# Modèle de contraintes temporelles pour systèmes polychrones

**Charles André — Frédéric Mallet**

*Université de Nice Sophia Antipolis & INRIA Sophia Antipolis Méditerranée*
*Laboratoire I3S (UMR 6070 CNRS) — 06902 Sophia Antipolis cedex*
*{Prénom.Nom}@sophia.inria.fr*

RÉSUMÉ. *La modélisation des systèmes répartis et des systèmes électroniques modernes nécessite des référentiels temporels multiples. Nous désignons ces systèmes sous le nom de "systèmes polychrones". Le profil UML pour les systèmes temps réel et embarqués (MARTE) permet leur modélisation ainsi que la spécification de contraintes temporelles avec CCSL (Clock Constraint Specification Language). Dans MARTE, CCSL est non normatif et sa sémantique est informelle. Nous proposons ici une sémantique formelle en termes d'évolutions d'un "Time System" pour un noyau de CCSL. Un "Time System" est un modèle dynamique qui associe un ensemble de configurations à un modèle structurel constitué d'un ensemble d'horloges discrètes et de relations sur ces horloges. Les Time Systems sont comparés à d'autres modèles de causalités asynchrones, synchrones et polychrones. CCSL et sa mise en œuvre sont illustrés sur un exemple de contrôleur d'ABS.*

ABSTRACT. *"Polychronous systems" are systems referring to multiple, usually interdependent, time bases. The UML profile for real-time and embedded system (MARTE) can deal with such systems, and the Clock Constraint Specification Language (CCSL), part of MARTE, can be used to specify temporal constraints. In MARTE, the semantics of CCSL is informal. In this paper, we propose a formal semantics of a kernel of CCSL, in terms of "Time System" evolutions. A Time System is a dynamic model which associates a set of configurations with a structural model made of discrete clocks and relationships among these clocks. Time Systems are compared to other existing causality models. An ABS controller illustrates the use of CCSL.*

MOTS-CLÉS : *modèle de temps, contraintes temporelles, parallélisme, polychronie*

KEYWORDS: *Time model, Time constraint, concurrency, polychrony*

JESA – 43/2009. MSR 2009, pages 725 à 739

## 1. Introduction

La modélisation des systèmes répartis et des systèmes électroniques, multi-cœurs ou multi-domaines d'horloge, nécessite des référentiels temporels multiples. Nous désignons ces systèmes sous le nom de *systèmes polychrones*. Le profil UML (OMG, 2007) pour les systèmes temps réel et embarqués (MARTE) (OMG, 2009) permet leur modélisation ainsi que la spécification de contraintes temporelles avec CCSL (Clock Constraint Specification Language).

Cet article introduit deux modèles. Un modèle structurel appelé *Clock Model* pour représenter des domaines d'horloges discrètes et un modèle dynamique appelé *Time System* obtenu en associant à un *Clock Model* une configuration et des règles d'évolution. Chaque horloge est une suite, généralement infinie, d'instants. Trois relations sont définies entre instants : la précédence, la coïncidence et l'exclusion. Ces relations s'étendent aux horloges et définissent des contraintes d'horloges que l'on peut qualifier d'asynchrones, synchrones et mixtes. Les premières s'appuient sur des précédences, les secondes sur des coïncidences, les dernières combinent les deux formes. CCSL a été créé pour spécifier ces contraintes d'horloges.

Le *Time System* reprend des concepts de la théorie du parallélisme (*concurrency*) et des approches réactives synchrones (mono- ou polychrones). Une évolution est une suite de réactions, chacune respectant toutes les contraintes d'horloges. Une réaction est caractérisée par un ensemble d'horloges franchies *simultanément* et peut modifier l'ensemble de contraintes temporelles. Le franchissement d'une horloge consiste à faire progresser le référentiel temporel représenté par cette horloge (passage à l'instant suivant).

Dans MARTE, CCSL est non normatif et sa sémantique, exprimée en anglais, est informelle. La relation entre MARTE et CCSL, ainsi qu'une caractérisation informelle de CCSL, ont déjà été présentées (André *et al.*, 2007). Nous proposons ici d'exprimer formellement la sémantique d'un sous-ensemble de CCSL en termes d'évolutions de *Time Systems*. Le sous-ensemble choisi est suffisant pour comprendre les spécificités de CCSL et sa mise en œuvre.

Le papier est organisé de la façon suivante. La section 2 situe nos travaux par rapport aux modèles de causalité asynchrones (*e.g.,* Réseaux de Petri), synchrones (*e.g.,* Esterel, Lustre) et polychrones (*e.g.,* Signal, Esterel multi-clocks). Les modèles formels *Clock Model* et *Time System* sont définis dans la section 3. La détermination effective de sous-ensembles d'horloges franchissables est détaillée dans la section suivante. Elle utilise un solveur booléen à base de BDD (Binary Decision Diagram). Diverses politiques sont proposées pour choisir un ensemble d'horloges franchies simultanément. L'usage de CCSL est illustré dans la section 5. L'application, empruntée au domaine automobile, est un contrôleur d'ABS qui est modélisé (pour ce qui concerne les aspects temporels), simulé et analysé dans l'environnement TimeSquare (`http://www.inria.fr/sophia/aoste/dev/time_square`), spécialement développé pour la modélisation et l'analyse avec MARTE et CCSL.

## 2. CCSL et modèles connexes

Le modèle de temps défini dans MARTE est *multiforme*, c'est-à-dire que, vis-à-vis du système modélisé, tout événement peut définir un référentiel temporel (base de temps). Les instants de cette base de temps correspondent aux occurrences de l'événement associé et sont donc *strictement ordonnés*. Le temps dit physique (une fois discrétisé) est un cas particulier pour lequel les occurrences de l'événement sont directement liées au temps physique. Si il n'existe pas de relation directe avec le temps physique on parle alors de temps *logique*.

CCSL permet de *spécifier* et d'*analyser* le comportement temporel de systèmes dont la modélisation inclut plusieurs (généralement de nombreux) référentiels temporels interdépendants. CCSL n'est pas un langage de programmation : il se limite aux aspects contrôle. Les traitements ne sont considérés qu'au travers d'événements particuliers représentant par exemple les activations d'un traitement, la réception ou l'émission d'une donnée. En fait, CCSL permet d'exprimer des *synchronisations complexes* entre systèmes communicants, il se rapproche ainsi des *contrôleurs* définis dans le modèle de composants pour systèmes embarqués hétérogènes appelé 42 (Maraninchi *et al.*, 2009). Par son caractère événementiel et réactif, CCSL se rapproche aussi des modèles tels que les réseaux de Petri (Reisig, 1985) et les modèles réactifs synchrones (Benveniste *et al.*, 2003).

