Synchronous Structures

David Nowak, Jean-Pierre Talpin, and Paul Le Guernic

INRIA-Rennes – IRISA, Campus de Beaulieu, F-35042 Rennes Cédex

Abstract. Synchronous languages have been designed to ease the development of reactive systems, by providing a methodological framework for assisting system designers from the early stages of requirement specifications to the final stages of code generation or circuit production. Synchronous languages enable a very high-level specification and an extremely modular design of complex reactive systems. We define an order-theoretical model that gives a unified mathematical formalization of all the above aspects of the synchronous methodology (from relations to circuits). The model has been specified and validated using a theorem prover as part of the certified, reference compiler of a synchronous programming language.

1 Introduction

Synchronous languages, such as SIGNAL [2], LUSTRE [9] and ESTEREL [4] have been designed to ease the development of reactive systems. The synchronous hypothesis provides a deterministic notion of concurrency where operations and communications are instantaneous. In a synchronous language, concurrency is meant as a logical way to decompose the description of a system into a set of elementary communicating processes. Interaction between concurrent components is conceptually performed by broadcasting events. Synchronous languages enable a very high-level specification and an extremely modular design of complex reactive systems by structurally decomposing them into elementary processes. The use of synchronous languages provides a methodological framework for assisting the users from the early stages of requirement specifications to the final stages of code generation or circuit production while obeying compliance to expressed and implied safety requirements. In that context, the synchronous language SIGNAL is particularly interesting, in that it allows the specification of (early) relational properties of systems which can then be progressively refined in order to obtain an executable specification. All the stages of this design process can easily be modeled and understood in isolation. The purpose of our presentation is to define a mathematical model which gives a unified formalization of all the aspects of a synchronous methodology and which contains each of them in isolation. The model uses basic notions of set-theory and order-theory. It

has been specified and validated using the Coq proof assistant [7]. This implementation is part of a certified, reference compiler of the Signal language. It completes and extends the results of [12] on the definition of a co-inductive trace semantics of Signal in Coq.

Influential Analogy. In 1545, the great Italian mathematician Gerolamo Cardano wrote an important and influential treatise on Algebra: "Ars Magna" [5] in which the first complete expression for the solution of a general cubic equation was put forward. Cardano noticed that, in the case of some equation with three real solutions, he was forced to take at a certain stage the square root of a negative number. The imaginary numbers were borned. Analogically, we generalize the classical notion of signal ([2, 3, 10]) with imaginary signals. This extension has no material counterpart. It is used to compute intermediate results. For instance, the temporal abstractions of signals (called clocks) have necessary a greatest lower bound but do not always have a (real) least upper bound. In that case, we need to define an imaginary least upper bound. This axiomatization allows to extend the notion of classical clocks (a clock is a temporal abstraction of a signal) with imaginary clocks and define a boolean lattice of clocks. In this lattice-theoretical model, temporal relations between signals always have a solution. If the solution contains imaginary signals, this means that the system has no real solution in the classical model and that it does not thus form an executable specification.

Plan We first introduce the synchronous language Signal in the section 2. In the section 3, we abstract the notion of control dependence in a mathematical structure that we call a synchronous structure. Within this structure we formalize the notions of signals, clocks and instants, and their relations. We define some internal operations on signals and clocks, prove their algebraic properties, prove that the set of clocks forms a boolean lattice, and define a Cartesian closed category of signals with product and coproduct. In the section 4, we add a valuation function and a data dependency relation to synchronous structure. In the section 5, we briefly expose the outcome of our model for the compilation of programs written in the synchronous language Signal.

2 Overview of Signal

Signal is an equational synchronous programming language. A Signal program is modularly organized into processes consisting of simultaneous equations on signals. In Signal, an equation is an elementary and instantaneous operation on input signals which defines an output. A signal

is a sequence of values defined over a totally ordered set of instants. At any given instant, a signal x is either present or absent.

$P ::=$	$(P \parallel P')$	parallel composition
$\mid$	$P/\mathtt{x}$	restriction
$\mid$	$R(x1, \cdots, xn)$	instantaneous relation
$\mid$	$z := x$ when y	selection
$\mid$	$z := x$ default y	deterministic merge
$\mid$	$y := x\$$ init v	delay

In SIGNAL, a process P is either an equation or the synchronous composition $P \parallel P'$ of processes. Parallel composition $P \parallel P'$ synchronizes the events produced by P and P'. P/x masks the signal x in the process P i.e. x is a local signal of the process P. An instantaneous relation $R(x_1, \cdots, x_n)$ forces the signals $x_1, \cdots, x_n$ to be synchronous and theirs instantaneous values to satisfy the relation R. A delay $y := x\$$ init v (called "shift register" in [2]) stores the value v' of x and outputs the previous value v of x to y. A deterministic merge $z := x$ default y outputs the value of x to z (if x is present) or the value of y (if x is absent). A selection (or down-sampling) $z := x$ when y outputs x to z when y is present and true. When all the inputs of an equation are absent, a transition takes place but no value is given to its output.

