From Multi-clocked Synchronous Processes to Latency-insensitive Modules*

Jean-Pierre Talpin Dimitru Potop-Butucaru Julien Ouy Benoit Caillaud
INRIA - IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
FirstName.LastName@inria.fr

ABSTRACT

We consider the problem of synthesizing correct-by-construction globally asynchronous, locally synchronous (GALS) implementations from modular synchronous specifications. This involves the synthesis of asynchronous wrappers that drive the synchronous clocks of the modules and perform input reading in such a fashion as to preserve, in a certain sense, the global properties of the system. Our approach is based on the theory of weakly endochronous systems, which gives criteria guaranteeing the existence of simple and efficient asynchronous wrappers. We focus on the transformation (by means of added signalling) of the synchronous modules of a multiclock synchronous specification into weakly endochronous modules, for which simple and efficient wrappers exist.

Categories and Subject Descriptors: D.2.2: CASE
General Terms: Algorithms, Reliability, Verification.
Keywords: separate compilation, compositional mapping.

1. INTRODUCTION

We start our presentation with a summary of the theory under consideration. We refer the reader to the continuation of the present work in [4] for motivating examples, throughout relation to previous works and applications to separate compilation and distributed code generation. The micro-step automata theory under consideration is a particular class of concurrent automata equipped with synchronous product and synchronous and asynchronous FIFO models allowing to represent computation and communication causality as well as communication through read/write primitives over given communication channels. A detailed description of this theory can be found in [3].

Micro-step automata communicate through signals $x \in X$. The *labels* $l \in L_X$ generated by the set of names X are represented by a partial map of *domain* from a set of signals X noted $\mathrm{vars}(l)$ to a set of values $V^\perp = V \cup \{\perp\}$ and tags. The label $\perp$ denotes the *absence* of communication during a transition of the automaton. We note $l' \leq l$ iff there exists l'' disjoint from l' such that $l = l' \cup l''$ and then $l \backslash l' = l''$. We say that l and l' (resp. t and t') are *compatible*, written $l \bowtie l'$, iff $l(x) = l'(x)$ for all $x \in \mathrm{vars}(l) \cap \mathrm{vars}(l')$ and, if so, note $l \cup l'$ their union. We write $\mathrm{supp}(l) = \{x \in X \,|\, l(x) \neq \perp\}$ for the *support* of a label l and $\perp_X$ for the empty support.

Definition 1. *An* automaton $A = (s^0, S, X, \rightarrow)$ *is defined by an initial state* s^0*, a finite set of states* S *noted* s *or* $x = v$*, labels* L_X *and by a transition relation* $\rightarrow$ *on* $S \times L_X \times S$*. The* product $A_1 \otimes A_2$ *of* $A_i = (s_i^0, S_i, X_i, \rightarrow_i)$ *for* $0 < i \leq 2$ *is defined by* $((s_1^0, s_2^0), S_1 \times S_2, X_1 \cup X_2, \rightarrow)$ *where* $(s_1, s_2) \rightarrow^l (s_1', s_2')$ *iff* $s_i \rightarrow^{l|_{X_i}} s_i'$ *for* $0 < i \leq 2$ *and* $l|_{X_i}$ *the projection of* l *on* X_i*. An automaton* $A = (s^0, S, X, \rightarrow)$ *is* concurrent *iff* $s \rightarrow^\perp s$ *for all* $s \in S$ *and if* $s \rightarrow^l s'$ *and* $l' \leq l$ *then there exists* $s'' \in S$ *such that* $s \rightarrow^{l'} s''$ *and* $s'' \rightarrow^{l \backslash l'} s'$*.*

Synchronous automata account for primitive communications using read and write operations on *directed communication channels* pairing variables x with directions represented by tags. Emitting a value v along a channel x is written $!x = v$ and receiving it $?x = v$. We write $\mathrm{vars}(D)$ for the channel names associated to a set of directed channels D. The undirected or untagged variables of a synchronous automaton are its *clocks* noted c.