CCSL utilise le terme d'horloge plutôt que celui de référentiel temporel. Une *horloge* est un ensemble strictement ordonnée d'instants. Cet ensemble peut être dense ou discret. Seul ce dernier cas est considéré dans cet article. Les instants sont alors indicés par les entiers naturels et pour une horloge $c$, $c[k]$ avec $k \in \mathbb{N}^\star$ désigne le $k^{\text{e}}$ instant de $c$. Une spécification CCSL impose des relations entre des instants appartenant à des horloges différentes. Ces relations sont la précédence ($\prec$), la coïncidence ($\equiv$) et l'exclusion ($\#$). La première exprime généralement une relation de causalité ; la seconde est une relation de dépendance très forte imposant la coïncidence de deux instants distincts ; la troisième exclut la coïncidence entre instants. Les horloges peuvent être considérées comme des processus concurrents dont les actions se réduisent au passage à leur instant suivant. Les contraintes imposées limitent les évolutions possibles. Les réseaux de Petri aussi bien que les *Tag Systems* (Benveniste *et al.*, 2004) peuvent modéliser de tels systèmes, toutefois aucun n'est pleinement satisfaisant pour exprimer CCSL.

### 2.1. *Réseaux de Petri*

Même si notre modèle de temps a fortement été inspiré des travaux de Petri sur la théorie du parallélisme (Petri, 1987), une différence conceptuelle majeure les sépare. Pour Petri, la coïncidence de deux événements ne peut désigner qu'un seul et unique événement se produisant en un seul point espace-temps. Pour nous, la coïncidence est une contrainte particulière imposée entre deux instants, sans remettre en cause leur individualité.

Les réseaux de Petri ont des fondements mathématiques bien établis et offrent des possibilités d'analyse riches qui en font un modèle très utilisé. Les réseaux de Petri permettent la modélisation du parallélisme (*concurrency*) et peuvent facilement représenter certaines de nos relations d'horloges purement asynchrones (celles liées à la relation de précédence). Cependant, la coïncidence ne peut pas être exprimée facilement car il est impossible de forcer le franchissement "simultané" de deux transitions distinctes.

Des extensions temporelles des réseaux de Petri existent. Dans un travail précédent (Mallet *et al.*, 2009) nous avons comparé l'expressivité de l'extension temporelle proposée par Merlin (Merlin, 1974) à celle de notre modèle. Cette extension associe aux transitions un intervalle temporel (deux dates $a$ et $b$, telles que $0 \leq a \leq b$ avec $b$ possiblement non bornée). Les dates $a$ et $b$ pour une transition $t$ sont relatives à l'instant $\theta$ auquel la transition $t$ devient validée. $t$ ne doit pas être franchie avant le temps $\theta + a$ et doit forcément être franchie avant ou au temps $\theta + b$, à moins que la transition ne soit invalidée entre temps par le franchissement d'une autre transition. Avec un tel modèle, deux transitions sont franchies simultanément si elles sont franchies à la même date. En cas de conflits, l'ordre de franchissement est important et un mécanisme de choix similaire au nôtre doit être mis en place. La version moderne des réseaux de Petri temporels (Berthomieu *et al.*, 2006) propose des extensions telles que les priorités et les arcs inhibiteurs ce qui permet de résoudre certains conflits. Notre analyse a montré que même si la coïncidence pouvait être exprimée avec un tel modèle, sous certaines conditions, elle nécessitait parfois des mécanismes complexes pour modéliser certaines de nos expressions primitives comme l'échantillonnage. Faire de la coïncidence une primitive rend les modèles plus simples.

### 2.2. *Modèles réactifs synchrones*

La relation de coïncidence est la relation fondatrice pour les langages synchrones (Benveniste *et al.*, 2003). Ces langages représentent les évolutions d'un système comme des séquences (infinies) de réactions disjointes. Une réaction est caractérisée par des occurrences simultanées (c'est-à-dire se produisant au même instant). Cette description s'appuie implicitement sur une hypothèse de synchronisation globale. Le modèle *polychrone réactif*, sous-jacent à un langage comme Signal, renonce à cette hypothèse (Benveniste *et al.*, 2004). Ses évolutions sont déclenchées par des arrivées de données ou par des événements endogènes. Les langages polychrones tels qu'Esterel multi-horloges (Berry *et al.*, 2001) et surtout Signal (Benveniste *et al.*, 1991) combinent des opérateurs synchrones avec des opérateurs asynchrones pour modéliser, entre autres, des systèmes de type GALS (Globalement Asynchrones Localement Synchrones). Par son caractère relationnel, Signal peut exprimer des dépendances entre horloges beaucoup plus riches qu'Esterel. CCSL est proche de Signal, toutefois il s'en distingue sur plusieurs points. Tout d'abord, les applications sont sensiblement différentes. Signal manipule des séquences de valeurs de même type appelées signaux. Les signaux sont *a priori* indépendants ; deux signaux n'ont donc pas nécessairement

des valeurs aux mêmes instants. L'ensemble des instants où un signal est présent définit l'*horloge* du signal. Afin de faire des calculs pertinents sur les signaux, un calcul sur les horloges est (souvent) nécessaire. En revanche, CCSL manipule uniquement les horloges et se désintéresse des valeurs. Le concept d'*event* de Signal ou celui de *signal pur* en Esterel sont beaucoup plus proches de notre concept d'horloge. Signal a deux types d'opérateurs. Les opérateurs synchrones s'appliquent à des signaux synchrones, c'est-à-dire qui ont les mêmes horloges. Les opérateurs polychrones manipulent des signaux avec des horloges différentes et peuvent produire des résultats qui ont une horloge également différente. CCSL a uniquement des opérateurs polychrones.

Le compilateur Signal analyse les spécifications qui lui sont fournies, il vérifie leur cohérence et signale les éventuelles anomalies. Il peut également générer un code exécutable lorsque la spécification est *endochrone*, c'est-à-dire que l'on peut construire une horloge plus fine que toutes les autres horloges, à partir de laquelle toutes les autres horloges peuvent être dérivées. Ce choix délibéré répond à l'exigence de produire une exécution déterministe. CCSL pour sa part considère les spécifications telles qu'elles sont et vérifient qu'elles ne présentent pas de contradictions sans juger de leur complétude. En cas de situations non déterministes, des politiques sont appliquées pour choisir une solution. Ce papier et la majorité des travaux sur CCSL s'intéressent à la simulation, cependant nous avons également exploré des possibilités de validation formelle avec CCSL (André *et al.*, 2009).