A set of equations can be encapsulated as a new reusable process. It consists of an interface providing parameters, input and output signals with their types. The pervasive operators when, default and \$ of SIGNAL offer a flexible mean for progressively specifying reactive systems, from the early specification of system properties or requirements to late executable programs. To illustrate this process, let us consider the design of a simple replenishable tank where *capacity* is an integer parameter, $fill$ is an input signal of type event (a subtybe of boolean onlay inhabited by $true$), and *empty* is an output signal of type boolean.

process $tank$ = {integer *capacity*} (? event $fill$! boolean *empty*)
(| synchro (when $(zn = 0), fill$)
| $zn := n\$$ init 0
| $n :=$ (*capacity* when $fill$) default $(zn - 1)$
| *empty* := when $(n = 0)$ default (not $fill$) |) / n, zn

This program uses an extended and more intuitive syntax of SIGNAL [11] that can be translated into the SIGNAL-kernel described in this overview. synchro is a process that forces its input signals to be synchronous. The following table illustrates an execution of the process *tank*

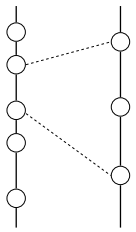
with a capacity equal to 3.

$fill$	t				t				t	
zn	0	3	2	1	0	3	2	1	0	3
n	3	2	1	0	3	2	1	0	3	2
$empty$	f			t	f			t	f	

It is easy to observe that without its first equation synchro {when $(zn = 0), fill$} the process *tank* would not form an executable specification. In this case, it would be impossible to relate the clock of the signal $fill$ with the other clocks of the program. But it would be a correct sub-specification that could be composed with another specification to remove the non-determinism. In this paper, we show how to deal with non-determinism using imaginary signals.

3 Control Dependence

In this section, we focus on a characterization of control dependencies, i.e., the temporal relations between events or the dates of events relative to some reference of time, not the value of events. Let us informally depict a synchronization scenario between two sequences of events (i.e. sets of ordered events). They exchange (dotted) synchronization messages using an asynchronous medium for their communications. This involve a synchronization relation between events. The natural structure of time of the whole system is that of a partial pre-order. In this section, we will abstract the notions involved in this example.



3.1 Synchronous Structure

We define a synchronous structure as an ordered set (its elements are called *events*) with a particular equivalence relation $\sim$. Intuitively, $x \sim y$ means that x and y are synchronous, that is to say the events x and y must occur simultaneously. The order relation $\leq$ is the temporal causality between two events: $x < y$ means that x must occur strictly before y.

Definition 1. $(\mathcal{E}, \ll)$ *is a synchronous structure iff* $\mathcal{E}$ *is a non empty set (of events) and* $\ll$ *is a preorder on* $\mathcal{E}$ *such that:*

$$\forall x \in \mathcal{E}, \{y \in \mathcal{E} \mid y \leq x\} \text{ is finite, where } \begin{array}{l} x \sim y \Leftrightarrow_{def} x \ll y \wedge y \ll x \\ x < y \Leftrightarrow_{def} x \ll y \wedge x \not\sim y \\ x \leq y \Leftrightarrow_{def} x < y \vee x = y \end{array}$$

For instance, the left part of the figure 1 depicts eight events which define a synchronous structure. To give easier explanations, the events are numbered from 1 to 8. Dotted lines represent the equivalence relation $\sim$ and bold lines represent the strict order relation $<$ as a Hasse diagram: $x < y$ iff there is a sequence of connected bold line segments moving downwards from x to y.

The preorder $\ll$ mixes the synchronicity relation and the temporal causality relation. It defines a notion of time for the whole system. We will explain this structure in more details after introducing the notion of signal. The right part of the figure depicts the preorder relation $\ll$ between events as a Hasse diagram where synchronous events are grouped in one node. From the fact that $\leq$ is well founded, we deduce that $\ll$ is a well founded preorder.

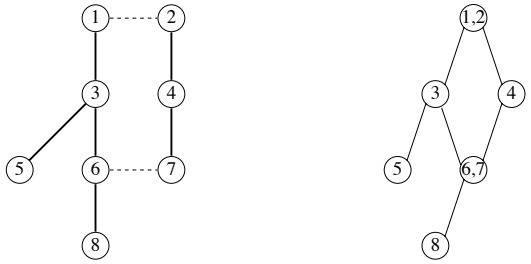


Fig. 1. Events and associated preorder

The following proposition comes directly from the definition of a synchronous structure. In the example, it guarantees that the events numbered 1 and 8 cannot be synchronous.

Proposition 1. $\forall (x, y_1, y_2, z) \in \mathcal{E}^4, x \ll y_1 \wedge y_1 < y_2 \wedge y_2 \ll z \Rightarrow \neg x \sim z$

We say that an event x is *covered by* an event y, and write $x \prec\!\!\!- y$, iff $x < y$ and there is no event z satisfying $x < z < y$. From the fact that $\leq$ is well founded, we can deduce the following proposition. This proposition is important to guarantee a discrete model of synchronous programming.

Proposition 2. $\forall (x,y) \in \mathcal{E}^2, x < y \Rightarrow \exists z \in \mathcal{E}, z \prec\!\!\!- y$

Indeed, $(\mathcal{E}, \leq)$ is not dense because $\leq$ is well founded.

3.2 Signal, Clock and Instant

In this subsection we define the objects of the model and their relations. First, we formalize the notion of *signal*. Usually, a (real) signal is a totally ordered set of events. This total order implies that two different events cannot be synchronous. We generalize this definition to enable partially ordered sets of events to be (imaginary) signals. A signal just have to satisfy the property that two different events cannot be synchronous. In the subsection 3.4, this relaxed condition is used to define internal operations.

