Sādhanā, Vol. 21, Part 2, April 1996, pp. 125–145. © Printed in India.

A graphic language based on timing diagrams

CHRISTIAN ANTOINE[1], BERNARD LE GOFF and
JEAN-ERIC PIN[2]

BULL Research and Advanced Development, Rue Jean-Jaurès, 78340 Les Clayes-sous-Bois, France
Present address: [1]LETI (CEA-Technologie Avancée), DEIN-CE/SLA Bât. 528, 91191 Gif-sur-Yvette Cedex, France
[2]LITP/IBP, Université Paris VI, Tour 55-65, 4 Place Jussieu, 75252 Paris Cedex 05, France
E-mail: cantoine@aigle.saclay.cea.fr, pin@litp.ibp.fr

Abstract. We present a new graphic language which can serve, for instance, as models for VLSI and control systems. Its primitives are based on standard timing diagrams, and this is a great advantage over other formalisms since designers can rapidly master it. The semantics is rigorously defined in the formalism of the theory of automata on infinite words. Using this formalism, we are able to give a rather precise upper-bound on the expressive power of our graphic language in terms of a language theoretic measure, the *concatenation level*. A detailed example is presented.

Keywords. Graphic language; timing diagrams; concatenation level.

1. Introduction

This paper emerged as the result of a discussion between circuit designers and researchers working in the area of specification languages on the one hand and automata theory on the other. It has a practical component, the description of new formal specification language resembling timing diagrams, as well as a strong theoretical flavour, since the semantics of the language is based on results from the theory of automata on infinite words.

Our work is motivated by the following observation : circuit designers are often discouraged by the complexity of the specification languages. In an effort to remedy this problem, we introduce a graphic language, called the *Chronogram Language* (Antoine & Le Goff 1991), the primitives of which are based on standard timing diagrams. Timing diagrams are a formalism which is commonly used in the community of circuit designers, so our language can be rapidly mastered. In other words, contrary to most formalisms, properties are *drawn* rather than written, and this pictural representation is much more convenient for the non-specialist than an abstract formalism.

On the other hand, the use of pictures does not preclude a precise syntax and semantics. It turns out that the primitives of our language can be conveniently interpreted as rational (also called *regular*) expressions on infinite words. It will follow from our syntax that *all* chronograms can be interpreted as rational expressions. This approach not only permits us to define rigorously the semantics of the chronogram language, but also gives precise information about its expressive power. Before stating our results, we need to briefly review some facts on rational sets of infinite words.

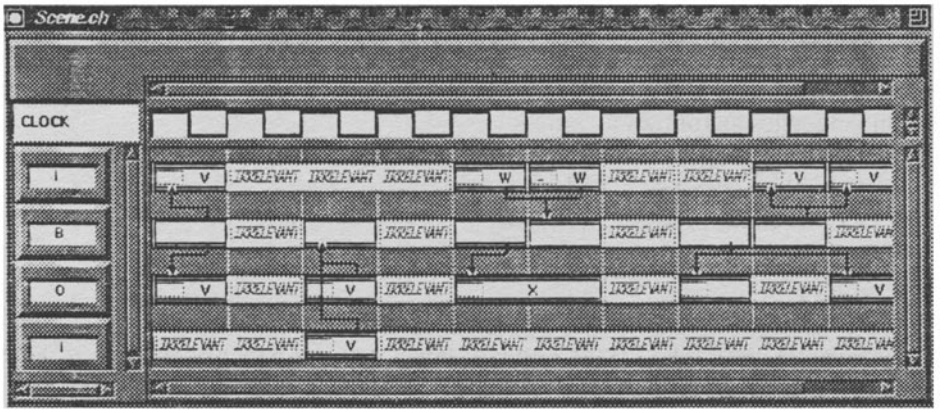
There are two well-known scales to measure the complexity of a rational set, the logical scale and the combinatorial scale. The logical scale branches into two main parts, corresponding to the first order logic and to the monadic second order logic, respectively. Within the first order logic one can define a hierarchy by counting the number of alternations between existential and universal quantifiers. The combinatorial scale, on the other hand, is based on the basic operations used to define the rational sets : boolean operations, concatenation product and iteration. It also branches into two main domains : the star-free sets (which can be defined without using iteration) and the rational sets. A hierarchy inside the star-free sets is obtained by counting the number of alternations between the use of the boolean operations and of the concatenation product. A nice (but non-trivial) feature is that the logical and the combinatorial scales coincide (Thomas 1982; Perrin & Pin 1986). Our main result states that the languages definable using chronograms are within level 3 in the star-free (or logical) hierarchy. This gives a rather precise upper bound to the expressive power of the Chronogram Language.

Although our language was originally designed as a language for specifying circuit behaviour, it can serve more generally for modelling temporal properties. The Chronogram Language has been designed to provide designers with a good expressive power for temporal properties. For instance, both safety and liveness properties can be expressed in the Chronogram Language, in contrast with other languages VHDL (Lipsett *et al* 1990), Lucid (Ashcroft & Wadge 1976), Lustre (Caspi & Halbwachs 1986), Signal (Le Goff *et al* 1989), etc. which cannot express liveness properties. To ensure compatibility with existing formalisms, the chronograms that represent safety properties can be compiled into VHDL (a standard description language used in circuit design) and Signal expressions, and liveness properties will be translated into CTL* in the future.

The paper is organized as follows. The Chronogram language is introduced through an example which is analysed later in § 6. The abstract syntax of the Chronogram Language is given in § 4. In order to keep the paper self-contained, the main definitions on languages and automata required for this paper are summarized in § 3. The semantics of the Chronogram Language are presented in § 5 and are illustrated by means of a detailed example in § 6. The paper concludes with our plan for future work. Our approach is illustrated by several examples of interpretations of chronograms involving rational sets of infinite words.

2. A presentation of the Chronogram Language

At the top of the chronogram in figure 1 is shown the CLOCK, which informally, represents the time. All the events are synchronized on the rising edges of the clock, except if all zones below the clock are *IRRELEVANT* zones. In the latter case, the duration of the signal is not



A chronogram.

Figure 1. A chronogram.

specified. This chronogram defines constraints on the events of three boolean signals: I, O and B. Each line is dedicated to a signal: the second one for B, the third one for O and the first and the last ones for I. Each line consists of *IRRELEVANT* zones and bold line boxes. Only the bold line boxes are relevant for the definition of constraints. On the second line (dedicated to the B signal), there are three boxes with a solid line at the bottom, and three boxes with a solid line at the top. This means that B must carry the true value during the period of time represented by the first three boxes, and the false value during the period of time represented by the last three boxes. On the first and third line, the boxes are labelled by a symbol (v, x or w). This means that during the period of time represented by the box, the signal carries the value v (resp. x or w). This value v (resp. x, w) is not specified in the chronogram but has to be the same in all boxes labelled by v (resp. x, w). A minus sign can be added in the left part of the box: in this case, the signal carries the value $\bar{v}$ opposite to the label v of the box. On the first line such a box is used with the symbol w.

