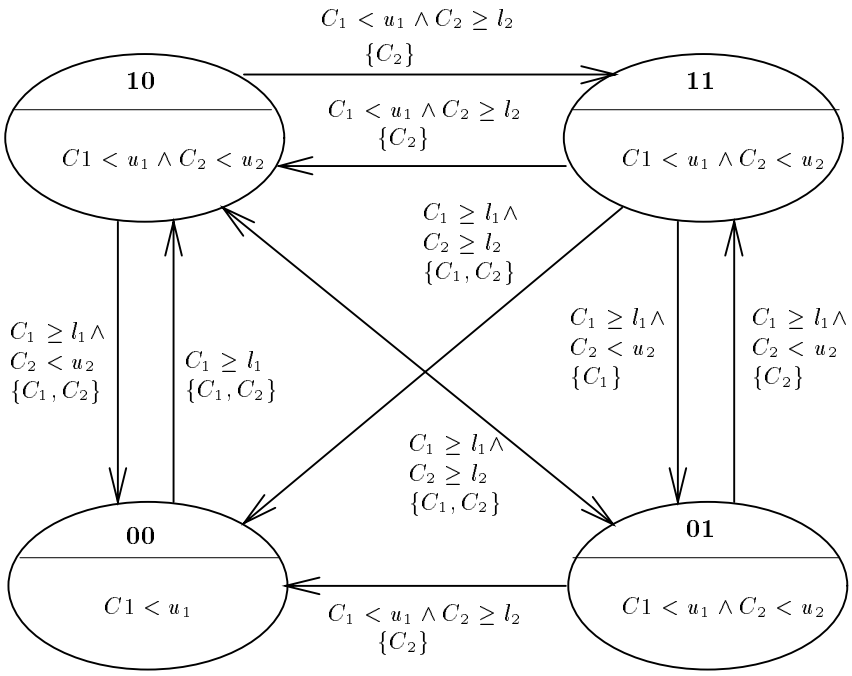

Fig. 5. The automaton $\mathcal{A}$ projected on the observed variables.

to express, for example, a fact like *every visit of the system in state q is followed within d time units by a visit in state q'.*

All these features are already implemented in the tool KRONOS ([HNSY94], [DOY94]) developed at VERIMAG. As an initial experiment we have used it to verify that a MOS circuit with 8 elements[10] and 4 input signals never reaches a certain "short-cut" state. This property has been verified against a non-deterministic specification of the relative rising and falling times of the input signals.

5.2 Synthesis

The synthesis problem for timed automata can be roughly phrased as follows: *Given an automaton $\mathcal{A} = (\mathcal{V}_\mathcal{A}, \mathcal{C}_\mathcal{A}, \mathcal{R}, \mathcal{O})$, find a systematic way to restrict its behavior such that some property is satisfied.* By "restricting the behavior" we mean to modify $\mathcal{A}$ into $\mathcal{A}' = (\mathcal{V}_\mathcal{A}, \mathcal{C}_\mathcal{A}, \mathcal{R}', \mathcal{O})$ such that for every $q, q' \in \mathcal{Q}_\mathcal{A}$, $\mathcal{R}'(q, q') \Rightarrow \mathcal{R}(q, q')$. A typical example of a restriction is to replace an inequality

[10] In this paper we have presented the model using Boolean gates, but any other basis consisting of functions over finite domains can be treated as well.

of the form $l \leq C \leq u$ in $\mathcal{R}$ by $l' \leq C \leq u'$ such that $l \leq l' \leq u' \leq u$. Clearly, by restricting $\mathcal{R}$ the set of signals generated by the automaton decreases and $L(\mathcal{A}') \subseteq L(\mathcal{A})$.

The algorithm presented in [MPS95], [AMP95-b], which is based on the same geometric ideas as the analysis algorithms for timed automata, can extract a restricted automaton all of whose behaviors satisfy a given property. If no such automaton exists, the algorithms can point out a configuration (state + clocks) from which the transition to the bad state cannot be avoided. We will demonstrate how this result can be used for solving two concrete problems in circuit analysis.