Definition 2. *A* synchronous *automaton* $(s^0, S, X, c, \rightarrow)$ *of clock* $c \in X$ *is a concurrent automaton* $(s^0, S, X, \rightarrow)$ *s.t.*

1. $s \rightarrow^l s$ *implies* $l = c$ *or* $c \not\leq l$
2. $s^0 \rightarrow^c s^0$
3. $s \rightarrow^c s'$ *implies* $s' \rightarrow^c s'$
4. *if* $s_{i-1} \rightarrow^{l_i} s_i$ *and* $l_i \neq 1$ *for* $0 < i, j \leq n$ *then* $\mathrm{vars}(l_i) \cap \mathrm{vars}(l_j) = \emptyset$ *iff* $i \neq j$*.*

We assume that a channel x connects at most one emitter with at most one receiver. Multicast will however be used in examples and is modeled by substituting variable names (one $!x = v$ and two $?x = w_{1,2}$ will be substituted by two $!x = v$, $!x_2 = v$ and two $?x = w_1$ $?x_2 = w_2$ by introducing a local signal x_2). For an automaton A, we define a *trace* $t \in T = L_X^*$ by a finite sequence of labels, write $|t|$ for its length and refer to t_i as the ist label in t and $\mathcal{T}_A(s)$ as the set of traces accessible by A from state s. For a synchronous automaton of clock c, we write $s \rightarrow_t^* s'$ iff there exists a series $(s_i)_{0 \leq i \leq n}$ with $n = |t|$ such that $s_0 = s$, $s_{i-1} \rightarrow^{t_i} s_i$ for $0 < i \leq n$ and $s_n = s'$. We note $s_0 \twoheadrightarrow^t s$ iff $s \rightarrow_t^* s'$ with $t_i \neq c$ for $0 < i \leq |t|$ and $s_{|t|} \rightarrow^c s$.

*This work is partly funded by the ARTIST2 Network of Excellence and the Regional Council of Brittany

EMSOFT'05, September 19–22, 2005, Jersey City, New Jersey, USA.

The composition of automata is defined by synchronized product, using first-in-first-out buffer models to represent communication through synchronous and asynchronous channels. Synchronous communication is modeled using 1-place synchronous FIFO buffers. The synchronous FIFO of clock c and channel x is noted sFIFO_c^x and the asynchronous FIFO buffer of channel x is written aFIFO^x. A synchronous FIFO buffer serializes the emission event $!x = v$ followed by the receipt event $?x = v$ within the same transition (the clock tick c occurs after). Silent transitions $s_i \rightarrow^{\perp} s_i$ for $0 \leq i \leq 2$ are left implicit as sFIFO_c^x is assumed to be a concurrent automaton. The model of an unbounded and asynchronous FIFO buffer is similarly defined by the repetition of the communication pattern of the synchronous buffer and by induction on its storage size represented by a word $w \in V^*$.

$$\text{sFIFO}_c^x \stackrel{\text{def}}{=} \left(s_0, \{s_{0..2}\}, \{?x, !x, c\}, c, \; {}^{c}\!\circlearrowright s_0 \xrightarrow{!x=v} s_1 \xrightarrow{?x=v} s_2 \xrightarrow{c} s_0 \right)$$

$$\text{aFIFO}^x \stackrel{\text{def}}{=} \left(\begin{array}{l} \epsilon, V^*, \cup_{v\in V} \{?x = v, !x = v\}, \\ \quad \cup_{n\geq 0} \left\{ vw \xrightarrow{?x=v} w \xrightarrow{!x=v} wv \mid v \in V,\, w \in V^n \right\} \end{array} \right)$$

Two synchronous automata are *composable* if their tagged variables are mutually disjoint. This rule enforces the point-to-point communication restriction previously mentioned and non-overlapping of clocks. The synchronous composition of two automata consists of its synchronous product with the synchronous FIFO buffer model instantiated for all channels x common to the vocabulary of the composed automata. The clocks $c_{1,2}$ of both automata are synchronized to the clock c of the FIFO (by the substitution $A_i[c/c_i]$ of c_i by c in A_i). Similarly, the asynchronous composition consists of the synchronous product of the composed automata with instances of asynchronous FIFO buffers for all common channels x and, this time, without clock synchronization.