Definition 2. *Let X be a subset of $\mathcal{E}$. X is a* signal *iff it satisfies the following axiom:*

$$\forall (x,y) \in X^2, x \sim y \Rightarrow x = y \tag{1}$$

Let $\mathcal{S}$ be the set of signals. For instance, in the figure 1, $\{1,3,5,8\}$ and $\{2,6,8\}$ are in $\mathcal{S}$. A *real signal* is then a particular case of signal which is totally preordered by $\ll$. For instance, in the figure 1, $\{1,3,5\}$, $\{2,6,8\}$ and $\emptyset$ are real signals but not $\{1,3,5,8\}$. An *imaginary signal* is a signal which is not a real signal. An imaginary signal enables to represent the lack of synchronization constraints in a sub-specified reactive system. In SIGNAL, a sub-specification is a correct specification that cannot be executed because of non-determinism. It needs to be composed with another specification to remove the non-determinism. Let X be a signal. From the axiom 1 we deduce that $\ll$ is antisymmetric on X and then is an order relation on X. X is totally ordered by $\ll$ iff X is a real signal. From the proposition 2, we deduce proposition 3. Then, we define a preorder relation $\preceq$ on $\mathcal{S}$ (definition 3, see, for instance, the figure 2). The preorder $\preceq$ gives rise to an equivalence relation $\widehat{=}$ (definition 4, we say that X and Y are *synchronous* iff $X \widehat{=} Y$).

Proposition 3. $\forall X \in \mathcal{S}, \forall (x,y) \in X^2, x < y \Rightarrow \exists z \in X, z \prec\!\!\!- y$

Definition 3. *For all signals X and Y, $X \preceq Y$ iff $\forall x \in X, \exists y \in Y, x \sim y$*

Definition 4. *For all signals X and Y, $X \widehat{=} Y$ iff $X \preceq Y$ and $Y \preceq X$.*

In order to study the temporal relations between signals, we define the equivalence classes of signals by $\widehat{=}$.

Definition 5. *The* set of clocks *$\mathcal{C}$ is the quotient of $\mathcal{S}$ by $\widehat{=}$.*

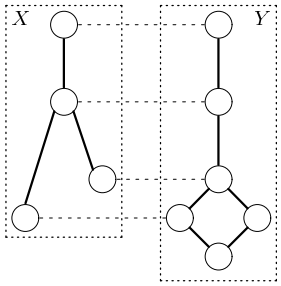


Fig. 2. Preordered signals ($X \preceq Y$)

For any signal X, we write $\widehat{X}$ its equivalence class that we call its *clock*. $\widehat{\emptyset}$ is called the *null clock*. The clock of a real (resp. imaginary) signal is a real (resp. imaginary) clock. The preorder $\preceq$ on $\mathcal{S}$ gives rise to an order $\sqsubseteq$ on $\mathcal{C}$.

Definition 6. *For all signals X and Y, $\widehat{X} \sqsubseteq \widehat{Y}$ iff $X \preceq Y$.*

We define the equivalence classes of events by $\sim$. Intuitively, these classes will represent the notion of logical instant.

Definition 7. *The* set of instants *$\mathcal{I}$ is the quotient of $\mathcal{E}$ by $\sim$.*

For any event x, we write $\widetilde{x}$ its equivalence class that we call its *instant*. The preorder $\ll$ gives rise to an order $\triangleright$ on $\mathcal{I}$.

Definition 8. *For all event x and y, $\widetilde{x} \triangleright \widetilde{y}$ iff $x \ll y$.*

Intuitively, it is clear that a clock should be related to a set of instants and conversely. We show that the set of clocks $\mathcal{C}$ and the powerset of $\mathcal{I}$ are isomorphic. Let $\mathcal{P}(\mathcal{I})$ be the powerset of $\mathcal{I}$. Using the Axiom of Choice, we prove the following theorem.

Theorem 1. *$(\mathcal{C}, \sqsubseteq)$ and $(\mathcal{P}(\mathcal{I}), \subseteq)$ are isomorphic.*

Let I be a set of instants. By definition of an instant, I is a set of disjoint sets of events. The Axiom of Choice is then necessary to "choose" a single event from each element of I. Then we can construct a signal and take its clock which is then the associated clock of I. Therefore there is a function f from $\mathcal{P}(\mathcal{I})$ to $\mathcal{C}$. We show that this function is invertible and f and f^{-1} are increasing. This is sufficient to prove that f is an isomorphism.

3.3 Trace

We can link this order-theoretic approach to our trace semantics of SIGNAL developped in [12]. Let $i \in \mathcal{I}$ be an instant. $t_X(i)$ is the event at the

intersection of X and i if it exists. Or else it is the special value $\perp \notin \mathcal{E}$ if the intersection is empty. t_X is called the *trace of* X.

$$t_X : \mathcal{I} \longrightarrow \mathcal{E} \cup \{\perp\}$$
$$i \longmapsto \begin{cases} x \text{ if } X \cap i = \{x\} \\ \perp \text{ else} \end{cases}$$

The following lemma guarantees that this definition is correct.

Lemma 1. *For any signal X, for any instant i, for any event x of X, $X \cap i = \emptyset \vee \exists x \in X, X \cap i = \{x\}$ and $X \cap \widetilde{x} = \{x\}$.*

The two approaches are linked by the logical property: $X = Y \Leftrightarrow t_X = t_Y$.

3.4 Operations on Signals and Clocks

In this subsection we define some operations on signals and clocks which denote the control part of the instructions of SIGNAL [2]. Let X and Y be signals. First we define the selection of a signal at the clock of another signal.