The bold line boxes can be connected by arrows. The resulting graph can have several (simply) connected components. Each component defines a constraint. The relative location of the boxes is relevant only inside a connected component. For instance, the properties 1, 2, 4, 5, 6 and 3 which are detailed in § 6 are specified in this order by the chronogram. Let us consider the first property: *when the gate is opened, I and O carry the same value*. The gate is opened if and only if B carries the false value. And in this case, I and O carry the value denoted by the symbol v in the chronogram: the arrows mean that "B carries the false value" implies that I and O carry the value v. The other properties can be read in a similar way in the chronogram: two linked arrows must be interpreted as a logical and.

3. Languages and automata

In this section, we briefly recall some basic definitions from the theory of automata needed in this article. For more details, the reader is referred to Eilenberg (1974, 1976), Perrin (1990) and Thomas (1990). We also define the language-theoretic hierarchy that will serve as a measure of the expressive power of the Chronogram Language.

We denote respectively by A^*, A^+ and A^ω the sets of finite words, non-empty finite words and infinite words on an alphabet A. A *language* is a set of finite words, that is, a subset of A^*. The *rational operations* are the three operations union, product and star, defined on languages as follows

(1) Union : $L_1 \cup L_2 = \{u \mid u \in L_1 \text{ or } u \in L_2\}$

(2) Product : $L_1 L_2 = \{u_1 u_2 \mid u_1 \in L_1 \text{ and } u_2 \in L_2\}$

(3) Star : $L^* = \{u_1 \cdots u_n \mid n \geq 0 \text{ and } u_1, \ldots, u_n \in L\}$

The set of rational (or regular) languages of A^* is the smallest set of subsets of A^* containing the finite sets and closed under finite union, product and star. For instance, $\{a, ab\}^*ab \cup (ba^*b)^*$ denotes a rational set. The rational subsets of A^+ are the rational subsets of A^* that do not contain the empty word. It is possible to generalize the concept of rational languages to infinite words as follows. First, the product can be extended to $A^* \times A^\omega$, by setting, for $X \subset A^*$ and $Y \subset A^\omega$,

$$XY = \{xy \mid x \in X \text{ et } y \in Y\}.$$

Next, we define an infinite iteration ω by setting, for every subset X of A^+

$$X^\omega = \{x_0 x_1 x_2 \cdots \mid \text{ for all } i \geq 0, x_i \in X\}.$$

That is, X^ω is the set of infinite words obtained by concatenating an infinite sequence of words of X. By definition, a subset of A^ω is *ω-rational* (or *ω-regular*) if it is equal to a finite union of sets of the form XY^ω where X and Y are non-empty rational sets of A^+.

Boolean operations comprise union, intersection, complementation and set difference. It can be shown that the rational subsets of A^* are closed under finite boolean operations. The set of *star-free* subsets of A^* is the smallest set of subsets of A^* containing the finite sets and closed under finite boolean operations and product.

For instance, A^* is star-free, since it is the complement of the empty set. More generally, if B is a subset of the alphabet A, the set B^* is also star-free since B^* is the complement of the set of words that contain at least one letter of $B' = A \setminus B$. This leads to the following star-free expression (where X^c denotes the complement of a set X).

$$B^* = A^* \setminus A^*(A \setminus B)A^* = (\emptyset^c(A \setminus B)\emptyset^c)^c = (\emptyset^c(A^c \cup B)^c\emptyset^c)^c$$

Of course, $B^+ = B^* \setminus \{\varepsilon\}$ is also star-free.

The set of star-free subsets of A^ω is the smallest set $\mathcal{S}$ of subsets of A^ω closed under finite boolean operations and such that if X is a star-free subset of A^+ and $Y \in \mathcal{S}$, then $XY \in \mathcal{S}$.

The definition of star-free languages of A^* makes use of two different types of operations: boolean operations and concatenation product. By alternating the use of these two operations, one gets a hierarchy, called the *concatenation hierarchy*, defined as follows.

(1) The sets of level 0 are the empty set and A^*,

(2) For every integer $n \geq 0$, the sets of level $n + 1/2$ are the finite unions of the sets of the form

$$L_0 a_1 L_1 a_2 \cdots a_k L_k$$

where $L_0, L_1, \ldots, L_k$ are sets of level n and $a_1, \ldots, a_k$ are letters

(3) For every integer $n \geq 0$, the sets of level $n + 1$ are finite boolean combinations of sets of level $n + 1/2$.

Note that a set of level m is also a set of level n for every $n \geq m$. The languages of level $1/2$ are the finite unions of languages of the form $A^*a_1A^*a_2 \cdots a_kA^*$, the languages of level 1 are finite boolean combinations of these languages, etc. The following languages are of level 1 on the alphabet A:

$$B^* = A^* \setminus \bigcup_{a \in A \setminus B} A^*aA^*$$

$$\{\varepsilon\} = A^* \setminus \bigcup_{a \in A} A^*aA^*$$

$$B^+ = B^* \setminus \{\varepsilon\}$$

The next proposition summarizes several results relative to this hierarchy.

PROPOSITION 1
(Brzozowski & Knast 1978; Perrin & Pin 1986),

(1) The finite languages have level 1.

(2) For each $n \geq 0$, the languages of level n are closed under union, intersection, and complement.

(3) For each $n \geq 0$, the languages of level $n + 1/2$ are closed under union, intersection, and product.

(4) Let $n \geq 0$ and let $\varphi : A^* \to B^*$ be a monoid morphism. If L is a language of level n (respectively $n + 1/2$), then $\varphi^{-1}(L)$ is also of level n (respectively $n + 1/2$).

(5) The hierarchy is strict for all n: there exist languages of level $n + 1$ that are not of level $n + 1/2$ and languages of level $n + 1/2$ that are not of level n.

Concatenation hierarchies can be extended to infinite words as follows (Perrin & Pin 1986).

(1) The sets of level 0 are the empty set $\emptyset$ and A^ω,

(2) For every integer $n \geq 0$, the sets of level $n + 1/2$ are the finite unions of the sets of the form XaY, where X is a set of A^* of level $n + 1/2$, Y is a subset of A^ω of level n and a is a letter.

(3) For every $n \geq 0$, the sets of level $n + 1$ are finite boolean combinations of sets of level $n + 1/2$.

4. An abstract syntax of the Chronogram Language

The abstract syntax of the language is given by a grammar, in which the initial of each non-terminal is a capital letter (e.g. `Clock`) and each terminal is either written in capital

letters (e.g. IDENTIFIER), or consists of a single lower-case letter (e.g. i) or of a non alphabetic sign (e.g. 1, *).