Definition 3. *Let $A_i = (s_i^0, S_i, X_i, c_i, \rightarrow_i)_{i=1,2}$ be two composable synchronous automata and c a clock and write $A[c_2/c_1]$ for the substitution of c_1 by c_2 in A. Synchronous composition $A_1 \,|^c A_2$ at clock c and asynchronous composition $A_1 \parallel A_2$ are defined by:*

$$A_1 \,|^c A_2 = \left(\bigotimes_{i=1,2} A_i[c/c_i] \right) \otimes \left(\bigotimes_{x \in \cap_{i=1,2} \text{vars}(X_i)} \text{sFIFO}_c^x \right)$$
$$A_1 \parallel A_2 = \left(\bigotimes_{i=1,2} A_i \right) \otimes \left(\bigotimes_{x \in \cap_{i=1,2} \text{vars}(X_i)} \text{aFIFO}^x \right)$$

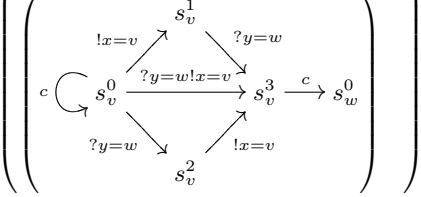
If $A = (s, S, X, c, T)$ then the restriction A/x is defined by $(s, S, X/\{x\}, c, \{s_1 \rightarrow^{l/x} s_2 \in T' \mid s_1 \rightarrow^l s_2 \in T\}$.

2. MICRO-STEP SEMANTICS OF SIGNAL

Micro-step automata provide an expressive operational semantics framework for the multi-clocked data-flow specification formalism Signal under consideration. In Signal [2], a process p is an infinite loop that consists of the synchronous composition $p \,|\, q$ of simultaneous equations $x = y\, f\, z$ over signals noted x, y, z. Restricting the lexical scope of a signal name x to a process p is noted p/x. A network of synchronous processes is noted P and $P \parallel Q$ stands for the asynchronous composition of P and Q.

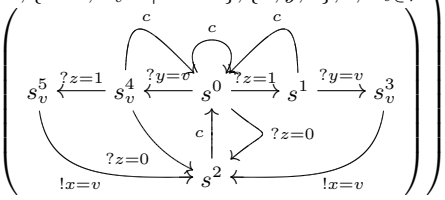
$$p, q \;::=\; (x = y\, f\, z) \mid p \,|\, q \mid p/x \qquad P, Q \;::=\; p \mid P \parallel Q$$

A delay equation $p \stackrel{\text{def}}{=} (x = y \,\text{pre}\, v_0)$ corresponds to an automata composed of four micro-states $s_v^{0..3}$ and for each value v of the signal x and starting from an initial state $s_{v_0}^0$ with the initial value v_0. The automaton concurrently performs both receive $?y = w$ and send $!x = v$ actions and then issues a clock transition c to the state s_w^0 where the next reaction takes place.

$$A_p = \left(s_{v_0}^0, \{s_v^{0..3}, \mid v \in V\}, \{x, y\}, c, \cup_{v,w\in V} \right.$$


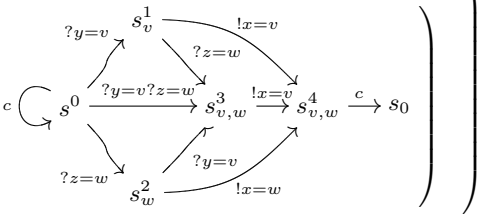
$$\left. \right)$$

A sampling equation $p \stackrel{\text{def}}{=} (x = y \,\text{when}\, z)$ starts from its initial state s^0 and concurrently performs either of two receive actions $?y = v$ and $?z = w$. If either y or z is absent or if z equals 0 then the automaton performs a clock transition to the initial state s^0. Otherwise, it performs the send action $!x = v$. This means that x is causal to both y and z. Notice that all intermediate states are parameterized with the value v under consideration.