Definition 9 (Selection). *For all signals X and Y,*

$$X \otimes Y =_{def} \{x \in X | \exists y \in Y, x \sim y\}$$

$\otimes$ is an internal operation on $\mathcal{S}$ i.e. for all signals X and Y, $X \otimes Y$ is a signal. For instance, in the left part of the figure 3, the selection of the signal Y at the clock of X is depicted. Although X and Y are imaginary signals, the result $Y \otimes X$ is a real signal in this example. The operator $\otimes$ on $\mathcal{S}$ gives rise to the greatest lower bound operator $\sqcap$ on $\mathcal{C}$.

Definition 10. *For all signals X and Y, $\widehat{X} \sqcap \widehat{Y} =_{def} \widehat{X \otimes Y}$.*

Proposition 4. *$C \sqcap D$ is the greatest lower bound of clocks C and D.*

We define the merge of two signals with priority to the left event.

Definition 11 (Deterministic Merge). *For all signals X and Y,*

$$X \oplus Y =_{def} X \cup \{y \in Y | \neg \exists x \in X, x \sim y\}$$

$\oplus$ is an internal operation on $\mathcal{S}$ i.e. for all signals X and Y, $X \oplus Y$ is a signal. For instance, in the right part of the figure 3, the deterministic merge of the signals X and Y is depicted. Although X and Y are real signals, their deterministic merge $X \oplus Y$ is an imaginary signal because its events are not totally preordered by $\ll$. The operator $\oplus$ on $\mathcal{S}$ gives rise to the least upper bound operator $\sqcup$ on $\mathcal{C}$.

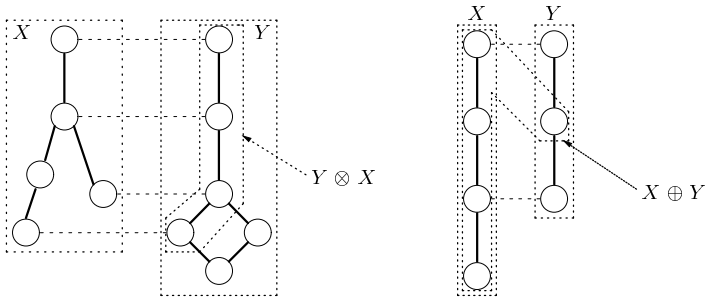


Fig. 3. Examples of selection and deterministic merge

Definition 12. *For all signals X and Y, $\widehat{X} \sqcup \widehat{Y} =_{def} \widehat{X \oplus Y}$.*

Proposition 5. *$C \sqcup D$ is the least upper bound of clocks C and D.*

Every couple of clocks $\{C, D\}$ has a least upper bound $C \sqcup D$ and a greatest lower bound $C \sqcap D$. Therefore $(\mathcal{C}, \sqcap, \sqcup)$ is a lattice[1]. From the isomorphism between $\mathcal{P}(\mathcal{I})$ and $\mathcal{C}$, we deduce that the lattice $(C, \sqcap, \sqcup)$ is boolean i.e. it is complete, distributive and there exists a null element $\widehat{\emptyset}$ and a universal element $\top$. Let f be the morphism from $\mathcal{P}(\mathcal{I})$ to $\mathcal{C}$. The universal element $\top$ is equal to $f(\mathcal{I})$. We define the operator $\backslash$ on clocks which is the counterpart of the operator $\backslash$ on sets of instants which subtracts a set from an other. Let f be the morphism from $\mathcal{C}$ to $\mathcal{P}(\mathcal{I})$: $C \backslash D =_{\text{def}} f^{-1}(f(C) \backslash f(D))$. The complementary of a signal X is a "chosen" signal $\overline{X}$ (using the Axiom of Choice) of clock $\top \backslash \widehat{X}$. Algebraic properties of these operations on signals and clocks are summarized in the figure 4. They are easily proved by case analysis using the trace semantics. We just have to translate the signal operators $\otimes$ and $\oplus$ into the trace semantics. We define an operator . on traces such that $t_{X \otimes Y} = t_X.t_Y$ and an operator $+$ on traces such that $t_{X \oplus Y} = t_X + t_Y$.

$$t_X.t_Y : \mathcal{I} \longrightarrow \mathcal{E} \cup \{\bot\} \qquad i \longmapsto \begin{cases} t_X(i) : t_X(i) \neq \bot, t_Y(i) \neq \bot \\ \bot \quad \text{othewise} \end{cases}$$

$$t_X + t_Y : \mathcal{I} \longrightarrow \mathcal{E} \cup \{\bot\} \qquad i \longmapsto \begin{cases} t_X(i) : t_X(i) \neq \bot \\ t_Y(i) \quad \text{othewise} \end{cases}$$

Note that $\oplus$ is not distributive to the right with respect to $\otimes$. Indeed, if $t_X(i) = x$, $t_Y(i) = \bot$ and $t_Z(i) = z$ then $((t_X.t_Y) + t_Z)(i) = z$ and $((t_X + t_Z).(t_Y + t_Z))(i) = x$.