The rules are grouped by level of derivation and every rule is written only once. As a consequence, the derivation rules of certain terms may precede some of their occurrences.

```
Constraint ::= Property Constraint | Property

Property ::= Clock Hypothesis Conclusion

Hypothesis ::= TimeDiagram
Conclusion ::= TimeDiagram

TimeDiagram ::= MultiColumnList

Clock ::= IDENTIFIER
MultiColumnList ::= MultiColumn MultiColumnList | MultiColumn

MultiColumn ::= StaticMultiColumn | DynamicMultiColumn

StaticMultiColumn ::= Width StaticRowList
DynamicMultiColumn ::= FiniteLowerBound UpperBound DynamicRowList

StaticRowList ::= StaticRow StaticRowList | StaticRow
DynamicRowList ::= DynamicRow DynamicRowList | DynamicRow

StaticRow ::= StaticIntervalList IDENTIFIER
DynamicRow ::= DynamicIntervalList IDENTIFIER

UpperBound ::= FiniteUpperBound | *
Width ::= INTEGER
Length ::= INTEGER
FiniteLowerBound ::= INTEGER
FiniteUpperBound ::= INTEGER

StaticIntervalList ::= StaticInterval StaticIntervalList | NIL
DynamicIntervalList ::= DynamicInterval DynamicIntervalList | NIL

StaticInterval ::= Length PrimitiveSymbol
DynamicInterval ::= PrimitiveSymbol

PrimitiveSymbol ::= i | f | e | r | s | 1 | 0 | SymbolicValue

SymbolicValue ::= - IDENTIFIER | IDENTIFIER
```

The intuitive meaning of the primitive symbols is the following:

i (Irrelevant) The value of the signal is not specified and can be either 0 or 1.

1 The signal is stable and its value is 1.

0 The signal is stable and its value is 0.

f (Falling) The signal owns one and only one falling edge (but may have 0, 1 or 2 rising edges).

r (Rising) The signal owns one and only one rising edge (but may have 0, 1 or 2 falling edges).

s (Stable) The signal is stable but its value is unknown.

e (Edge) The value of the signal changes once and only once.

5. Semantics of the Chronogram Language

The formal semantics of the Chronogram Language are given in terms of ω-rational languages. More precisely, a certain rational language is associated with each of the graphic primitives of the Chronogram Language and with each variable. Next, to each operator of the Chronogram Language (generation of intervals, rows, columns, multicolumns, time diagrams, etc.) corresponds an operation on languages that preserves rationality. A distinguishing feature of the Chronogram Language is the use of *symbolic values* or *boolean variables*. We shall first detail this peculiar aspect.

5.1 *Boolean variables and valuations*

If v denotes a boolean variable, $\bar{v}$ will denote its complement. Thanks to boolean variables, one can specify in the Chronogram Language not only properties like "The value of the signal at time t is 0 (resp. 1)", but also properties of the form "the value of the signal is v at time t and $\bar{v}$ at time $t + 3$". In order to take in account these variables, it is convenient, in the first place, to represent a signal not as an infinite word on the alphabet $B = \{0, 1\}$, but as an infinite word on the extended alphabet $C = B \cup V \cup \bar{V}$, where V is the set of variables used in the chronogram.

One goes back to the binary alphabet B by associating a value with each variable. This can formally be realized by a *valuation*, that is a map $\nu : C \to B$ such that

a) for all $b \in B$, $\nu(b) = b$

b) for all $v \in V$, $\nu(\bar{v}) = \overline{\nu(v)}$.

For instance, the previous example would be interpreted as "the value of the signal is 0 at time t and 1 at time $t + 3$" (which corresponds to the valuation ν defined by $\nu(v) = 0$) or "the value of the signal is 1 at time t and 0 at time $t + 3$" (which corresponds to the valuation ν defined by $\nu(v) = 1$).

A valuation $\nu : C \to B$ defines in a natural way a function $\nu : C^* \to B^*$, by setting, for every word $c_1 c_2 \cdots c_n \in C^*$,

$$\nu(c_1 c_2 \cdots c_n) = \nu(c_1)\nu(c_2) \cdots \nu(c_n)$$

If L is a subset of C^*, the set $\nu(L)$ is called the valuation of L.

5.2 *Constraints on a single signal*

A signal is considered as an infinite word u on the binary alphabet B. As we shall see later, the constraints defined on a given signal in our language can always be formulated under the form $u \in LB^{\omega}$, where L is a certain rational language of B^*, that we shall now compute in more detail.

If the chronogram contains variables, we first identify the signal with an infinite word u on the alphabet C, as was explained before. The constraint in which the variables are not interpreted can be formulated under the form $u \in LB^{\omega}$, where L is a certain rational language of C^*, while the final constraint can be expressed under the form

$$\in \bigcup_{\nu \text{ valuation}} \nu(L)B^{\omega}.$$

There are in fact two types of constraints, the "static" constraints, which correspond to the case where L is a finite language, and the "dynamic" constraints, that correspond to the case where L can be an infinite language.

In the case of a static constraint, the language L is obtained as a finite concatenation of rational languages corresponding to static intervals. For instance, the following sequence of static intervals defines a constraint: "between time n_1 and n_2, the signal has a unique rising edge, between time n_2 and n_3, its value is a constant v and between time n_3 and n_4, its value is always 0". Note that in this case, the values of $n_2 - n_1$, $n_3 - n_2$ and $n_4 - n_3$ are the length of the static intervals.

In the case of a dynamic constraint, the language L is obtained as a finite concatenation of rational languages corresponding to dynamic intervals. For instance, the following sequence of dynamic intervals defines a constraint: "There exist instants n_2, n_3, n_4 such that between time n_1 and n_2, the signal has a unique rising edge, between time n_2 and n_3, its value is a constant v and between time n_3 and n_4, its value is always 0". The difference with the previous case is that the values of $n_2 - n_1$, $n_3 - n_2$ and $n_4 - n_3$ are not specified in the dynamic constraints, that is, can be chosen arbitrarily.

The languages associated with (static or dynamic) intervals are themselves obtained from the so-called *primitive* languages associated with the *primitive* symbols. This vocable concerns the elements of the set

$$V \cup \bar{V} \cup \{\mathbf{i}, \mathbf{0}, \mathbf{1}, \mathbf{f}, \mathbf{r}, \mathbf{s}, \mathbf{e}\},$$

that is, all symbols of variables (possibly overlined) and the symbols associated with the graphic primitives of the Chronogram Language. Recall the intuitive meaning of these primitives.

- **i** (Irrelevant) The value of the signal is not specified and can be either 0 or 1.
- **1** The signal is stable and its value is 1.
- **0** The signal is stable and its value is 0.
- **f** (Falling) The signal owns one and only one falling edge (but may have 0, 1 or 2 rising edges).
- **r** (Rising) The signal owns one and only one rising edge (but may have 0, 1 or 2 falling edges).

s (Stable) The signal is stable but its value is unknown.

e (Edge) The value of the signal changes once and only once.

This leads to the following table of the primitive languages associated with the graphic primitives:

$$L(\mathbf{i}) = \{0, 1\}^+ \qquad L(\mathbf{1}) = 1^+ \qquad L(\mathbf{0}) = 0^+$$
$$L(\mathbf{f}) = 0^*1^+0^+1^* \qquad L(\mathbf{r}) = 1^*0^+1^+0^* \qquad L(\mathbf{s}) = 0^+ \cup 1^+$$
$$L(\mathbf{e}) = 0^+1^+ \cup 1^+0^+$$

On the other hand, the primitive language associated with each variable v is

$$L(v) = v^+ \quad \text{and} \quad L(\bar{v}) = \bar{v}^+.$$

We, therefore, have

PROPOSITION 2
The primitive languages and their valuations are star-free languages of level 3/2.