$$A_p = \left(s^0, \{s^{0..2}, s_v^{3..5} \mid v \in V\}, \{x, y, z\}, c, \cup_{v\in V} \right.$$


$$\left. \right)$$

A merge equation $p \stackrel{\text{def}}{=} (x = y \,\text{default}\, z)$ performs two concurrent receive actions $?y = v$ and $?z = w$. If y is present then the send action is always $!x = v$ to s_4. If only z is present (in s^2) then the send action is $!x = w$ to s^4 before a clock transition back to s^0. Otherwise the only choice is a silent transition at s_0. Again, all intermediate states are parameterized with the possible pairs $(v, w) \in V^2$.

$$A_p = \left(s^0, \{s_v^1, s_w^2, s_{v,w}^{3..4} \mid v, w \in V\}, \{x, y, z\}, c, \cup_{v,w\in V} \right.$$


$$\left. \right)$$

Composition $p \,|\, q$ or $P \parallel Q$ and restriction p/x are defined by structural induction starting from the previous axioms and by using equivalent composition concepts of the model of micro-step automata.

$$A_{p\,|\,q} = A_p \,|^c A_q \quad A_{p/x} = (A_p)/x \quad A_{P\parallel Q} = A_P \parallel A_Q$$

3. FORMAL PROPERTIES

To address this issue, the property of weak endochrony [3] defines the class of deterministic micro-automata which satisfy insensitivity to latency.

Definition 4. *Let us write $s \twoheadrightarrow^t$ iff there exists s' such that $s \twoheadrightarrow^t s'$. A micro-automaton $A = (s^0, S, X, c, \rightarrow)$ is* weakly-endochronous *iff it is synchronous and:*

1. deterministic*: if $s \rightarrow^l s_1$ and $s \rightarrow^l s_2$ then $s_1 = s_2$*

2. step-independent: *if* $s \rightarrow^{l_1} s_1$, $s \rightarrow^{l_2} s_2$ *and* $\mathrm{supp}(l_1) \cap \mathrm{supp}(l_2) = \emptyset$ *then there exists* s' *such that* $s_1 \rightarrow^{l_2} s'$, $s_2 \rightarrow^{l_1} s'$ *and* $s \rightarrow^{l_1 \cup l_2} s'$
3. clock-independent: $s \rightarrow^{c} s'$ *implies*
   (a) *if* $s \xrightarrow{t}{}^{*}$ *and* $c \notin \mathrm{supp}(t)$ *then* $s' \xrightarrow{t}{}^{*}$
   (b) *if* $s \twoheadrightarrow^{t} s''$ *then* $s' \twoheadrightarrow^{t} s''$
   (c) *if* $s' \twoheadrightarrow^{tt''}$ *then* $s \twoheadrightarrow^{tt'}$ *and* $t' \leq t''$
   (d) *if* $s \twoheadrightarrow^{t}$ *and* $s \twoheadrightarrow^{t'}$ *and* $t \bowtie t'$ *then* $s \twoheadrightarrow^{t(t' \setminus t)}$
4. choice-independent: *if* $\mathrm{supp}(l_1) = \mathrm{supp}(l_2)$, $s \xrightarrow{t_1 l_1}{}^{*}$, $s \xrightarrow{t_2 l_2}{}^{*}$ *and* $t_1 \bowtie t_2$ *then* $s \xrightarrow{t_1 l_2}{}^{*}$ *and* $s \xrightarrow{t_1 l_2}{}^{*}$

Let $(A_i)_{0<i\leq n}$ be weakly endochronous automata, if the synchronous composition $\|^c A_i, 0 < i \leq n$ is non-blocking (i.e. $s \rightarrow^{l} s' \Rightarrow s \twoheadrightarrow^{lt}$) then the $(A_i)_{0<i\leq n}$ are weakly isochronous. Weakly isochronous automata satisfy the desynchronization correctness criterion of [3].