[1] This is not true for real clocks as they do not always have a real least upper bounds

$$X \otimes Y \preceq X \qquad X \preceq X \oplus Y$$
$$X \otimes Y \preceq Y \qquad Y \preceq X \oplus Y$$
$$X \otimes Y \widehat{=} Y \otimes X \qquad X \oplus Y \widehat{=} Y \oplus X$$
$$X \otimes (Y \otimes Z) = (X \otimes Y) \otimes Z \qquad X \oplus (Y \oplus Z) = (X \oplus Y) \oplus Z$$
$$X \otimes (Y \oplus Z) = (X \otimes Y) \oplus (X \otimes Z) \qquad X \oplus (Y \otimes Z) = (X \oplus Y) \otimes (X \oplus Z)$$
$$(X \oplus Y) \otimes Z = (X \otimes Z) \oplus (Y \otimes Z)$$

Fig. 4. Algebraic properties of $\otimes$ and $\oplus$

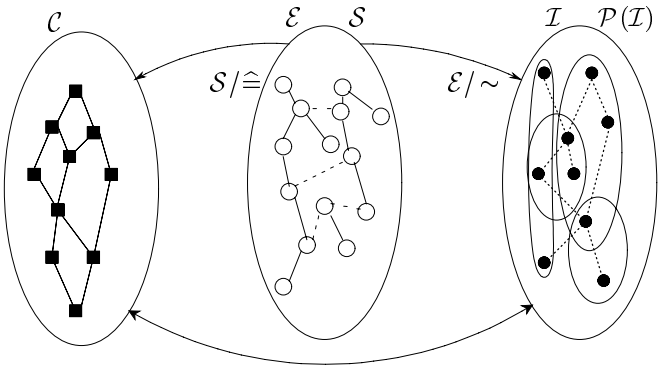


Fig. 5. Summary of the results presented so far

3.5 The Category of Signals

Another way to study temporal relations between signals is to define a category of signals in which a morphism describes the temporal relation between two signals. Suppose that X and Y are two signals such that $X \preceq Y$. Then, for any event $x \in X$, there exists an event $y \in Y$ such that $x \sim y$, by definition of $\preceq$. This event y is unique by definition of a signal. Hence, we can define a total function $[Y]_X$, called *signal morphism*, from X to Y:

$$[Y]_X : X \longrightarrow Y$$
$$x \longmapsto y \text{ such that } x \sim y$$

For all signals X and Y such that $X \preceq Y$,

1. $[Y]_X$ is injective: $\forall (x, x') \in X^2, [Y]_X(x) = [Y]_X(x') \Rightarrow x = x'$
2. $[Y]_X$ is strictly monotonic: $\forall (x, x') \in X^2, x < x' \Rightarrow [Y]_X(x) < [Y]_X(x')$
3. $[Y]_X$ is bijective (with $[Y]_X^{-1} = [X]_Y$) iff $X \widehat{=} Y$.

The identity $[X]_X$ is a signal morphism and signal morphisms can be composed: for all signals X, Y and Z such that $X \preceq Y \preceq Z$, $[Z]_X = [Z]_Y \circ [Y]_X$. The set of signals and the set of morphisms define a small (preorder) category Sig with product $\otimes$ and coproduct $\oplus$. More precisely,

let X and Y be two objects (i.e. signals) of the category Sig. The product object $X \otimes Y$ and the two projections $[X]_{X\otimes Y}$ and $[Y]_{X\otimes Y}$ are a product of X and Y. These data satisfy the property that, for any object Z and all morphisms $f : Z \longrightarrow X$ and $g : Z \longrightarrow Y$, there exists a unique morphism $\langle f, g\rangle : Z \longrightarrow X \otimes Y$ such that the left-diagram of figure 6 commutes. The coproduct object $X \oplus Y$ and the two injections $[X \oplus Y]_X$ and $[X \oplus Y]_Y$ are a coproduct of X and Y. These data satisfy the property that, for all object Z and all morphisms $f : X \longrightarrow Z$ and $g : Y \longrightarrow Z$, there exists a unique morphism $[f, g] : X \oplus Y \longrightarrow Z$ such that the right-diagram of figure 6 commutes.

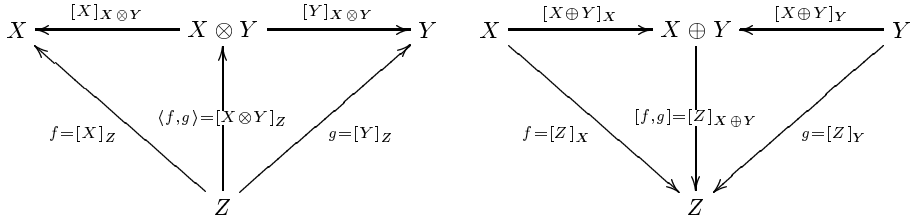


Fig. 6. Morphisms for product and coproduct objects

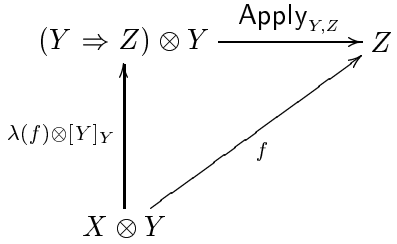
The signal $\emptyset$ is the unique initial object of the category Sig i.e. for any object X of Sig there exists a unique morphism $[X]_\emptyset : \emptyset \longrightarrow X$. And the coproduct $\oplus$ is defined for each ordered pair of objects of Sig. Hence the category Sig has finite coproducts. It is also possible to construct a terminal object. Let C be the clock corresponding to the set of all instants $\mathcal{I}$. Let $Y \in C$ be a signal of clock C. This signal Y is a terminal object i.e. for any object X of Sig there exists a unique morphism $[Y]_X : X \longrightarrow Y$. And the product $\otimes$ is defined for each ordered pair of objects of Sig. Hence the category Sig has finite products. Let $Y \Rightarrow Z$ be the object $\overline{Y} \oplus Z$ and $\mathsf{Apply}_{Y,Z} : (Y \Rightarrow Z) \otimes Y \longrightarrow Z$ be the morphism $[Z]_{(Y\Rightarrow Z)\otimes Y}$. $\mathsf{Apply}_{Y,Z}$ is correctly defined because $(Y \Rightarrow Z) \otimes Y \preceq Z$:

$$(Y \Rightarrow Z)\otimes Y = (\overline{Y}\oplus Z)\otimes Y = (\overline{Y}\otimes Y)\oplus(Z\otimes Y) = \emptyset\oplus(Z\otimes Y) = (Z\otimes Y) \preceq Z$$

In addition, $(Y \Rightarrow Z)\otimes Y = (Z\otimes Y)$. Therefore $\mathsf{Apply}_{Y,Z} = [Z]_{Z\otimes Y}$. Sig is Cartesian closed i.e. for all objects Z and each morphism $f : X\otimes Y \longrightarrow Z$ there exists a unique morphism $\lambda(f) = [Y \Rightarrow Z]_X : X \longrightarrow (Y \Rightarrow Z)$ such that the following diagram[2] commutes. The proof consists in proving that $X \preceq (Y \Rightarrow Z)$ i.e. $\lambda(f) = [Y \Rightarrow Z]_X$ is correctly defined. We

[2] $\forall f : X \longrightarrow Y, f : X' \longrightarrow Y', f \otimes g =_{\text{def}}: X \otimes X' \longrightarrow Y \otimes Y' = \langle f \circ [X]_{X\otimes X'}, g \circ [X']_{X\otimes X'}\rangle$

conjecture that the category Sig can be related to the category of event strutures ([15]) through functors.



4 Data Dependence

In this section, we complete our notion of partial ordered time to deal with data dependence.

4.1 Valuated Synchronous Structure

We associate a valuation function v and a data dependency relation $\rightarrow$ to synchronous structure.

Definition 13. *Let $\mathcal{D}$ be a set.* $(\mathcal{E}, \ll, v : \mathcal{E} \longrightarrow \mathcal{D}, \rightarrow)$ *is a* valuated synchronous structure *iff* $(\mathcal{E}, \ll)$ *is a synchronous structure, v a function from $\mathcal{E}$ to $\mathcal{D}$ and $\rightarrow$ is a partial order included in $\ll$ i.e. :*

$$\forall (x, y) \in \mathcal{E}^2, x \rightarrow y \Rightarrow x \ll y \tag{2}$$

The definition 8 of the partial order $\triangleright$ on instants and the axiom 2 guarantee that the value of an event cannot depend on the value of a future event. The data dependencies of an event come only from past or present values of other events. A signal is said *of domain* $D \subseteq \mathcal{D}$ iff all its events x are such that $v(x) \in D$. Let $\ll_{\mathrm{v}}$ be the transitive closure of the union of the relations $\ll$ and $\rightarrow$ The preorder $\ll_{\mathrm{v}}$ defines a notion of time which takes into account the synchronicity relation, control dependencies and data dependencies. We define a preorder relation $\preceq_{\mathrm{v}}$ on $\mathcal{W}$.

Definition 14. $X \preceq_v Y$ *iff* $X \preceq Y \wedge \forall x \in X, v(x) = v([Y]_X(x))$

The preorder $\preceq_{\mathrm{v}}$ gives rise to an equivalence relation $\widehat{=}_{\mathrm{v}}$. Intuitively, $X \widehat{=}_{\mathrm{v}} Y$ means that X and Y are synchronous and provide same values in same order.

Definition 15. $X \widehat{=}_v Y$ *iff* $X \preceq Y \wedge Y \preceq X$

4.2 Scheduling Specification

We define a ternary relation, called *conditional dependency*. Intuitively, $X \xrightarrow{C} Y$ states that, at the instants of the clock C, there are dependencies $\rightarrow$ from an event of X to an event of Y in the same instant. In this relation, we are only interested in instantaneous dependencies. Practically

this relation is used to schedule the computation that have to be done in the same logical instant. A set of conditional dependencies is called a *scheduling specification.*

Definition 16. $X \stackrel{\widehat{Z}}{\rightarrow} Y \Leftrightarrow_{def} \forall x \in X \otimes Z, \exists y \in Y, x \sim y \wedge x \rightarrow y$

The following theorem enables to compute the transitive closure of a scheduling specification.

Theorem 2. *For all signals X, Y and Z, for all clocks C and D,*

$$X \stackrel{C}{\rightarrow} Y \wedge Y \stackrel{D}{\rightarrow} Z \Rightarrow X \stackrel{C \sqcap D}{\longrightarrow} Z \qquad X \stackrel{C}{\rightarrow} Y \wedge X \stackrel{D}{\rightarrow} Y \Rightarrow X \stackrel{C \sqcup D}{\longrightarrow} Y$$

In the figure 7, the diagram on the left depicts a scheduling specification involving local variables. These are hidden in the diagram on the right, using the theorem 2.