Proof. We have already seen that the languages $L(\mathbf{i}) = \{0, 1\}^+$, $L(\mathbf{1}) = 1^+$, $L(\mathbf{0}) = 0^+$, $L(v) = v^+$ and $L(\bar{v}) = \bar{v}^+$ are languages of level 1. It follows that $L(s) = 0^+ \cup 1^+$ is also of level 1. On the other hand, $L(\mathbf{f}) = 0^*1^+0^+1^*$ is a product of languages of level 1/2 and thus is of level 3/2. A similar argument would show that the languages $L(\mathbf{r})$ and $L(\mathbf{e})$ are of level 3/2. Finally, each valuation of the languages $L(v)$ and $L(\bar{v})$ is equal to either 0^+ or 1^+, which are languages of level 1. □

We can now formally define the notion of interval. A *static interval* is a couple $I = (\ell, t)$ where ℓ is a positive integer and t is a primitive symbol. Intuitively, the integer ℓ represents the length of the interval on which the condition defined by t will be considered. For example, if $\ell = 5$ and $t = e$, the value of the signal will change once and only once in the interval [0, 5]. The language associated with I is the subset of C^* defined by

$$L(I) = L(\ell, t) = L(t) \cap C^\ell.$$

For example, if $\ell = 5$ and $t = e$, then

$$L(I) = (0^+1^+ \cup 1^+0^+) \cap C^5$$
$$= \{01111, 00111, 00011, 00001, 10000, 11000, 11100, 11110\}$$

A *dynamic interval* is simply a primitive symbol and thus the corresponding language is already defined.

A *static* (resp. *dynamic*) row is a sequence of static (resp. dynamic) intervals (figure 2). The language associated with a row $(I_1, I_2, \ldots, I_s)$ is defined by

$$L(I_1, I_2, \ldots, I_s) = L(I_1)L(I_2)\cdots L(I_s).$$

For instance, the language of C^+ associated with the row represented below is

$$11v\{0, 1\}\bar{v}\bar{v}\{01111, 00111, 00011, 00001\}.$$

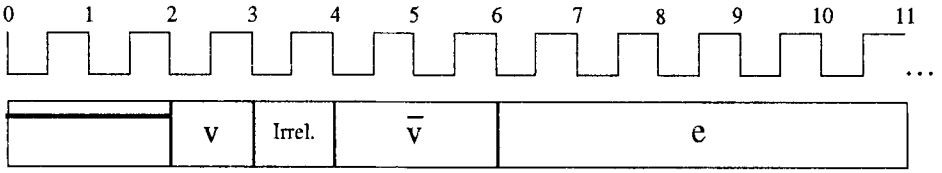


Figure 2. A static row.

Here is another example, for a dynamic row. If $I_1 = v$, $I_2 = e$ and $I_3 = \bar{v}$, then

$$L(I) = v^+(0^+1^+ \cup 1^+0^+)\bar{v}^+.$$

The languages associated with rows are described in the next proposition

PROPOSITION 3
The languages associated with a static row and their valuations are finite languages. The languages associated with a dynamic row and their valuations are languages of level 3/2.

Proof. The language associated with a static row is an intersection of languages associated with static intervals, which are finite languages. Since the valuation of a finite language is finite, the first part of the statement follows.

The language associated with a dynamic row is a product of languages of dynamic intervals. Now, by proposition 2, the languages associated with dynamic intervals and their valuations are of level 3/2 and by proposition 1, the product of languages of level 3/2 is also of level 3/2. □

Finally, if L is the language associated with a (static or dynamic) row, the constraint defined by this row is the set

$$\bigcup_{\nu \text{ is a valuation}} \nu(L)B^\omega.$$

In other words, in order to compute the constraint defined by a row, one first computes the language L associated with this row on the extended alphabet C and then one simply gives a value to the variables. For instance, for the row represented in figure 2, the constraint can be written

$$(111\{0, 1\}00\{01111, 00111, 00011, 00001\}$$
$$\cup 110\{0, 1\}11\{01111, 00111, 00011, 00001\})B^\omega$$

5.3 *Constraints on several signals*

We now define the language associated with a constraint on a set of k signals. We first introduce some auxiliary notation. Let A be an alphabet. For each integer k, A_k denotes the alphabet consisting of k-uple of letters of A, denoted as a column matrix. For instance, B_8 is the set of bytes, and the triple $\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ is a letter of B_3. Thus

$$u = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

is a word on the alphabet B_3. By reading in parallel the lines of the previous representation, one gets the three words 10110, 11000 and 10000. Therefore the word u given above can be represented by the triple (10110, 11000, 10000) of words of B^*. More generally, it is always possible to represent a word of length n on the alphabet A_k as a k-array of words of A^*.

$$\begin{pmatrix} a_{1,1} \\ a_{1,2} \\ \vdots \\ a_{1,k} \end{pmatrix} \begin{pmatrix} a_{2,1} \\ a_{2,2} \\ \vdots \\ a_{2,k} \end{pmatrix} \cdots \begin{pmatrix} a_{n,1} \\ a_{n,2} \\ \vdots \\ a_{n,k} \end{pmatrix}$$

We denote by $\pi_A : A_k^* \to \underbrace{A^* \times A^* \times \cdots \times A^*}_{k \text{ times}}$ the function defined by

$$\pi_A \left(\begin{pmatrix} a_{1,1} \\ a_{1,2} \\ \vdots \\ a_{1,k} \end{pmatrix} \begin{pmatrix} a_{2,1} \\ a_{2,2} \\ \vdots \\ a_{2,k} \end{pmatrix} \cdots \begin{pmatrix} a_{n,1} \\ a_{n,2} \\ \vdots \\ a_{n,k} \end{pmatrix} \right) =$$

$$(a_{1,1}a_{2,1} \cdots a_{n,1},\ a_{1,2}a_{2,2} \cdots a_{n,2},\ \ldots,\ a_{1,k}a_{2,k} \cdots a_{n,k})$$

This function π_A is in fact a monoid morphism of A_k^* into $A^* \times A^* \times \cdots \times A^*$: this simply means that it preserves the concatenation product. However, it is not an isomorphism (except if $k = 1$) because an element of $A^* \times A^* \times \cdots \times A^*$ may have components of different length. Let

$$D_k(A) = \{(u_1, u_2, \ldots, u_k) \in A^* \times A^* \cdots \times A^* \mid |u_1| = |u_2| = \ldots = |u_k|\}$$

denote the set of k-tuples of words of the same length. Now, since π_A induces an isomorphism from A_k^* onto $D_k(A)$, one can identify the k-tuples of $D_k(A)$ with the words of A_k^*.

Returning once again to signals, a *static multicolumn* is a couple $M = (p, R)$ where p is a positive integer and $R = (R_1, \ldots, R_k)$ is a k-uple of static rows. Intuitively, to each row corresponds a signal, but it is important to observe that two rows or more can represent the same physical signal. This allows one to impose several distinct constraints on a given signal and to conveniently display hypothesis-conclusion pairs, when a signal can figure in one set of hypotheses and in another set of conclusions. By definition, the language associated with a static multicolumn (p, R) is

$$L(p, R) = C_k^p \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)$$

In other words, the k-tuples $(u_1, u_2, \ldots, u_k)$ such that $u_1 \in L(R_1), u_2 \in L(R_2), \cdots, u_k \in L(R_k)$ and $|u_1| = |u_2| = \ldots = |u_k| = p$ are selected and identified with words of C_k^*.