## 3. Modèles d'horloges et de contraintes d'horloges

### 3.1. *Clock Model et Time System*

Un *Clock Model* $\mathcal{M} = \langle \mathcal{C}, \mathcal{P}, \mathcal{S} \rangle$ est un triplet constitué d'un ensemble fini non vide d'horloges discrètes $\mathcal{C}$, un ensemble fini $\mathcal{P}$ de variables typées utilisées comme paramètres dans certaines contraintes d'horloges et d'un ensemble fini de contraintes sur ces horloges $\mathcal{S}$. Il s'agit d'un modèle structurel.

Soit $\mathcal{BW}$ l'ensemble des mots binaires finis ou infinis ($\mathcal{BW} = \{0,1\}^* \cup \{0,1\}^\omega$). Un *Time System* est un triplet constitué d'un Clock Model et deux applications $\sigma$ (schedule) et $\chi$ (configuration) :

$$\mathcal{T} = \langle \mathcal{M}, \sigma, \chi \rangle \text{ avec } \mathcal{M} = \langle \mathcal{C}, \mathcal{P}, \mathcal{S} \rangle ;\ \sigma : \mathcal{C} \to \mathcal{C} \times \mathcal{BW};\ \chi : \mathcal{C} \to \mathbb{N}$$

Une configuration $\chi$ de $\mathcal{T}$ est un uplet d'entiers naturels tel que pour toute horloge $c \in \mathcal{C}$, $\chi(c)$ est l'indice de l'instant courant de l'horloge $c$. La configuration initiale $\chi^0$ est telle que $(\forall c \in \mathcal{C})\chi^0(c) = 0$. $\sigma$ est une fonction *partielle* qui associe à une horloge, une autre horloge prise pour référence et un mot binaire. Elle se nomme *schedule* car elle permet de planifier les évolutions futures de l'horloge par rapport à celles de l'horloge de référence. Ainsi $\sigma$, lorsqu'il est défini, modélise pour une horloge donnée, l'échéancier de ses synchronisations futures avec une autre horloge.

Les évolutions d'un Time System $\mathcal{T}$ sont représentées par des *runs*. Un run est une exécution particulière, constituée d'une séquence, généralement infinie, de *réactions*.

Lors d'une réaction, sous un environnement $\rho$ (application qui assigne des valeurs aux éléments de $\mathcal{P}$), les horloges appartenant à un ensemble $F \subseteq \mathcal{C}$ passent à leur instant suivant. Par la suite, nous dirons que l'horloge est *franchie* et $F$ est appelé ensemble d'horloges franchies. Une réaction cause un changement d'état du Time System. Cet état comprend la configuration, elle-même liée à l'état courant de chaque horloge. Il comprend également les mots binaires associés par $\sigma$ à certaines horloges. Une réaction correspond donc à une transition du Time System, denotée comme indiqué dans l'équation 1.

$$\langle \mathcal{M}, \sigma, \chi \rangle \xrightarrow[\rho]{F} \langle \mathcal{M}, \sigma', \chi' \rangle \quad [1]$$

Dans la nouvelle configuration $\chi'$ l'indice de l'instant courant de chaque horloge ayant été franchie est incrémenté de 1 (équation 2).

$$\chi'(c) = \text{ si } c \in F \text{ alors } \chi(c) + 1 \text{ sinon } \chi(c) \quad [2]$$

Les changements dans les mots binaires seront détaillés plus loin (*cf.* section 3.2). $\rho$ affecte une valeur aux variables du Clock Model. Nous nous limitons au cas où toutes les variables sont des entiers naturels ($\rho : \mathcal{P} \rightarrow \mathbb{N}$). Un *run* s'écrit alors

$$\left\langle \mathcal{M}, \sigma^0, \chi^0 \right\rangle \xrightarrow[\rho_1]{F_1} \left\langle \mathcal{M}, \sigma^1, \chi^1 \right\rangle \xrightarrow[\rho_2]{F_2} \cdots \left\langle \mathcal{M}, \sigma^{n-1}, \chi^{n-1} \right\rangle \xrightarrow[\rho_n]{F_n} \langle \mathcal{M}, \sigma^n, \chi^n \rangle \cdots \quad [3]$$

L'ensemble des horloges franchies $F$ doit bien sûr satisfaire les contraintes d'horloges $\mathcal{S}$. Il existe généralement plusieurs ensembles possibles. Nous dénotons le fait que $F$ soit franchissable pour $\chi$ et $\sigma$, sous l'environnement $\rho$, par les expressions :

$$\langle \mathcal{M}, \sigma, \chi \rangle \xrightarrow[\rho]{F} \quad \text{ou plus elliptiquement par } \mathcal{T} \xrightarrow[\rho]{F}$$

Pour déterminer ces ensembles, nous allons définir une application $[\![\ ]\!]$ qui associe une expression booléenne à tout quadruplet (contrainte d'horloge, schedule, configuration, environnement). Pour alléger les notations, les trois derniers arguments seront laissés implicites et $[\![\mathcal{S}, \sigma, \chi, \rho]\!]$ sera noté simplement $[\![\mathcal{S}]\!]$. La détermination des ensembles franchissables se ramène alors à un problème de satisfaction. Soit $\pi : \mathcal{C} \rightarrow \mathscr{C}$ une bijection entre l'ensemble $\mathcal{C}$ des horloges du Time System et un ensemble $\mathscr{C}$ de variables booléennes. Une solution $f : \mathscr{C} \rightarrow \{0, 1\}$ est interprétée comme un ensemble $F$ d'horloges franchissables tel que $F = \left\{ c \in \mathcal{C} \mid f\big(\pi(c)\big) = 1 \right\}$ (équation 4).

$$(\forall f : \mathscr{C} \rightarrow \{0,1\}) \left( \left\langle \langle \mathcal{C}, \mathcal{P}, \mathcal{S} \rangle, \sigma, \chi \right\rangle \xrightarrow[\rho]{F} \right) \Leftrightarrow \left( [\![\mathcal{S}]\!] (f) = 1 \right)$$
$$\text{avec } F = \left\{ c \in \mathcal{C} | f\big(\pi(c)\big) = 1 \right\} \quad [4]$$

Dans la sous-section suivante, nous définissons l'application $[\![\ ]\!]$ pour un sous-ensemble très réduit de CCSL. Un rapport de recherche (André, 2009) contient une spécification complète pour un noyau de CCSL.