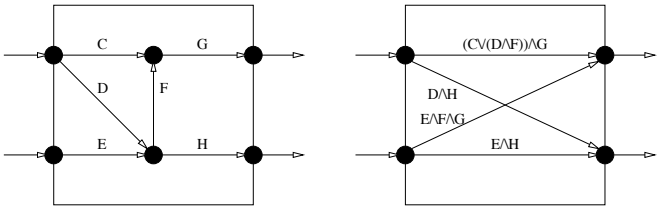


Fig. 7. Abstraction of Scheduling Specifications

5 Compilation of SIGNAL

First, we define the functions ${}^t.$ and $[.]$ which compute the sub-signal of the true valued events of a signal of domain $\{false, true\}$, and its clock.

Definition 17. *Let X be a signal of domain $\{false, true\}$.*

$${}^tX = \{x \in X | v(x) = true\} \qquad [X] = \widehat{{}^tX}$$

The delay enables to move forward the valuation of a real signal[3]. The value of an event of a delayed real signal is the value of the previous event if it exists. In the other case, a default value is given. $\mathsf{Pre}(u, X, Y)$ states that Y is the delayed signal of X initialized with u.

$$\mathsf{Pre}(u, X, Y) \Leftrightarrow_{\text{def}} X \widehat{=} Y \wedge \forall y \in Y, \begin{cases} y \text{ minimal element of } Y \Rightarrow v(y) = u \\ \exists y^- \in Y, y^- \prec y \Rightarrow v(y) = v([X]_Y(y^-)) \end{cases}$$

[3] It would make no sense to apply delay to imaginary signal.

We define a predicate that constrains a set of signals to be synchronous and to satisfy a predicate between their values at every instant. In SIGNAL, it is called an *instantaneous relation*.

Definition 18 (Instantaneous Relation). *Let $X_1, \cdots, X_n$ be n signals and P be a predicate on $\mathcal{D}^n$.*

$$R^n_P(X_1, \cdots, X_n) \Leftrightarrow_{def} \\ X_1 \widehat{=} \cdots \widehat{=} X_n \wedge \forall (x_1, \ldots, x_n) \in X_1 \times \ldots \times X_n, x_1 \sim \ldots \sim x_n \Rightarrow P(x_1, \ldots, x_n)$$

The denotational semantics of SIGNAL in this model is given in figure 8. The symbol `:=` is not only denoted by $\widehat{=}_{\mathrm{v}}$ but also by dependence relations from the signals involved in the right part of an equation to the signal of the left part at the clock of the latter signal.

$$\begin{aligned}
[\![\mathtt{v}]\!] &\in \mathcal{D} \\
[\![\mathtt{f}]\!] &\in \mathcal{D} \times \cdots \times \mathcal{D} \longrightarrow \mathcal{D} \\
[\![\mathtt{x}]\!]_\rho &= \rho(\mathtt{x}) \in \mathcal{S} \\
[\![\mathtt{y:=f(x1,...,xn)}]\!]_\rho &= F^n_{[\![\mathtt{f}]\!]}([\![\mathtt{x1}]\!]_\rho, \cdots, [\![\mathtt{xn}]\!]_\rho) \wedge \textstyle\bigwedge_{i=1}^{n} [\![\mathtt{xi}]\!]_\rho \xrightarrow{\widehat{[\![\mathtt{y}]\!]_\rho}} [\![\mathtt{y}]\!]_\rho \\
[\![\mathtt{z:=x\ when\ y}]\!]_\rho &= [\![\mathtt{z}]\!]_\rho \widehat{=}_{\mathrm{v}} [\![\mathtt{x}]\!]_\rho \otimes^t [\![\mathtt{y}]\!]_\rho \wedge [\![\mathtt{x}]\!]_\rho \xrightarrow{[\![\mathtt{x}]\!]_\rho \sqcap [[\![\mathtt{y}]\!]_\rho]} [\![\mathtt{z}]\!]_\rho \\
[\![\mathtt{z:=x\ default\ y}]\!]_\rho &= [\![\mathtt{z}]\!]_\rho \widehat{=}_{\mathrm{v}} [\![\mathtt{x}]\!]_\rho \oplus [\![\mathtt{y}]\!]_\rho \wedge [\![\mathtt{x}]\!]_\rho \xrightarrow{\widehat{[\![\mathtt{x}]\!]_\rho}} [\![\mathtt{z}]\!]_\rho \wedge [\![\mathtt{y}]\!]_\rho \xrightarrow{\widehat{[\![\mathtt{y}]\!]_\rho} \setminus \widehat{[\![\mathtt{x}]\!]_\rho}} [\![\mathtt{z}]\!]_\rho \\
[\![\mathtt{y:=x\$\ init\ v}]\!]_\rho &= \mathtt{Pre}([\![\mathtt{v}]\!], [\![\mathtt{x}]\!]_\rho, [\![\mathtt{y}]\!]_\rho) \\
[\![P_1 \mid P_2]\!]_\rho &= [\![P_1]\!]_\rho \wedge [\![P_2]\!]_\rho \\
[\![P/\mathtt{x}]\!]_\rho &= \exists X \in \mathcal{S}, [\![P]\!]_{\rho, \mathtt{x} \mapsto X} \\
[\![(?\mathtt{x}_{1,..m} ! \mathtt{y}_{1,..n}) P]\!]_\rho &= \lambda(x_{1,..m}, y_{1,..n}).[\![P]\!]_{\rho, \mathtt{x}_1 \mapsto x_1, \cdots, \mathtt{x}_m \mapsto x_m, \mathtt{y}_1 \mapsto y_1, \cdots, \mathtt{y}_n \mapsto y_n}
\end{aligned}$$