A *dynamic multicolumn* is a triple $M = (n, m, R)$ where n is a integer, m is either an integer or the symbol $*$ and $R = (R_1, \ldots, R_k)$ is a k-tuple of dynamic rows. Define

$$C_k^{[0,m]} = \bigcup_{0 \le i \le m} C_k^i.$$

The language associated with a dynamic multicolumn is by definition

$$L(n, m, R) = C_k^n\Big(C_k^{[0,m]} \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big),$$
$$L(n, *, R) = C_k^n\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big).$$

The difference between the types of multicolumns is that, in a dynamic multicolumn, there may be no upper bound on the common length of the u_i's.
One can show for the multicolumns a result similar to the one obtained for rows

PROPOSITION 4
The languages associated with a static multicolumn and their valuations are finite languages. The languages associated with a dynamic multicolumn and their valuations are languages of level 3/2.

Proof. Let $M = (p, R)$ be a static multicolumn. Then the language associated with M is a subset of C_k^p and hence is finite. The valuations are subsets of B_k^p and are also finite. The case of a dynamic multicolumn $M = (n, m, R)$, where m is an integer, is similar.

Finally, let $M = (n, *, R)$ be a dynamic multicolumn. Then $L(M) = C_k^n\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big)$. Since C_k^n is a finite language, it is of level 1 by proposition 1. By the same proposition, the languages of level 3/2 are closed under intersection and product and it remains to show that the language $\pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)$ is of level 3/2. Denote by π_i the ith projection of C_k^* on C^*, defined by $\pi_i(c_1, c_2, \ldots, c_k) = c_i$. We first observe that

$$\pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big) = \bigcap_{1 \le i \le k} \pi_i^{-1}(L(R_i)).$$

Indeed, the above language is actually the set of k-tuples $(u_1, u_2, \ldots, u_k)$ such that $|u_1| = |u_2| = \ldots = |u_k|$ and $u_i \in L(R_i)$ for $1 \le i \le k$. Now the languages $L(R_i)$ are of level 3/2 by proposition 3, and by proposition 1, so are the languages $\pi_i^{-1}(L(R_i))$ and their intersection. Therefore, $\pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)$ is of level 3/2, as required.

Let $\nu : C \to B$ be a valuation. By definition, one has

$$L_\nu(M) = \nu\Big(C_k^n\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big)\Big)$$
$$= B_k^n\,\nu\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big).$$

A lemma is in order to treat this expression:

Lemma 1. Let $\nu : C \to B$ be a valuation and let $L_1, L_2, \ldots, L_k$ be languages of C^*. Then the following formula holds

$$\nu\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big)$$

$$= B_k^* \cap \bigcap_{1 \le i \le k} \pi_i^{-1}(\nu(L_i)).$$

Proof. One has successively

$$\nu\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big) =$$

$$\nu\left\{ \begin{pmatrix} c_{1,1} \\ c_{1,2} \\ \vdots \\ c_{1,k} \end{pmatrix} \begin{pmatrix} c_{2,1} \\ c_{2,2} \\ \vdots \\ c_{2,k} \end{pmatrix} \cdots \begin{pmatrix} c_{m,1} \\ c_{m,2} \\ \vdots \\ c_{m,k} \end{pmatrix} \mid m \ge 0 \text{ and } \right.$$

$$\left. c_{1,1}c_{2,1}\cdots c_{m,1} \in L_1,\ \ldots,\ c_{1,k}c_{2,k}\cdots c_{m,k} \in L_k \right\} =$$

$$\left\{ \begin{pmatrix} \nu(c_{1,1}) \\ \nu(c_{1,2}) \\ \vdots \\ \nu(c_{1,k}) \end{pmatrix} \begin{pmatrix} \nu(c_{2,1}) \\ \nu(c_{2,2}) \\ \vdots \\ \nu(c_{2,k}) \end{pmatrix} \cdots \begin{pmatrix} \nu(c_{m,1}) \\ \nu(c_{m,2}) \\ \vdots \\ \nu(c_{m,k}) \end{pmatrix} \mid m \ge 0 \text{ and } \right.$$

$$\left. c_{1,1}c_{2,1}\cdots c_{m,1} \in L_1,\ \ldots,\ c_{1,k}c_{2,k}\cdots c_{m,k} \in L_k \right\} =$$

$$\left\{ \begin{pmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,k} \end{pmatrix} \begin{pmatrix} b_{2,1} \\ b_{2,2} \\ \vdots \\ b_{2,k} \end{pmatrix} \cdots \begin{pmatrix} b_{m,1} \\ b_{m,2} \\ \vdots \\ b_{m,k} \end{pmatrix} \mid m \ge 0 \text{ and } \right.$$

$$\left. b_{1,1}b_{2,1}\cdots b_{m,1} \in \nu(L_1),\ \ldots,\ b_{1,k}b_{2,k}\cdots b_{m,k} \in \nu(L_k) \right\} =$$

$$\{u_1u_2\cdots u_m \mid m \ge 0 \text{ and } \pi_1(u_1u_2\cdots u_m) \in \nu(L_1),\ \ldots,$$
$$\pi_k(u_1u_2\cdots u_m) \in \nu(L_k)\} = B_k^* \cap \bigcap_{1 \le i \le k} \pi_i^{-1}\nu(L_i) \qquad \square$$

Let us achieve the proof of proposition 4. By lemma 1, one has

$$L_\nu(M) = B_k^n\, \nu\Big(C_k^* \cap \pi_C^{-1}\big(L(R_1) \times L(R_2) \times \cdots \times L(R_k) \cap D_k(C)\big)\Big)$$
$$= B_k^n\Big(B_k^* \cap \bigcap_{1 \le i \le k} \pi_i^{-1}(L_\nu(R_i))\Big).$$

The languages $L_\nu(R_i)$ are of level 3/2 by Proposition 3, B_k^n is finite and B_k^* is of level 1. Since the languages of level 3/2 are closed under intersection, the previous formula shows that $L_\nu(M)$ is of level 3/2. $\square$

5.4 *Timing diagrams and properties*

A *timing diagram* (TD) is a sequence of multicolumns. A *property* is a pair $P = (M, N)$ of timing diagrams: $M = (M_1, M_2, \ldots M_r)$ is the hypothesis and $N = (N_1, N_2 \ldots N_r)$ is the conclusion. A property defines a particular binary relation on k-tuples of signals. The property is satisfied if every k-tuple of signals that satisfies the hypothesis satisfies the conclusion, too.

In the rational language formalism, this can be translated as follows: an infinite word w on the alphabet B_k satisfies a property $P = (M, N)$ if, for each suffix s of w and for all valuations ν, if there exists a factorization $u_1u_2 \cdots u_ru_{r+1}$ of s, where $u_1 \in L_\nu(M_1), u_2 \in L_\nu(M_2), \ldots, u_r \in L_\nu(M_r), u_{r+1} \in B_k^\omega$, then there exists a factorization $u'_1u'_2 \cdots u'_ru'_{r+1}$ of s, where $u'_1 \in L_\nu(M_1) \cap L_\nu(N_1), u'_2 \in L_\nu(M_2) \cap L_\nu(N_2), \ldots, u'_r \in L_\nu(M_r) \cap L_\nu(N_r)$, $u'_{r+1} \in B_k^\omega$. This can be reformulated as follows.