## 3.2. *Syntaxe et sémantique de CCSL*

### 3.2.1. *Syntaxe*

La grammaire BNF du sous-ensemble CCSL considéré est donnée dans le tableau 1. CC est un non-terminal abréviation de *Clock Constraint*. Une contrainte d'horloge est constituée d'une ou plusieurs *Clock Relations* (CR) en parallèle. Une relation d'horloges peut être réduite à la référence à une horloge (C). Les autres sont binaires. Elles définissent successivement les relations de précédence entre horloges, de sous-horloge, d'égalité. La dernière prend comme membre gauche une horloge et comme membre droit une *Clock Expression* (CE). L'opérateur $\triangleq$ se lit « est définie par ». Pour cette présentation, nous n'avons retenu que quatre expressions d'horloges qui sont nommées sup, inf, filtrage et retard, respectivement. BW est un mot binaire. Le non-terminal nat peut être soit un littéral dénotant un entier naturel, soit une variable (qui ne peut être que du type entier naturel dans ce sous-ensemble de CCSL).

| | | |
|---|---|---|
| CC | ::= | CR ( \| CR )* |
| CR | ::= | C \| C $\boxed{\prec}$ C \| C $\boxed{\subset}$ C \| C $\boxed{=}$ C \| C $\boxed{\triangleq}$ CE |
| CE | ::= | C $\vee$ C \| C $\wedge$ C \| C $\blacktriangledown$ BW \| C ( nat ) $\rightsquigarrow$ C |
| nat | ::= | naturel \| Var |

**Tableau 1.** *Syntaxe (simplifiée) de* CCSL

Partant d'une spécification écrite dans ce langage, pour chaque définition de filtrage ou de retard, on définit l'application $\sigma$. Par exemple, pour les relations de définition d'horloges $c_1 \boxed{\triangleq} c_2 \blacktriangledown w$ et $c_3 \boxed{\triangleq} c_4(n) \rightsquigarrow c_5$, on définit la fonction partielle $\sigma$ par $\sigma : c_1 \mapsto (c_2, w)$ et $\sigma : c_3 \mapsto (c_5, 0^\omega)$. $0^\omega$ est le mot binaire infini qui ne comporte que des 0. Le rôle de ces initialisations apparaîtra lors de la définition des règles de réécriture.

### 3.2.2. *Sémantique*

Nous donnons tout d'abord une caractérisation mathématique des contraintes définies ci-dessus, puis nous donnerons les règles de traduction en expressions booléennes. La composition parallèle de contraintes d'horloge impose la conjonction des contraintes. Les relations d'horloge imposent des relations entre paires d'instants. $c_1$ précède $c_2$ (équation 5) se dit aussi « $c_1$ est plus rapide que $c_2$ ». Pour l'équation 6 il convient de rajouter que la correspondance entre les instants de $c_1$ et $c_2$ préserve l'ordre, c'est-à-dire que si $c_1[k_1] \equiv c_2[j_1]$ et $c_1[k_2] \equiv c_2[j_2]$ alors $k_1 < k_2 \Leftrightarrow j_1 < j_2$.

$$c_1 \boxed{\prec} c_2 \Leftrightarrow \forall k \in \mathbb{N}^\star . c_1[k] \prec c_2[k] \qquad (c_1 \text{ précède } c_2) \qquad [5]$$

$$c_1 \boxed{\subset} c_2 \Leftrightarrow \forall k \in \mathbb{N}^\star . \exists j \in \mathbb{N}^\star . c_1[k] \equiv c_2[j] \qquad (c_1 \text{ sous-horloge de } c_2) \qquad [6]$$

$$c_1 \boxed{=} c_2 \Leftrightarrow \forall k \in \mathbb{N}^\star . c_1[k] \equiv c_2[k] \qquad (c_1 \text{ et } c_2 \text{ sont synchrones}) \qquad [7]$$

sup et inf se rapportent à la relation précède. Dans l'équation 8, $c$ est la plus rapide des horloges plus lentes que $c_1$ et $c_2$. De même, dans l'équation 9, $c$ est la plus lente des horloges plus rapides que $c_1$ et $c_2$. Pour toute paire d'horloges, sup et inf sont toujours définis. Le filtrage (équation 10) définit une sous-horloge particulière. L'opération $w \uparrow k$ retourne l'indice du $k^{\text{e}}$ 1 de $w$.

$$c \boxed{\triangleq} c_1 \vee c_2 \Leftrightarrow \forall \chi \in \mathbb{N}^{\mathcal{C}}.\chi(c) = \min\{\chi(c_1), \chi(c_2)\} \qquad \text{(sup de } c_1 \text{ et } c_2\text{)} \qquad [8]$$

$$c \boxed{\triangleq} c_1 \wedge c_2 \Leftrightarrow \forall \chi \in \mathbb{N}^{\mathcal{C}}.\chi(c) = \max\{\chi(c_1), \chi(c_2)\} \qquad \text{(inf de } c_1 \text{ et } c_2\text{)} \qquad [9]$$

$$c \boxed{\triangleq} c_1 \blacktriangledown w \Leftrightarrow \forall k \in \mathbb{N}^{\star}.c_1[k] \equiv c[w \uparrow k] \qquad (c_1 \text{ filtrée par } w) \qquad [10]$$

La formulation du retard est plus complexe, elle pourra être déduite des règles sémantiques données plus loin. Nous donnons ici sa sémantique en français. $c$ définie par $c_1(n) \rightsquigarrow c_2$ impose que $c$ soit une sous-horloge de $c_2$. $c_1$ sert de déclencheur. A chaque franchissement de $c_1$, on enregistre dans le *schedule* de $c$ une prévision de franchissement en coïncidence avec le $n^{\text{e}}$ franchissement strictement futur de $c_2$. Si $n$ n'est pas une constante, sa valeur $\rho(n)$ est donnée par l'environnement lors du franchissement de $c_1$. La figure 1 illustre le cas $c \boxed{=} c_1(2) \rightsquigarrow c_2$ avec visualisation de relations entre instants. Le chronogramme montre bien le caractère polychrone de cette relation qui est plus générale que le retard de Signal. $c \boxed{=} c_1(2) \rightsquigarrow c_1$ serait quasiment l'équivalent du délai de Signal (c := c1$2).