Theorem 1. *If the automata* $A_i, 0 < i \leq n$ *are weakly isochronous then their desynchronization is correct.*

4. FLOW ANALYSIS

In order to define decision criteria, section 5, to validate definition 4 and meet the property of theorem 1, we consider an intermediate representation of multi-clocked specification that exposes its control and data-flow properties for the purpose of their analysis.

A process p is represented as a data-flow graph G. In this graph, a vertex g is a data-flow relation that partially defined a clock or a signal. A signal vertex $c \Rightarrow x = f(y_{1..n})$ partially defines x by $f(y_{1..n})$ at the clock c. We note $\hat{g}$ the clock c of a signal vertex g, $\mathrm{use}(g)$ its set of used signal names $\{y_{1..n}\}$, $\mathrm{def}(g)$ its defined signal name x, $\mathrm{vars}(g) = \mathrm{use}(g) \cup \mathrm{def}(g)$ and $\mathrm{fun}(g)$ its function f, which can either be the identity, a boolean function ($\wedge, \vee$ or $\neg$) or the delay operator pre.

$$G, H ::= g \mid (G \,|\, H) \mid G/x \quad g, h ::= \hat{x} = e \mid c \Rightarrow x = f(y_{1..n})$$

A clock vertex $\hat{x} = e$ defines a relation between two clocks. A clock c defines a condition upon which a data-flow relation can be executed. It expresses control. The clock $\hat{x}$ defines when signal x is present (its value is available). Clocks x and $\neg x$ mean that x is true and false, respectively, and hence present. A clock expression e is Boolean expression that defines the way a clock is computed. 0 means never.

$$c ::= \hat{x} \mid x \mid \neg x \qquad e ::= 0 \mid c \mid e_1 \vee e_2 \mid e_1 \wedge e_2$$

The decomposition of a process into the synchronous composition of clock and signal vertices is defined by induction on the structure of p. Each equation is decomposed into data-flow functions guarded by a condition, the clock $\hat{x}$ of the output. This clock will need to be computed for the function to be executed. Notice the particular decomposition of a merge equation. The partial equations $x = y$ and $x = z$ are differentiated by the true and false value of a boolean signal δ, which we call a *differential clock*. A subtlety is that the sub-clock $\neg\delta$ is not locally defined while $\hat{\delta}$ and δ are. It is non-constructively defined by the difference between $\hat{z}$ and $\hat{y}$.

$$\begin{aligned}
G_{[x=y \text{ pre } v]} &= (\hat{x} \Rightarrow x = \mathsf{pre}(y, v)) \,|\, (\hat{x} = \hat{y}) \\
G_{[x=y \text{ when } z]} &= (\hat{x} \Rightarrow x = y) \,|\, (\hat{x} = \hat{y} \wedge z) \\
G_{[x=y \text{ default } z]} &= (\ (\delta \Rightarrow x = y) \,|\, (\hat{\delta} = \hat{x}) \\
&\quad\ |\, (\neg\delta \Rightarrow x = z) \,|\, (\delta = \hat{y}) \\
&\quad\ |\, (\hat{x} = \hat{y} \vee \hat{z}))/\delta \\
G_{[p \,|\, q]} &= G_{[p]} \,|\, G_{[q]} \qquad G_{[p/x]} = G_{[p]}/x
\end{aligned}$$

For the sake of a clear separation of concerns, the form of the graph G_p of a process p is decomposed into its clock vertices C_p, that defines its control and timing model, and signal vertices S_p, that define its (untimed) data-flow. The set of bound signal names X_p of G_p is further extracted by commutativity and for any substitution $(G/x) \,|\, H = ((G[y/x]) \,|\, H)/y$ of x by $y \notin \mathrm{vars}(G) \cup \mathrm{vars}(H)$ in G to yield the decomposition of the graph G_p of a process p as $G_p =^{\mathrm{def}} (C_p \,|\, S_p)/X_p$.