Theorem 1. *An infinite word w on the alphabet B_k satisfies P if and only if none of its suffixes belong to the set*

$$K(P) = \bigcup_{\nu \text{ valuation}} L_\nu(M_1)L_\nu(M_2) \cdots L_\nu(M_r)B_k^\omega \setminus \Big(\big(L_\nu(M_1) \cap L_\nu(N_1)\big)\big(L_\nu(M_2) \cap L_\nu(N_2)\big) \cdots \big(L_\nu(M_r) \cap L_\nu(N_r)\big)\Big)B_k^\omega.$$

Proof. It is easier to consider the negation of the condition. By definition, an infinite word w on the alphabet B_k does not satisfy P if and only if there exist a suffix s of w, and a valuation ν such that there exists a factorization of s of the form $u = u_1u_2 \cdots u_ru_{r+1}$ with

$$u_1 \in L_\nu(M_1), u_2 \in L_\nu(M_2), \ldots, u_r \in L_\nu(M_r), u_{r+1} \in B_k^\omega$$

but such that

$$s \notin \big(L_\nu(M_1) \cap L_\nu(N_1)\big)\big(L_\nu(M_2) \cap L_\nu(N_2)\big) \cdots \big(L_\nu(M_r) \cap L_\nu(N_r)\big)\Big)B_k^\omega.$$

The formula of the statement follows immediately. □

COROLLARY 1
An infinite word w on the alphabet B_k satisfies a property P if and only if it belongs to the set $L(P) = B_k^\omega \setminus B_k^*K(P)$.

We arrive at our main result.

Theorem 2. *For every property P, the set $L(P)$ is a star-free set of level* 3.

Proof. Proposition 4 shows that the languages $L_\nu(M_i)$ and $L_\nu(N_i)$ are of level 3/2. Since the languages of level 3/2 are closed under intersection and product, the sets $L_\nu(M_1)L_\nu(M_2) \cdots L_\nu(M_r)B^\omega$ and $\Big(\big(L_\nu(M_1) \cap L_\nu(N_1)\big)\big(L_\nu(M_2) \cap L_\nu(N_2)\big) \cdots \big(L_\nu(M_r) \cap L_\nu(N_r)\big)\Big)B^\omega$ are also of level 3/2. Therefore $K(P)$ is of level 2 and $L(P)$ is of level 3. □

5.5 *Constraints*

We call a *constraint* a finite sequence of properties. Let $(P_1, P_2, \ldots, P_n)$ be a constraint. Let $L(P_1), L(P_2), \ldots, L(P_n)$ be the sets of infinite words defined by $P_1, P_2, \ldots, P_n$, respectively. Then the set of words defined by $(P_1, P_2, \ldots, P_n)$ is the language $L(P_1) \cap L(P_2) \cap \ldots \cap L(P_n)$. In other words, a constraint is a conjunction of properties. Now, the languages of level 3 are closed under intersection. Therefore, theorem 2 implies the following result.

COROLLARY 2
The set of infinite words defined by a constraint is a star-free language of level 3.

6. An example

This section is devoted to the detailed study of an example. Consider a car washing machine. We propose to specify its control system using chronograms. The wash does not take more than one car at a time. There is a gate at the entrance. This gate is closed while a car is in the wash and opened if the wash is empty. Moreover, there are two switches set respectively at the entrance and at the exit of the wash. These switches may be either on or off at any instant, subject to the constraint that the one at the entrance toggles every time a car enters the wash, and the one at the exit toggles every time a car exits from the wash.

The car wash control system can be modelled by three boolean signals denoted by B, I, and O. The signal B (*Entrance*) carries the 0 value to model the opened gate and the 1 value to model the closed gate. The signals I (*In*) and O (*Out*) model the entrance switch and exit switch, respectively. Initially, the value of the signals B, I, and O is set to 0, which means the wash is empty and its gate is open.

The following five properties specify the car wash control system. This set of properties may be neither consistent nor minimal. The set of figures below show the chronograms for these properties. An automaton model of the system induced from these properties is also proposed. Then we prove that a sixth constraint is effectively satisfied by the automaton. Here are the first five properties.

(1) During any instant, if the gate is opened, I and O carry the same value.

(2) During any instant, if I and O carry the same value, the gate is opened.

(3) If the gate is closed during an instant t, then I is stable between t and $t + 1$ (since the machine cannot wash more than one car at a time).

(4) If the gate is open during an instant t, then O is stable between t and $t + 1$ (since the machine is empty).

(5) As soon as a car enters the wash, the gate closes.

The property to be proved is the following:

(6) When the gate is closed, the car that is in the machine will eventually exit.

The chronograms of these properties are drawn in the figures 3 to 5. An explanation is in order for the chronogram representing property 6 (figure 5). Indeed, the clock

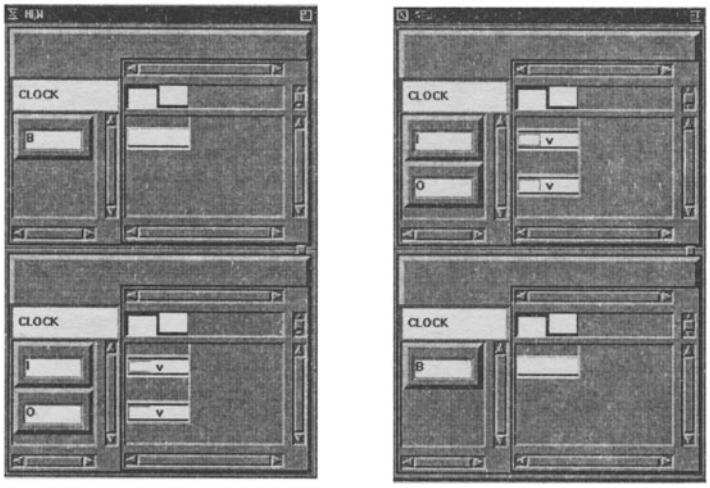


Figure 3. The chronograms of properties 1 and 2.

signal seems to count off one time unit between v and $\bar{v}$. However, since the signals I and O carry the value IRRELEVANT during the same period, the clock signal is irrelevant. Thus property 6 is an unbounded liveness property.

The semantics of these chronograms can be expressed by rational ω-expressions. The basic alphabet is $B_3 = \{0, 1\}^3$. Each matrix $\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$ represents one of the value of the $\begin{pmatrix} B \\ I \\ O \end{pmatrix}$ triple. The set of these matrices is the alphabet of the language on which the previous properties are defined. In the following definitions, v, v_1, v_2, etc. will denote boolean variables.