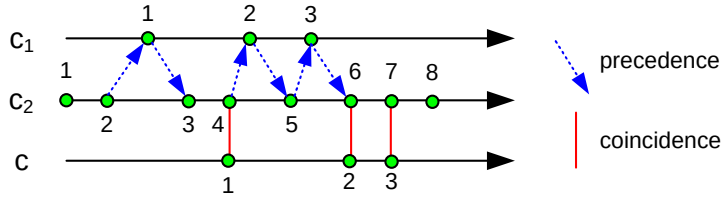


**Figure 1.** *Opérateur retard*

### 3.2.3. *Interprétation booléenne*

Les relations sur les horloges ($c \in \mathcal{C}$) se traduisent par des opérations booléennes sur les variables associées ($\mathsf{c} \in \mathscr{C}$). Dans les expressions booléennes, nous utilisons les opérateurs logiques $\Rightarrow$ (implication), $=$ (égalité) tels que pour toute expression booléenne $t_1$, $t_2$ : $t_1 \Rightarrow t_2 \Leftrightarrow \neg t_1 \vee t_2$, $t_1 = t_2 \Leftrightarrow (t_1 \wedge t_2) \vee (\neg t_1 \wedge \neg t_2)$. $[\![c]\!]$ se transforme en $\pi(c)$. Pour alléger les règles de transformation nous notons $\mathsf{c}$ (fonte droite et sans empattement) pour $\pi(c)$.

$$[\![\mathsf{SCR}_1 \mid \mathsf{SCR}_2]\!] = \left([\![\mathsf{SCR}_1]\!] \wedge [\![\mathsf{SCR}_2]\!]\right) \qquad \text{(parallèle)}$$

$$[\![c_1 \boxed{\prec} c_2]\!] = \Big(\big(\chi(c_1) = \chi(c_2)\big) \Rightarrow \neg \mathsf{c}_2\Big) \qquad \text{(précède)}$$

$$[\![c_1 \boxed{\subset} c_2]\!] = (\mathsf{c}_1 \Rightarrow \mathsf{c}_2) \qquad \text{(sous-horloge)}$$

$$[\![c_1 \boxed{=} c_2]\!] = (\mathsf{c}_1 = \mathsf{c}_2) \qquad \text{(synchronie)}$$

$$\left[\!\!\left[c \boxed{\triangleq} c_1 \vee c_2\right]\!\!\right] = \Big(\mathsf{c} = \big((\chi(c_1) = \chi(c)) \Rightarrow \mathsf{c}_1\big) \wedge \big((\chi(c_2) = \chi(c)) \Rightarrow \mathsf{c}_2\big)\Big) \text{(sup)}$$

$$\left[\!\!\left[c \boxed{\triangleq} c_1 \wedge c_2\right]\!\!\right] = \Big(\mathsf{c} = \big((\chi(c) = \chi(c_1)) \Rightarrow \mathsf{c}_1\big) \vee \big((\chi(c) = \chi(c_2)) \Rightarrow \mathsf{c}_2\big)\Big) \text{(inf)}$$

Pour les deux expressions d'horloges qui suivent il existe en plus des règles de réécriture conditionnelles pour $\sigma$ que l'on note *ancienne* $\twoheadrightarrow$ *nouvelle* valeur.

$$\left[\!\!\left[c \boxed{\triangleq} c_1 \blacktriangledown w\right]\!\!\right] = \Big(\mathsf{c} = \big((\sigma(c) = 1.v) \wedge \mathsf{c}_1\big)\Big) \qquad \text{(filtre)}$$

$$\sigma(c) : (c_1, b.v) \twoheadrightarrow (c_1, v) \text{ si } c_1 \in F \qquad \text{(rw-filtre)}$$

$$\left[\!\!\left[c \boxed{\triangleq} c_1 \ (\ n\ )\!\rightsquigarrow c_2\right]\!\!\right] = \Big(\mathsf{c} = \big((\sigma(c) = 1.v) \wedge \mathsf{c}_2\big)\Big) \qquad \text{(retard)}$$

$$\sigma(c) : (c_2, b.v) \twoheadrightarrow \begin{cases} (c_2, v) & \text{si } c_1 \notin F \wedge c_2 \in F, \\ \big(c_2, v \text{ or } 0^{[\![n]\!]-1}.1.0^{\omega}\big) & \text{si } c_1 \in F \wedge c_2 \in F, \\ \big(c_2, b.v \text{ or } 0^{[\![n]\!]-1}.1.0^{\omega}\big) & \text{si } c_1 \in F \wedge c_2 \notin F. \end{cases} \qquad \text{(rw-retard)}$$

or est le ou logique bit à bit sur les mots binaires.

### 3.2.4. *Extensions*

A partir de ces contraintes, il est possible d'en construire de nouvelles. Nous illustrons par deux contraintes utilisées dans la section 5. La relation asynchrone d'*alternance* dont la sémantique est donnée par l'équation 11, peut être spécifiée en CCSL à l'aide de la relation précède et d'une horloge auxiliaire $c$ qui est synchrone avec l'horloge $a$ retardée d'un instant (équation 12).

$$a \boxed{\sim} b \Leftrightarrow \forall k \in \mathbb{N}^{\star}.\big(a[k] \prec b[k]\big) \wedge \big(b[k] \prec a[k+1]\big) \qquad [11]$$

$$c \boxed{\triangleq} a \blacktriangledown 0.1^{\omega} \mid a \boxed{\prec} b \mid b \boxed{\prec} c \qquad [12]$$

Un autre exemple est le sup à 3 horloges. On utilise l'associativité de cette opération pour le définir à partir du sup binaire. Les équations 13 et 14 précisent la sémantique et la spécification en CCSL.

$$d \boxed{=} \mathrm{sup3}(a, b, c) \Leftrightarrow \forall \chi \in \mathbb{N}^{\mathcal{C}}.\chi(d) = \min\{\chi(a), \chi(b), \chi(c)\} \qquad [13]$$

$$e \boxed{\triangleq} a \vee b \mid f \boxed{\triangleq} e \vee c \mid d \boxed{=} f \qquad [14]$$

## 4. Détermination des réactions

### 4.1. *Ensembles remarquables*

#### 4.1.1. *Définitions*

Soit $\mathscr{V}$ l'ensemble des valuations $v$ qui satisfont l'expression booléenne $[\![\mathcal{S}]\!]$ (équation 15). De $\mathscr{V}$ on déduit l'ensemble des ensembles d'horloges franchissables $\mathcal{F}$ (équation 16).

$$\mathscr{V} \triangleq \left\{v : \mathscr{C} \to \{0,1\} \mid [\![\mathcal{S}]\!]\,(v) = 1\right\} \quad [15]$$

$$\mathcal{F} \triangleq \{|v| \mid v \in \mathscr{V} \wedge |v| \neq \emptyset\} \text{ avec } |v| \triangleq \{c \in \mathcal{C} \mid v\big(\pi(c)\big) = 1\} \quad [16]$$

Les horloges qui sont *validées*, c'est-à-dire potentiellement franchissables, constituent l'ensemble $E$ (Enabled). Les horloges qui sont en aucun cas franchissables sont rassemblées dans l'ensemble $D$ (Disabled).

$$E \triangleq \bigcup_{F \in \mathcal{F}} F \qquad D \triangleq \mathcal{C} \setminus E$$