The remainder requires a couple of notations to be defined: we write $G \models e = f$ iff the system of Boolean equations C_p in G implies that $e = f$ always holds. In addition, and for all process p and all boolean signal x in p, we assume that $C_p \models \hat{x} = x \vee \neg x$ and $C_p \models x \wedge \neg x = 0$. We note $c \leq d$ for syntactic clock inclusion: $x < \hat{x}$ and $\neg x < \hat{x}$ and $\hat{x} \leq \hat{x}$. We write $\mathrm{vars}(p)$ for the set of free signal names of a process p. The free signals of a process are those which appear in equations and whose scope is not bound by restriction. The free output signals $\mathrm{out}(p)$ of a process p are free signal names occurring on the left-hand side of equations. The free input signals $\mathrm{in}(p)$ of a process p are the remainder $\mathrm{vars}(p) \setminus \mathrm{out}(p)$. The state variables $\mathrm{state}(p)$ of a process p are bound signals $x \in X_p$ defined by a delay equation.

The flow analysis of graphs G_p comprises control-flow aspects whose aim is to determine signal clocks from the information available to the process p: its input signals interface and its clock relations C_p. To this end, the relation $c \prec C$ is defined between a clock c and the set of supposedly known clocks C is can be deduced from in order to express it by a Boolean function f that satisfies $C_p \models c = f(C)$.

Definition 5. *The clock* c *is computable from the input signals* $\mathrm{in}(p)$*, written* $c \prec C$*, iff*
- *if* $x \in \mathrm{in}(p)$ *and* $c \leq x$ *then* $c \prec \{c\}$
- *if* $x \in \mathrm{state}(p)$ *and* $\hat{x} \prec C$ *and* $c < \hat{x}$ *then* $c \prec \{c\}$
- *if* $c_1 \prec C$ *and* $C_p \models c_1 = c_2$*, then* $c_2 \prec C$
- *if* $c_1 \prec C_1$*,* $c_2 \prec C_2$ *and* $C_p \models d = c_1 \vee c_2$ *or* $d = c_1 \wedge c_2$ *then* $d \prec C_1 \cup C_2$

The clocks of p *are computable iff for all bound signals* $x \in X_p$ *of* p *and all* $c \leq x$ *there exists* C *such that* $c \prec C$.

Data-flow analysis relates the vertices of a graph G by a scheduling relation $g \gg h$. It is defined iff the name defined by g can eventually be used by h. Two non-causal vertices are said independent, noted $g \lesseqgtr h$, and then they are either exclusive, written $g \# h$, synchronous, written $g \sim h$, or concurrent, written $g \diamond h$.

Definition 6. *Two signal vertices* $g, h \in S_p$ *are* causal*, written* $g \gg h$*, iff* $\mathrm{def}(g) \in \mathrm{use}(h)$*,* $\mathrm{fun}(h) \neq \mathsf{pre}$ *and* $C_p \models \hat{g} \wedge \hat{h} \neq 0$*;* independent*, written* $g \lesseqgtr h$*, iff* $g \not\gg h$ *and* $h \not\gg g$*. Two independent vertices* g *and* h *are* exclusive*, written* $g \# h$*, iff* $C_p \models \hat{g} \wedge \hat{h} = 0$*;* synchronous*, written* $g \sim h$*, iff* $C_p \models \hat{g} = \hat{h}$*. Two independent vertices* g *and* h *are either* connected*, written* $g \tilde{\#} h$*, iff* $g \# h$ *or* $g \sim h$*;* concurrent*, written* $g \diamond h$*, iff* $\neg g \tilde{\#} h$.

Definition 6 provides a complete structure for the vertices of a data-flow graph: two vertices are either causal or independent. Two independent vertices are either concurrent, synchronous or exclusive.