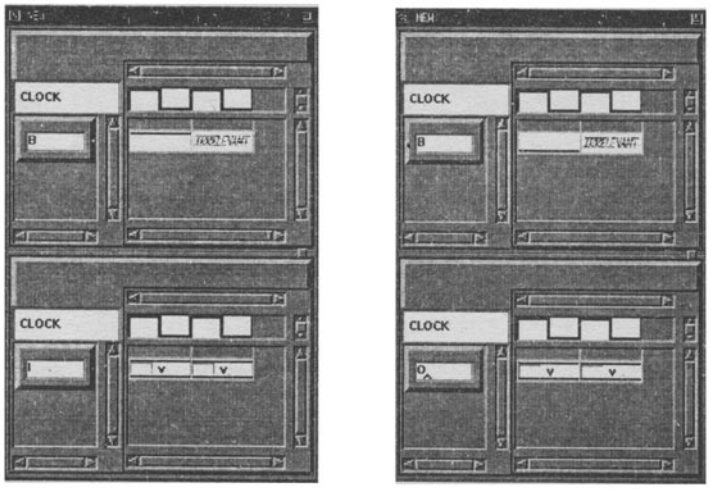


Figure 4. The chronograms of properties 3 and 4.

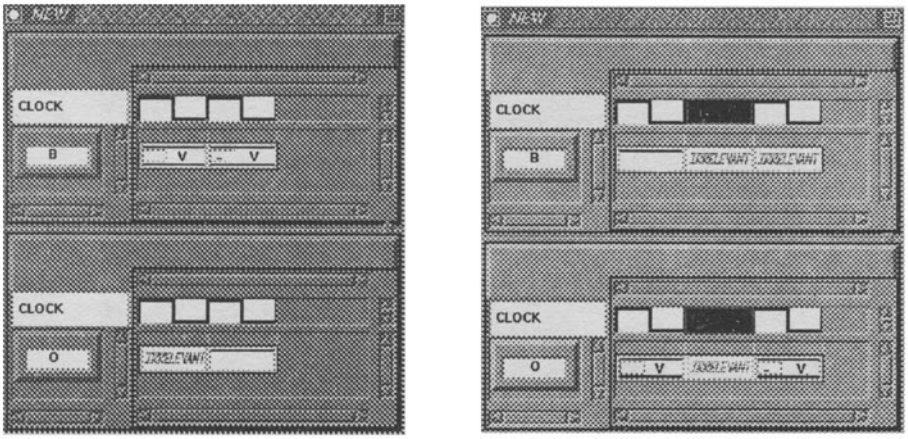


Figure 5. The chronograms of properties 5 and 6.

a) The first constraint states that if $B = 0$, then I and O carry the same value. In other words, the letters $\begin{pmatrix}0\\0\\1\end{pmatrix}$ and $\begin{pmatrix}0\\1\\0\end{pmatrix}$ cannot occur. Thus the set C_1 associated with the first constraint is defined by

$$C_1 = \left\{ \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix} \right\}^{\omega}.$$

b) The second constraint states that if I and O carry the same value, then $B = 0$. In other words, the letters $\begin{pmatrix}1\\0\\0\end{pmatrix}$ and $\begin{pmatrix}1\\1\\1\end{pmatrix}$ cannot occur, either. Therefore, the set C_2 associated with the second constraint is defined by

$$C_2 = \left\{ \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix} \right\}^{\omega}.$$

c) The third constraint states that if the letter $u(t)$ is equal to $\begin{pmatrix}1\\v\\v_1\end{pmatrix}$, then the letter $u(t+1)$ is equal to $\begin{pmatrix}v_2\\v\\v_3\end{pmatrix}$. What can be written as $B_3^{\omega} \setminus B_3^* K_1 B_3^{\omega}$, where

$$K_1 = \left\{ \begin{pmatrix}1\\1\\v_1\end{pmatrix}\begin{pmatrix}v_2\\0\\v_3\end{pmatrix}, \begin{pmatrix}1\\0\\v_1\end{pmatrix}\begin{pmatrix}v_2\\1\\v_3\end{pmatrix} \mid v_1, v_2, v_3 \in B \right\}.$$

d) The fourth constraint states that if the letter $u(t)$ is equal to $\begin{pmatrix}0\\v_1\\v\end{pmatrix}$, then the letter

$u(t+1)$ is equal to $\begin{pmatrix} v_2 \\ v_3 \\ v \end{pmatrix}$. What can be written $B_3^\omega \setminus B_3^* K_2 B_3^\omega$, where

$$K_2 = \left\{ \begin{pmatrix} 0 \\ v_1 \\ 0 \end{pmatrix} \begin{pmatrix} v_2 \\ v_3 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ v_1 \\ 1 \end{pmatrix} \begin{pmatrix} v_2 \\ v_3 \\ 0 \end{pmatrix} \mid v_1, v_2, v_3 \in B \right\}.$$

e) The fifth constraint states that if $u(t)$ is equal to $\begin{pmatrix} v_1 \\ v \\ v_2 \end{pmatrix}$, and if $u(t+1)$ is equal to $\begin{pmatrix} v_3 \\ \bar{v} \\ v_4 \end{pmatrix}$, then $v_3 = 1$. This can be written as $B_3^\omega \setminus B_3^* K_3 B_3^\omega$, where

$$K_3 = \left\{ \begin{pmatrix} v_1 \\ v \\ v_2 \end{pmatrix} \begin{pmatrix} 0 \\ \bar{v} \\ v_3 \end{pmatrix} \mid v, v_1, v_2, v_3 \in B \right\}.$$

Let us consider the system specified by these five properties. The first two ones define the following set of words

$$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right\}^\omega.$$

The last three ones define the following set of words $B_3^\omega \setminus B_3^* K B_3^\omega$, where

$$K = K_1 \cup K_2 \cup K_3.$$

The five properties altogether define the following set of words $S = C^\omega \setminus C^* R C^\omega$, where

$$C = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right\}$$

and

$$R = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

Since the initial value of the three boolean signals is set to 0, the car wash control system can be represented by the automaton shown in figure 6.

Note that the states of this automaton are solutions of the equation $I + O = B \bmod 2$.

Consider the sixth property, which is a liveness property. It says that, given an instant t such that $u(t)$ is of the form $\begin{pmatrix} 1 \\ v_2 \\ v_3 \end{pmatrix}$, then there exists a subsequent instant s ($s > t$) such that

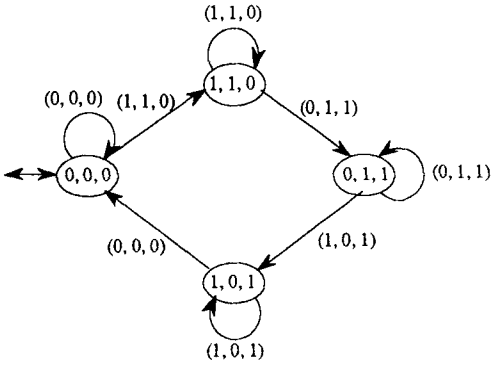


Figure 6. The automaton of the five properties.