#### 4.1.2. *Exemple*

Reprenons l'exemple du sup de trois horloges (équation 13). Son expression en CCSL a nécessité l'introduction de 2 horloges auxiliaires $e$ et $f$. Il faut construire

$$[\![\mathcal{S}]\!] = \left[\!\!\left[ e \boxed{\triangleq} a \vee b \mid f \boxed{\triangleq} e \vee c \mid d \boxed{=} f \right]\!\!\right]$$

Pour la configuration $\chi$ telle que $\chi(a) = 2$ et $\chi(\_) = 1$ pour toutes les autres horloges, les règles de transformations conduisent à $[\![\mathcal{S}]\!] = (\mathsf{e} = \mathsf{b}) \wedge (\mathsf{f} = \mathsf{e} \wedge \mathsf{c}) \wedge (\mathsf{d} = \mathsf{f})$. Partant de cette expression booléenne on déduit (avec l'aide des techniques décrites dans la sous-section suivante) $\mathcal{F} = \big\{\{a,b,c,d,e,f\}, \{b,c,d,e,f\}, \{a,b,e\}, \{b,e\}, \{a\}, \{a,c\}, \{c\}\big\}$, $E = \{a,b,c,d,e,f\}$, $D = \emptyset$.

On constate que n'importe quel sous-ensemble $E'$ de $E$ ne constitue pas forcément un ensemble d'horloges franchissables : par exemple $E' = \{d\} \subset E$ (franchissement de l'horloge $d$ uniquement) n'est pas dans $\mathcal{F}$. La raison est que certaines horloges sont liées (pour une réaction donnée) par des relations de dépendance qui ont été imposées lors de la traduction. C'est le cas de la relation logique $\mathsf{d} = \mathsf{f}$ qui a pour conséquence que ou bien $d$ et $f$ sont simultanément dans un ensemble d'horloges franchissables, ou bien aucune n'apparaît dans un tel ensemble. Pour choisir parmi les solutions possibles, différentes stratégies sont envisageables. Une solution *minimale* $F_{\min}$ est telle que pour tout $F \in \mathcal{F}$, $F \subseteq F_{\min} \Rightarrow F = F_{\min}$. $\{a\}$, $\{c\}$ et $\{b,e\}$ sont des solutions minimales. Une solution *maximale* $F_{\max}$ est telle que pour tout $F \in \mathcal{F}$, $F_{\max} \subseteq F \Rightarrow F = F_{\max}$. $\{a,b,c,d,e,f\}$ est solution maximale. Une solution $F_{\text{dep}}$ est *strictement dépendante* pour une horloge $c$ si $(\forall c' \in F_{\text{dep}})(\forall F \in \mathcal{F})\ c \in F \Rightarrow c' \in F$. $F_{\text{dep}} = \{b,e\}$ est strictement dépendante pour $b$ et pour $e$, par contre $\{a,b,e\}$

ne l'est pas. La stratégie qui consiste à choisir aléatoirement une solution de $\mathcal{F}$ est nommée *random*. Le coût du calcul de ces solutions est très différent et sera évoqué dans la sous-section suivante.

### 4.2. *Détermination d'un ensemble d'horloges franchissables*

Nous avons construit un solveur basé sur les ROBDD (Reduced Ordered Binary Decision Diagram) (Brace *et al.*, 1990).

#### 4.2.1. *Algorithmes*

Soit S le ROBDD représentant $[\![\mathcal{S}]\!]$. Pour trouver une solution $v$ qui satisfasse S il suffit de faire une traversée du ROBDD depuis son sommet jusqu'au nœud terminal 1. A chaque nœud interne, lorsqu'un choix est possible entre les successeurs, un tirage aléatoire est effectué. La fonction AnySat, généralement disponible dans les bibliothèques d'implantation de BDD, implémente cette traversée. Dans la solution renvoyée, des variables peuvent ne pas avoir été affectées. Nous avons créé une fonction ComputeRandom, qui appelle AnySat et donne aléatoirement la valeur 0 ou 1 à ces variables. ComputeRandom peut dans certains cas renvoyer un ensemble vide, solution qui a été exclue dans la définition donnée pour $\mathcal{F}$ (équation 16). On peut modifier ComputeRandom pour qu'il signale le blocage du Time System si cette solution nulle est l'unique solution. Si ce n'est pas le cas, il doit faire un autre choix aléatoire. Une autre façon de résoudre ce problème est de rajouter une horloge Always qui est franchie dans toutes les réactions. Cette horloge est l'équivalent du "tick" dans le langage Esterel. Elle se définit en CCSL en disant que cette horloge est l'*union* [1] de toutes les autres horloges. En ajoutant la contrainte booléenne $Always = 1$, on exclut les réactions vides.

#### 4.2.2. *Complexité*

Deux facteurs interviennent dans la complexité : le nombre de variables que l'on note $n = |\mathcal{C}|$ et le nombre de nœuds dans le BDD, noté $|\mathsf{S}|$ pour le ROBDD S. Les implémentations de BDD font intervenir de nombreuses optimisations et utilisent en particulier des techniques de programmation dynamique qui peuvent améliorer considérablement l'efficacité des calculs (Meinel *et al.*, 2007).

Le temps de calcul pour la satisfaction, c'est-à-dire la détermination d'une solution quelconque est en $\mathcal{O}(n)$. La détermination des ensembles $E$ et $D$ peut se faire en testant pour chaque variable c, le cofacteur de $[\![\mathcal{S}]\!]$ pour c (fonction obtenue en remplaçant toutes les occurrences de c par 1). Lorsque ce cofacteur est la fonction nulle, aucune solution ne contient c sous sa forme positive, donc l'horloge correspondante $c$

1. l'opérateur d'union d'horloges n'a pas été introduit dans cette présentation, mais il fait partie des opérateurs du noyau de CCSL.

n'est dans aucun ensemble d'horloges franchissables. En BDD, le cofacteur se réalise par une opération appelée *restriction*.

$$\forall c \in \mathcal{C},\ \text{si RESTRICT}(\mathsf{S}, \mathsf{c}, 1) = \mathbf{0} \text{ alors } c \text{ est ajouté à } D \text{ sinon } c \text{ est ajouté à } E$$

La complexité en temps pour calculer une restriction est linéaire en taille du BDD (pour les versions avec programmation dynamique). On a donc une complexité en $\mathcal{O}(n * |\mathsf{S}|)$. En fait, on veut simplement savoir si le cofacteur est nul. Ce test est bien moins coûteux que la construction effective du cofacteur lorsqu'on utilise l'algorithme dit *ite_constant* (Brace *et al.*, 1990). La caractérisation des stratégies minimale, maximale ou strictement dépendante fait intervenir l'ensemble $\mathcal{F}$. La recherche de toutes les solutions (fonction AllSat) étant exponentielle, il faut absolument l'éviter. Des solutions sont proposées dans un rapport (André, 2009).