Fig. 8. The denotational semantics of SIGNAL

Endochrony refers to the Ancient Greek: "$\varepsilon\nu\delta o$", and literally means "time defined from the inside". An endochronous specification defines a reactive system where "time defined from the inside" translates into the property that the production of its outputs only depends on the presence of its inputs. An endochronous system reacts to inputs by having an activation clock computable from that of its inputs. This activation clock directs the execution of the program. By contrast with the classical synchronous programming model, in which the activation clock of a system is not always definable, it is always possible to manipulate real or imaginary clocks in our model (because the set of clock $\mathcal{C}$ is a complete lattice) and eventually to compute a real (endochronous) signal. Hierarchization is the implementation of the property of endochrony for the compilation of SIGNAL programs. It is the medium used in SIGNAL for compiling the parallelism specified using synchronous composition. It consists of organizing the computation of signals as a tree that defines a correct scheduling of

computations into tasks. Each node of the tree consists of synchronous signals. It denotes the task of computing them when the clock is active. Each relation of a node with a sub-tree represents a sub-task of smaller clock.

6 Related Works

There are several ways to characterize the essentials of the synchronous paradigm. In [12], we introduce a co-inductive semantics of SIGNAL. A theorem library is developed and enable to express and prove not only liveness and safety properties of a synchronous program but also its correctness and its completeness. But it is not powerful enough to deal with more theoretical aspect of synchronous programming such as dependencies. The semantics of a synchronous language can be described in a better way with Symbolic Transition System (STS) [13]. This is a formalism on which fundamental questions can be investigated. But it manipulates the absence of a signal as a special value. This is not consistent with reality: the absence of a signal has to be inferred by the program (endochrony). In [3], STS is extended with preorders and partial orders to model causality relations, schedulings and communications. This pre-order theoretic model is put into practice in the design of BDL ([14]), a synchronous specification language that uses families of pre-orders to specify systems. In [6], the problem of characterizing synchrony without using a special symbol for absence is addressed in terms of multiple onput-output sequential machines. In [8], the language SIGNAL has been modelled in interaction categories ([1]) where processes are morphisms and objects are types of processes.

7 Conclusion

We have defined a unified model which formalizes all aspects of the development of a reactive system using the underlying programming methodology of synchronous languages (from relations to circuits). This model uses basic notions of set-theory and order-theory and has been specified and validated using the COQ theorem prover. This implementation is part of the development of a certified SIGNAL compiler.

References

1. Samson Abramsky. Interaction Categories (Extended Abstract). In G. L. Burn, Simon J. Gay, and M. D. Ryan, editors, *Theory and Formal Methods 1993: Proceedings of the First Imperial College Department of Computing Workshop on Theory*

and Formal Methods, pages 57–70. Springer-Verlag Workshops in Computer Science, 1993.

2. Albert Benveniste, Paul Le Guernic, and Christian Jacquemot. Synchronous programming with events and relations : the Signal language and its semantics. *Science of Computer Programming*, 16:103–149, 1991.
3. Albert Benveniste, Paul Le Guernic, and Pascal Aubry. Compositionality in dataflow synchronous languages: specification & code generation. Research report 3310, INRIA, 1997.
4. Gérard Berry and Georges Gonthier. The ESTEREL synchronous programming language: design, semantics, implementation. *Science of Computer Programming*, 19:87–152, 1992.
5. Gerolamo Cardano. *Ars Magna*. 1545.
6. P. Caspi. Clocks in dataflow languages. *Theoretical Computer Science*, 94(1):125–140, March 1992.
7. Bruno Barras et al. *The Coq Proof Assistant Reference Manual - Version 6.2*. INRIA, Rocquencourt, May 1998.
8. Simon J. Gay and Raja Nagarajan. Modelling SIGNAL in Interaction Categories. In Geoffrey L. Burn, Simon J. Gay, and Mark D. Ryan, editors, *Theory and Formal Methods 1993: Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods*. Springer-Verlag Workshops in Computer Science, 1993.
9. N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language Lustre. *Proc. of the IEEE*, 79(9):1305–1320, September 1991.
10. Nicolas Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Pub., 1993.
11. D. Nowak, J.P. Talpin, and T. Gautier. Un système de modules avancé pour Signal. Technical Report 3176, Irisa / Inria-Rennes, 1997.
12. David Nowak, Jean-René Beauvais, and Jean-Pierre Talpin. Co-inductive Axiomatization of a Synchronous Language. In *Proceedings of Theorem Proving in Higher Order Logics (TPHOLs'98)*, number 1479 in LNCS, pages 387–399. Springer Verlag, September 1998.
13. Amir Pnueli, Natarajan Shankar, and Eli Singerman. Fair Synchronous Transition Systems and their Liveness Proofs. Technical Report SRI-CSL-98-02, The Weizmann Institute of Science and SRI International, 1998.
14. J.-P. Talpin, A. Benveniste, B. Caillaud, C. Jard, Z. Bouziane, and H. Canon. Bdl, a language of distributed reactive objects. In *Proceedings of the International Symposium on Object-Oriented Real-Time Distributed Computing*. IEEE press, april 1998.
15. G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course,* Bad Honnef, September 1986, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer-Verlag, 1987.