$u(s)$ is equal to $\begin{pmatrix} v_1 \\ v_2' \\ \bar{v}_3 \end{pmatrix}$. Thus, property (6) is described by a rational expression involving six multicolumns: M_1, M_2, M_3, N_1, N_2, N_3. The languages they define are, respectively,

$$L(M_1) = \left\{ \begin{pmatrix} 1 \\ v_2 \\ v_3 \end{pmatrix} \mid v_2, v_3 \in B \right\} \qquad L(M_2) = B_3^* \qquad L(M_3) = B_3,$$

$$L(N_1) = \left\{ \begin{pmatrix} v_1 \\ v_2' \\ v \end{pmatrix} \mid v_1, v_2' \in B \right\} \qquad L(N_2) = B_3^*$$

$$L(N_3) = \left\{ \begin{pmatrix} v_1' \\ v_2'' \\ \bar{v} \end{pmatrix} \mid v_1', v_2'' \in B \right\}.$$

Let $K(P)$ be the language representing the property. It follows

$$K(P) = K_1 K_2^\omega \cup K_3 K_4^\omega,$$

where

$$K_1 = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\} \qquad K_2 = \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

$$K_3 = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\} \qquad K_4 = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\}$$

It is theoretically possible to compute the automaton associated with $K(P)$, and then to prove that S is a subset of $B_3^\omega \setminus B_3^* K(P)$, where $B_3^\omega \setminus B_3^* K(P)$ is given by corollary 1. More directly, one can observe that $S \subset B_3^\omega \setminus B_3^* K(P)$ is equivalent to $S \cap B_3^* K(P) = \emptyset$ which can be rewritten into

$$T \cap K(P) = \emptyset,$$

where the set $T = \{u \in B_3^\omega \mid vu \in S \text{ for some } v \in B_3^*\}$ is recognized by the Büchi automaton (Thomas 1990) given in figure 6, by taking all the states as initial and final states. Recall that an infinite word u is accepted by a Büchi automaton if there is *at least*

an infinite run with label u starting at some initial state and visiting a final state infinitely often. Thus the equality $T \cap K(P) = \emptyset$ can be directly verified on this automaton, since no word of $K(P)$ can have an infinite run on the Büchi automaton for T.

7. Conclusion

We have presented a new formal language for the specification of temporal properties of Discrete Event Dynamic Systems. This language, called the Chronogram Language, is based on a well-known graphic metaphor: waveforms. It allows specifying certain complex temporal properties in a more convenient way than textual temporal logics (CTL, CTL* ...). Although we do not consider this language as a universal one, we think that its graphical approach might be appealing to designers. In fact, we view the chronogram language as a basic part of a future Computer Aided Design (CAD) environment including validation tools. Several authors have developed similar work (Borriello 1992; Cingel 1993; Coombes & McDermid 1993; Dillon *et al* 1994; Helbig *et al* 1993; Khordoc *et al* 1991; Tiedemann 1992; Tiedemann *et al* 1992). It would be too long to compare in detail these related works with ours. However, the idea of using rational expressions to define the semantics seems to be new in this context and can probably be successfully applied in other cases. This is not really surprising, since the equivalence between proportional (linear) temporal logic, first order logic over the non negative integers with signature $(<, R_a(a \in A))$ (where R_a is a predicate giving the positions of the letter a in a given infinite word of A^ω) and star-free sets of infinite words is a well-known fact.

The Chronogram Language is graphic and fully declarative. In this paper, we defined rigorously its semantics by using automata theory. The main result of this work is that it is possible to associate a finite automaton with any chronogram. This means that chronograms are ω-rational. In fact, as shown in this paper, chronograms correspond to a much smaller class that the class of ω-rational sets, and this may lead to some specific compilation algorithms in the future.

We are now studying new developments: an extension of the Chronogram Language allowing designers to specify properties without any reference to some clock or including timing aspects (having physical time delays) and a consistency checking tool for sets of chronograms. We are also working on the improvement of the compilation algorithm since it is crucial to compile chronograms into as small as possible automata. Currently, compilers generate VHDL code and Signal code. New output languages will also be available in the future.

Thanks are due to the anonymous referees for their numerous suggestions and remarks that greatly improved the quality of this paper.

References

Antoine C, Le Goff L 1991 Timing diagrams for writing and checking logical and behavioral

properties of integrated systems. *CHARME'91, Correct hardware design methodologies* (eds) P Prineto, P Camurati (Turin, Italy: Elsevier) pp 441–453

Ashcroft E A, Wadge W W 1976 Lucid - a formal system for writing and proving programs. *SIAM J. Comput.* 5: 336–354

Borriello G 1992 Formalized timing diagrams. In *Proceedings of the European Conference on Design Automation*, Los Alamitos, CA (IEEE Comput. Soc. Press) pp 372–377

Brzozowski J A, Knast R 1978 The dot-depth hierarchy of star-free languages is infinite, *J. Comput. Syst. Sci.* 16: 37–55

Caspi P, Halbwachs N 1986 A functional model for describing and reasoning about time behaviour of computing systems. *Acta Info.* 22: 595–627

Cingel V 1993 A graph-based method for timing diagrams representation and verification. In *CHARME'93, Correct hardware design and verification methods* (eds) G J Milne, L Pierre (Berlin: Springer) pp 1–14

Coombes A C, McDermid J A 1993 Using diagrams to give a formal specification of timing constraints in Z. In *Proceedings of the Seventh Annual Z User Meeting* (eds) J P Bowen, J E Nicholls (Berlin: Springer-Verlag) pp 119–130

Dillon L K, Kutty G, Melliar-Smith P M, Moser L E *et al* 1994 Visual specifications for temporal reasoning. *J. Visual Languages Comput.* 5: 61–81

Eilenberg S 1974 *Automata, languages and machines* (New York: Academic Press) vol. A

Eilenberg S 1976 *Automata, languages and machines* (New York: Academic Press) vol. B

Helbig J, Schlor R, Damm W, Dohmen G *et al* 1993 VHDL/S – integrating statecharts, timing diagrams and VHDL. *MicroProcess. Microprogramming* 38: 571–580

Khordoc K, Dufresne M, Cerny E 1991 A stimulus/response system based on hierarchical timing diagrams. In *1991 IEEE International Conference on Computer-Aided Design. Digest of Technical papers*, Los Alamitos, CA (IEEE Comput. Soc.) pp 358–361

Le Goff B, Benveniste A, Figueira C, Le Guernic P, 1989 CAD environment for real-time control system. In *American Control Conference ACC'89* (Pittsburgh, PA: IEEE)

Lipsett R, Schaeffer C F, Ussery C 1990 *VHDL: Hardware description and design* (Boston, MA: Kluwer)

Perrin D 1990 *Automata*. In *Handbook of theoretical computer science. Vol B: Formal models and semantics* (ed.) J Van Leeuwen (Amsterdam: Elsevier)

Perrin D, Pin J E 1986 First order logic and star-free sets. *J. Comput. Syst. Sci.* 32: 393–406

Thomas W 1982 Classifying regular events in symbolic logic. *J. Comput. Syst. Sci.* 25: 360–375

Thomas W 1990 Automata on infinite objects. In *Handbook of theoretical computer science. Vol B: Formal models and semantics* (ed.) J Van Leeuwen (Amsterdam: Elsevier)

Tiedemann W-D 1992 Bus protocol conversion: from timing diagrams to state machines. In *Computer aided systems theory, EUROCAST'91* (eds) F Pichler, R M Diaz (Berlin: Springer) pp 365–377

Tiedemann W-D, Lenk S, Grobe C, Grass W 1993 Introducing structure into behavioural descriptions obtained from timing diagram specifications. *MicroProcess. Microprogramming* 38: 581–588