## 5. Illustration : ABS

Pour illustrer l'utilisation de CCSL, nous considérons un contrôleur ABS (*Anti-lock Braking System*) d'une automobile. Cet exemple et les exigences temporelles associées sont extraits du rapport final du projet ATESST (Johansson *et al.*, 2008). Les entrées de ce système sont fournies par quatre capteurs qui mesurent la vitesse de rotation de chacune des roues : $i_{avg}$ (avant gauche), $i_{avd}$ (avant droite), $i_{arg}$ (arrière gauche), $i_{ard}$ (arrière droite). Les cinq sorties concernent la vitesse du véhicule ($o_{vit}$) et la force de freinage à appliquer sur chacune des roues : $o_{avg}$, $o_{avd}$, $o_{arg}$, $o_{ard}$.

Le système est *a priori* asynchrone et le calcul est effectué périodiquement de façon cyclique. Les capteurs sont répartis dans le véhicule ce qui signifie que leur valeur n'est pas disponible nécessairement simultanément. De même, les actionneurs sont également répartis et le temps entre la commande et l'action varie selon les actionneurs. Le rapport ATESST définit un ensemble d'exigences temporelles sur ce système et nous nous proposons de modéliser ces exigences avec CCSL sans se poser la question de leur pertinence. La première exigence impose d'exécuter un cycle avec une période ($R$) égale à 5 ms. Une deuxième exigence impose une durée d'échantillonnage ($L_s$) comprise entre 1 et 3 ms. Ce qui signifie que la première donnée du capteur doit parvenir au contrôleur d'ABS entre 1 et 3 ms après le début de la période. Le délai de synchronisation des entrées ($J_{ii}$) impose une borne maximum de 0.5 ms entre l'arrivée de la première entrée et celle de la dernière. Un exigence similaire sur les sorties ($J_{oo}$) impose une borne maximum de 0.5 ms entre l'action due à la première sortie et celle due à la dernière. Le calcul effectif de la fonction ABS ($L_{io}$) dure entre 3 et 5 ms. Cette durée exclut les délais de synchronisation des entrées et des sorties.

Pour modéliser ces exigences avec CCSL, il faut déterminer les éléments du modèle à associer à une horloge. Chaque cycle de calcul réalise l'échantillonnage des entrées, la transmission de ces valeurs au contrôleur, le calcul de la fonction ABS, l'émission des sorties et leur transmission vers les actionneurs. A chaque capteur et actionneur on associe une horloge (9 horloges en tout). Les instants de ces horloges modélisent les instants de capture des entrées ou d'action sur les sorties. Ces horloges portent

le même nom que le capteur/actionneur associé. Enfin, l'horloge $abs$ représente le début du calcul de l'ABS et $R$ est le début du cycle de calcul. Les exigences formulées sont alors de deux types. Le premier type concerne la période de répétition qui définit une distance minimum et maximum entre les instants successifs de l'horloge $R$, à la fois une période et une propriété de sporadicité. Le deuxième type concerne toutes les autres exigences et met en relation les instants de même indice d'horloges différentes. Il s'agit également d'une distance, mais entre le $i^{e}$ tic d'une horloge et le $i^{e}$ tic d'une ou plusieurs autres horloges. Dans tous les cas, les distances en question sont exprimées en millisecondes. En CCSL, cela se traduit par l'utilisation d'une horloge dite chronométrique discrète et dont les tics successifs sont espacés d'une durée physique pré-déterminée (la période). L'opérateur discretizedBy de CCSL permet cela. Cet opérateur n'a pas été discuté ici car il n'a aucune influence sur le calcul d'horloges présenté. La seule influence de cet opérateur est de forcer une représentation régulièrement espacée des instants successifs des horloges chronométriques.

La spécification CCSL de l'exemple est donnée ci-dessous. Les alternances à la fin garantissent que le cycle est respecté, c'est-à-dire que toutes les horloges évoluent au même rythme.

$$c_{10} \boxed{=} IdealClk \text{ discretizedBy } 10^{-4} \quad \text{(10kHz)}$$

$$c_1 \boxed{=} c_{10} \blacktriangledown (1.0^9)^{\omega} \quad \text{(1kHz)}$$

$$R \boxed{=} c_1 \blacktriangledown (1.0^4)^{\omega} \quad \text{(R=5ms)}$$

$$i_{inf} \boxed{=} i_{avd} \wedge i_{avg} \wedge i_{ard} \wedge i_{arg} \quad \text{(première entrée)}$$

$$i_{sup} \boxed{=} i_{avd} \vee i_{avg} \vee i_{ard} \vee i_{arg} \quad \text{(dernière entrée)}$$

$$o_{inf} \boxed{=} o_{avd} \wedge o_{avg} \wedge o_{ard} \wedge o_{arg} \quad \text{(première sortie)}$$

$$o_{sup} \boxed{=} o_{avd} \vee o_{avg} \vee o_{ard} \vee o_{arg} \quad \text{(dernière sortie)}$$

$$(i_{sup}(3) \rightsquigarrow c_1) \boxed{\prec} o_{inf} \boxed{\prec} (i_{sup}(5) \rightsquigarrow c_1) \quad (\mathsf{L_{io}}=]3,5[\text{ ms})$$

$$(R(1) \rightsquigarrow c_1) \boxed{\prec} i_{inf} \boxed{\prec} (R(3) \rightsquigarrow c_1) \quad (\mathsf{L_{s}}=]1,3[\text{ ms})$$

$$i_{sup} \boxed{\prec} (i_{inf}(5) \rightsquigarrow c_{10}) \quad (\mathsf{J_{ii}}=0.5\text{ms})$$

$$o_{sup} \boxed{\prec} (o_{inf}(5) \rightsquigarrow c_{10}) \quad (\mathsf{J_{oo}}=0.5\text{ms})$$

$$R \boxed{\sim} i_{inf} \mid R \boxed{\sim} o_{sup} \mid i_{sup} \boxed{\sim} o_{abs} \mid abs \boxed{\sim} o_{inf}$$

Après analyse de ces contraintes, le solveur de TIMESQUARE a produit l'évolution donnée dans la figure 2. Si aucune horloge ne peut évoluer alors la simulation s'arrête et indique un blocage du système. TIMESQUARE produit une sortie au format VCD (Value Change Dump) qui est un standard, défini par l'IEEE, très utilisé dans le domaine de l'EDA. L'éditeur VCD de TIMESQUARE permet de visualiser les contraintes pour justifier la simulation. Les flèches en pointillés matérialisent les relations de précédence et les flèches verticales (rouges) matérialisent les coïncidences requises par

la spécification. Les impulsions pleines (vertes) sont des “fantômes” que l’on peut cacher ou montrer. Un fantôme indique une horloge active qui n’a pas été franchie dans la réaction. En sur-impression sur la capture d’écran, ont été rajoutées des barres de cotations qui reflètent certaines contraintes de la spécification ($L_{io}$ et $J_{oo}$).