A notion of grammar establishes the duality between the automaton and the graph of a process p by linking these representations. The grammar M_p of a process p consists of signal vertices related by the connectors $\gg$, $\#$ $\sim$ and $\diamond$. It

can be viewed as a refinement of data-flow graph G_p which make the event structure implied by the clock relations C_p explicit.

$$M, N ::= g \mid M \gg N \mid M \# N \mid M \sim N \mid M \diamond N$$

To construct the event grammar of a process, we make use of a few functions to manipulate the graph structure implied by the relation of causality. The immediate successors and predecessors of a vertex g in a graph G are noted $\text{succ}_g(G)$ and $\text{pred}_g(G)$ and their transitive closures $\text{succ}^*_g(G)$ and $\text{pred}^*_g(G)$, respectively. The minimal and maximal vertices of a graph g are noted $\min(G)$ and $\max(G)$. The neighbors of g in G are $\text{next}_g(G) = \cup_{h \in \text{pred}_g(G)} \text{succ}_h(G)$.

$$\begin{array}{ll} \text{pred}_g(G) = \{h \in G \mid h \gg g\} & \min(G) = \{g \in G \mid \forall h \in G,\ h \not\gg g\} \\ \text{succ}_g(G) = \{h \in G \mid g \gg h\} & \max(G) = \{g \in G \mid \forall h \in G,\ g \not\gg h\} \end{array}$$

Definition 7. *The* grammar M *is defined from the signal vertices of a graph S by the function* $\text{fork}(S)$. *We write* $\text{fork}_g(S)$ *for the prefix of g in the grammar* $\text{fork}(S)$.

$$\begin{array}{rl} \text{fork}(S) = & \text{let } \diamond_{i=1}^{m}(\tilde{\#}_{j=1}^{n_i} g_{i,j}) = (\max(S))_{\tilde{\#}} \\ & \text{in } \diamond_{i=1}^{m}\left(\tilde{\#}_{j=1}^{n_i}\left(\text{fork}\left(\text{pred}^*(g_{i,j})\right)\right) \gg g_{i,j}\right) \end{array}$$

The partition $S_{\tilde{\#}} = (S_i)_{i=1}^n$ of a set of independent signal vertices S into exclusive vertices is defined by $S = \uplus_{i=1}^n S_i$ and, for all $0 < i, j \leq n$, for all $(g, b) \in S_i \times S_j$, $(g \tilde{\#} h \Leftrightarrow i = j)$ and $(g \diamond h \Leftrightarrow i \neq j)$.

A sequential component of M is a sub-grammar that does not contain a concurrency or synchrony relation (only $\gg$ or $\#$). A thread T is a sequence of signal vertices that represents the possible scheduling of a transition. We call $|T|$ its length and T_i its ist element. The threads $\text{join}(M)$ of a grammar M are constructed by structural induction.

$$\begin{array}{rl} \text{join}(g) = g & \text{join}(M \# N) = \text{join}(M) \cup \text{join}(N) \\ \text{join}(M \gg N) = \{TU \mid (T, U) \in \text{join}(M) \times \text{join}(N)\} & \\ \text{join}(M \sim N) = \text{join}((M \gg N) \# (N \gg M)) & \\ \text{join}(M \diamond N) = \text{join}((M \sim N) \# M \# N) & \end{array}$$

5. DECISION PROCEDURES

The control and data-flow analysis of a process define an event structure represented by a grammar M_p that allows us criteria upon which a process p can be guaranteed to be weakly-endochronous. First, we show that checking a process p deterministic and step independent amounts to the satisfaction of clock relations in C_p, some of which being related to the causal structure implied by its graph G_p.

Definition 8. *Let (g, h) two signal vertices of a graph G_p. If $g \gg h$ then $(\hat{g} \wedge \hat{h}) \Rightarrow \text{def}(g) \gg^* \text{def}(h)$. If $e_1 \Rightarrow x \gg^* y$ and $e_2 \Rightarrow x \gg^* y$ then $(e_1 \vee e_2) \Rightarrow x \gg^* y$. If $e_1 \Rightarrow x \gg^* y$ and $e_2 \Rightarrow y \gg^* z$ then $(e_1 \wedge e_2) \Rightarrow x \gg^* z$. A process p is* schedulable *iff $e \Rightarrow x \gg^* x$ implies $C_p \models e = 0$ for all x.*

A process p is predictable *iff its clocks are computable and all distinct vertices $g \neq h$ of S_p such that $\text{def}(g) = \text{def}(h)$ are exclusive $g \# h$.*

Second, grammars provide the necessary framework to finitely explore the state-space of a process and define decision procedures for the properties of clock and choice-independence.