Consider a circuit with given delay characteristics. We want to know what constraints must be imposed on the input signals in order that some state q will never be reached. We built the equations for the internal signals and let the input signals be unconstrained (or specified by $x_i \in \Delta_{[d,\infty]}(\neg x_i)$ for some minimal propagation constant d). Then, after translating the system into an automaton $\mathcal{A}$, our algorithm will search from q backwards, trying to eliminate bad transitions by putting further restrictions on the formulae. The restrictions can be made only for those parts of a formula which originate from the equations that correspond to the input. In the ideal case, we will get as a solution an automaton $\mathcal{A}'$ admitting a reverse translation into a similar system of delay equations where possibly smaller delay interval are associated with the input signals. This would mean that it is sufficient to restrict the variability of every input signal individually. In more complicated cases we will have restrictions that relate several input signals. For example the condition in $\mathcal{A}'$ corresponding to a transition that changes some value of x_i may refer to a clock C_j for some $i \neq j$. In this case the set of input signals should satisfy more complicated constraints, such as: *input signal x_i will never change unless some time has elapsed since the last change in input x_j.*

In more complicated cases, a formula that corresponds to an input x_i in the solution $\mathcal{A}'$ may refer to clocks of internal signals. This will indicate that the input cannot be constrained in a feed-forward manner, but that we need a feedback from the internal components in order to select admissible inputs. This will generally indicate bad design. It should be noted, however, that if the gate delays are deterministic, the problem of calculating the maximal set of input signals against which the circuits operates properly can be solved without reference to the clocks associated with the gates, as the values of those can be inferred from other variables.

In a similar manner we can solve the opposite problem: given a circuit and a class of input signals, find the delay characteristics of the gates that will satisfy some reachability property. Here we will start with the most general delay parameters of the non-input signals and use our algorithm to restrict them and obtain a good automaton. As in the previous problem, ideal solutions could be translated back into delay equations (gate parameters are independent) while in more complicated cases the relation between gate delays will be of a more intricate and dynamic nature.

The classification of these problems and solutions as well as the implementation of efficient data-structures and algorithms for timed reachability analysis is subject of an ongoing research. We believe that, as in the case of synchronous circuits and ordinary untimed automata, the translation into automata clarifies the issues, allows a uniform treatment of a class of problems that might look different at a first glance, and helps to focus on practical solutions of the algorithmic issues.

Acknowledgment

This work grew out of discussions with J. Sifakis. We have also benefitted from comments made by other members of VERIMAG, in particular, A. Bouajjani, C. Daws, Y. Lakhneche, R. Robbana and S. Yovine.

References

[ACD93] R. Alur, C. Courcoubetis, and D.L. Dill, Model Checking in Dense Real Time, *Information and Computation* 104, 2–34, 1993.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, The Algorithmic Analysis of Hybrid Systems, *Theoretical Computer Science* 138, 3–34, 1995.

[AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.

[AMP95-a] A. Asarin, O. Maler and A. Pnueli, Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives, *Theoretical Computer Science* 138, 35-66, 1995.

[AMP95-b] A. Asarin, O. Maler and A. Pnueli, Symbolic Synthesis of Discrete and Timed Systems, in A. Nerode (Ed.), *Hybrid Systems II*, Springer LNCS, to appear, 1995.

[BD91] B. Berthomieu and M. Diaz, Modeling and Verification of Time Dependent Systems using Time Petri Nets, *IEEE Trans. on Software Engineering* 17, 259-273, 1991.

[BM83] 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.

[BS91] 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.

[DOY94] C. Daws, A. Olivero and S. Yovine, Verifying ET-LOTOS Programs with KRONOS, *Proc. FORTE'94*, Bern, 1994.

[D89] 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.

[HNSY94] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, Symbolic Model-checking for Real-time Systems, *Information and Computation* 111, 193–244, 1994.

[HWT92] G. Hoffmann and H. Wong-Toi, The Input-Output Control of Real-Time Discrete Event Systems, *Proc. Real-Time Systems Symposium*, IEEE, 1992.

[KM91] R.P. Kurshan and K.L. McMillan, Analysis of Digital Circuits Through Symbolic Reduction, *IEEE Trans. on Computer-Aided Design*, 10, 1350-1371, 1991.

[LB94] W.K.C. Lam and R.K. Brayton, *Timed Boolean Functions*, Kluwer, 1994.

[L89] H.R. Lewis, Finite-state Analysis of Asynchronous Circuits with Bounded Temporal Uncertainty, TR15-89, Harvard University, 1989.

[LL94] M. Linderman and M. Lesser, Simulation of Digital Circuits in the Presence of Uncertainty, TR EE-CEG-94-2, School of EE, Cornell University, 1994.

[ML93] A. Martello and S.P. Levitan, Temporal Analysis of Time Bounded Digital Systems, in G.J. Milne and L. Pierre (Eds), *Proceedings of Charme'93,* LNCS 683, 27-38, Springer, 1993.

[MPS95] A. Maler, O. Pnueli and J. Sifakis. On the Synthesis of Discrete Controllers for Timed Systems, in E.W. Mayr and C. Puech (Eds), *Proceedings of STACS'95,* LNCS 900, 229-242, Springer, 1995.

This article was processed using the LaTeX macro package with LLNCS style