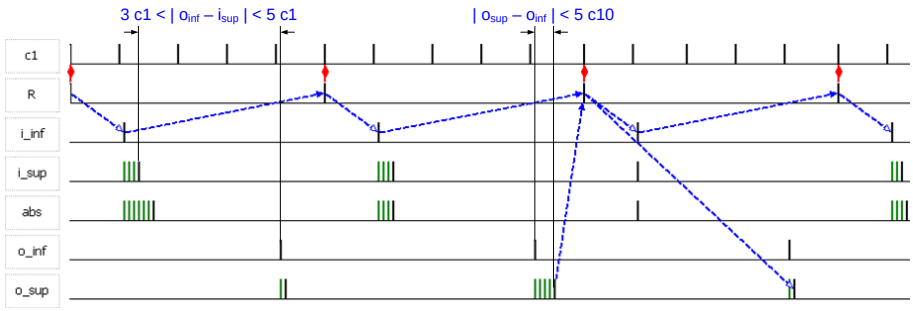


**Figure 2.** *Simulation avec TimeSquare*

## 6. Conclusion

MARTE a défini le langage de spécification d’exigences temporelles CCSL. Nous avons introduit ici les modèles formels “Clock Model” (structurel) et “Time System” (dynamique) sur lesquels repose CCSL. Ces modèles empruntent des caractéristiques à la fois aux modèles du parallélisme (Réseaux de Petri) et au réactif synchrone. Ils rejoignent ainsi les formalismes réactifs polychrones. CCSL se différentie en proposant des primitives qui sont impossibles à exprimer avec les langages synchrones et sont problématiques même pour les langages polychrones. De plus, CCSL admet les spécifications non déterministes et propose des politiques de choix. Enfin, contrairement aux autres modèles polychrones, CCSL est un modèle *acausal* (au sens de Modélica), *i.e.,* ses horloges sont indifféremment des entrées ou des sorties.

La définition d’une sémantique opérationnelle permet de rendre exécutables les spécifications CCSL. La mise en œuvre s’appuie sur un ensemble de plugins Eclipse appelé TIMESQUARE. TIMESQUARE contient un éditeur de contraintes intégré à l’environnement PapyrusUML, un solveur BDD pour le calcul d’horloges et une visualisation interactive des chronogrammes résultants.

Deux extensions sont en cours d’étude. La première répond à un souci pragmatique d’amélioration de la convivialité de TIMESQUARE en permettant la construction de bibliothèques utilisateurs pour un domaine spécifique (DSL) et en réalisant un couplage fort avec les modèles comportementaux d’UML pour les animer. La deuxième, plus fondamentale, vise à faciliter l’analyse formelle de spécifications CCSL par des outils externes (*Model Checker*). Dans ce but, nous voulons identifier un sous-ensemble de CCSL suffisamment expressif (non complétement déterministe) mais autorisant une caractérisation formelle et efficace de l’espace des solutions.

## 7. Bibliographie

André C., Syntax and Semantics of the Clock Constraint Specification Language (CCSL), Research Report n° 6925, INRIA and University of Nice, May, 2009.

André C., Mallet F., « Specification and Verification of Time Requirements with CCSL and Esterel », *LCTES*, ACM, p. 167-176, June, 2009.

André C., Mallet F., de Simone R., « Modeling Time(s) », *MoDELS*, vol. 4735 of *Lecture Notes in Computer Science*, Springer, p. 559-573, 2007.

Benveniste A., Caillaud B., Carloni L. P., Caspi P., Sangiovanni-Vincentelli A. L., « Heterogeneous reactive systems modeling : capturing causality and the correctness of loosely time-triggered architectures (LTTA) », *EMSOFT '04 : Proc. of the 4th Int. Conf. on Embedded software*, ACM, New York, NY, USA, p. 220-229, 2004.

Benveniste A., Caspi P., Edwards S., Hallbwachs N., Guernic P. L., de Simone R., « The synchronous languages Twelve years later », *Proceedings of the IEEE*, 2003.

Benveniste A., Guernic P. L., Jacquemot C., « Synchronous Programming with Events and Relations : the SIGNAL Language and Its Semantics », *Sci. Comput. Program.*, vol. 16, n° 2, p. 103-149, 1991.

Berry G., Sentovich E., « Multiclock Esterel », *in* T. Margaria, T. F. Melham (eds), *CHARME*, vol. 2144 of *LNCS*, Springer, p. 110-125, September, 2001.

Berthomieu B., Vernadat F., « Time Petri Nets Analysis with TINA », *QEST*, IEEE Computer Society, p. 123-124, September, 2006.

Brace K. S., Rudell R. L., Bryant R. E., « Efficient implementation of a BDD package », *DAC'90 : Proc. of the 27th Conf. on Design automation*, ACM, New York, NY, USA, p. 40-45, 1990.

Johansson R., Lönn H., Frey P., ATESST Timing Model, Technical report, ITEA, 2008. Deliverable D2.1.3.

Mallet F., André C., « On the Semantics of UML/MARTE Clock Constraints », *ISORC*, IEEE Computer Society, p. 305-312, March, 2009.

Maraninchi F., Bouhadiba T., 42 : Programmable Models of Computation for the Component-Based Virtual Prototyping of Heterogeneous Embedded Systems, Technical Report n° TR-2009-1, Verimag, Centre Équation, 38610 Gières, January, 2009.

Meinel C., Theobald T., *Algorithms and Data Structures in VLSI Design*, Springer-Verlag, Berlin, 2007.

Merlin P., A Study of the Recoverability of Computer Systems, PhD, University of California, Irvine, 1974.

OMG, *Unified Modeling Language, Superstructure*. November, 2007, Version 2.1.2 formal/2007-11-02.

OMG, *UML Profile for MARTE, beta 3*, Object Management Group. May, 2009, OMG document number : ptc/09-05-13.

Petri C., « Concurrency Theory », *Petri Nets : Central Models and their properties*, vol. 254 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 4-24, 1987.

Reisig W., *Petri nets : an introduction*, Monograph on Theoretical Computer Science, Springer-Verlag, Berlin, 1985.