Definition 9. *Thread T is compatible with U on I, written $T \triangleright_I U$, iff for all $x \in I$ and $0 < i \leq |T|$, there exists $0 < j \leq |U|$ s.t. $\hat{T}_i \Rightarrow \hat{x}$, $\hat{U}_j \Rightarrow \hat{x}$ and $\hat{T}_i \wedge \hat{U}_j \neq 0$.*

Two threads T and U are compatible *on I, written $T \bowtie_I U$, iff $T \triangleright_I U$ and $U \triangleright_I T$. Two grammars are compatible, written $M \bowtie_I N$, iff $T \bowtie_I U$ for all $(T, U) \in \text{join}(M) \times \text{join}(N)$. Two processes p and q are compatible $I = \text{vars}(p) \cap \text{vars}(q)$ iff $M_p \bowtie_I M_q$.*

We write $S|_I$ for the restriction of S on vertices that have used or defined variables in I.

Definition 10. *A process p is* conflit-free *iff, for all $g, h \in S = (S_p)|_{\text{vars}(p)}$, (1) if $g \diamond h$ then for all $(g', h') \in \text{pred}_g(S) \times \text{pred}_h(S)$, $g' \neq h'$, $\text{use}(g') \cap \text{use}(h') \cap \text{vars}(p) = \emptyset$ and $g' \lesseqgtr h'$ and (2) if $g \# h$ and $\text{fork}_g(S) \bowtie_{\text{vars}(p)} \text{fork}_h(S)$ then there exists $(g', h') \in \text{next}_g(S) \times \text{next}_h(S)$, $g \sim h'$ and $h \sim g'$.*

A conflict-free process p is clock and choice independent: for instance, the grammar of the switch restricted to its free variables is conflict-free: $(g_1^{x_1} \# g_2^{x_1}) \diamond (g_1^{x_2} \# g_2^{x_2})$. At last, Definition 11 summarizes the above and defines a notion of weak controllability that is sufficient for a process to be weakly endochronous and a pair of processes weakly isochronous.

Definition 11. *A process p is* weakly controllable *iff it is schedulable, predictible and conflict-free. Two processes p and q are* mutually controllable *iff p and q are compatible and $p \mid q$ is schedulable.*

Our main result is in accordance to this definition.

Theorem 2. *If p is weakly controllable then A_p is weakly endochronous. If p and q are weakly and mutually controllable then A_p and A_q are weakly isochronous.*

6. CONCLUSIONS

The micro-step automata theory of [3], and with the data-structure and analysis we introduce, allow us to attack a central issue of desynchronization: to locally and compositionally minimize synchronization constraints. It preludes optimized code generation and communication synthesis schemes to be accordingly defined. In this aim, a subsequent technical report [4] defines a procedure for compositionally synthesizing a weakly-endochronous automaton and modularly generating C-like code starting from the intermediate representation of a weakly controllable process p.

7. REFERENCES

[1] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The Synchronous Languages Twelve Years Later. *Proceedings of the IEEE*. IEEE Press, 2003.

[2] P. Le Guernic, J.-P. Talpin, and J.-C. Le Lann. Polychrony for system design. *Journal of Circuits, Systems and Computers*. World Scientific, 2003.

[3] D. Potop-Butucaru and B. Caillaud. Correct-by-construction asynchronous implementation of modular synchronous specification. In *Application of Concurrency to System Design*. IEEE Press, 2005.

[4] J.-P. Talpin, D. Potop-Butucaru, J. Ouy, B. Caillaud. Compositional synthesis of latency-insensitive systems from multi-clocked synchronous specifications. Research Report n. 1730. IRISA, June 2005.