



## Project IST-2001-38314, COLUMBUS

### Design of Embedded Controllers for Safety Critical Systems

#### Deliverables DMM1 and DMM2

#### Report on formal framework of meta-models – resp. INRIA

**Authors:** B. CAILLAUD (INRIA) and E. JACKSON (Vanderbilt University)

**August 2004**

#### Project Co-ordinator

**Organisation:** *Department of Electrical and Computer Engineering  
University of Cambridge*

**Responsible person:** *Dr John Lygeros*

**Address:** *University Campus  
Rio, Patras, GR 26500, Greece*

**Phone:** *+30 2610 996458*

**Fax:** *+30 2610 991812*

**E-mail:** *lygeros@ee.upatras.gr*

#### Consortium

	<b>Participant name</b>	<b>Acronym</b>	<b>Role</b>
1	<i>University of Patras</i>	<i>UPAT</i>	<i>Coordinator</i>
2	<i>University of Cambridge</i>	<i>UCAM</i>	<i>Contractorr</i>
3	<i>University of l'Aquila</i>	<i>AQUI</i>	<i>Contractor</i>
4	<i>Institut National de Recherche en Informatique et en Automatique</i>	<i>INRIA</i>	<i>Contractor</i>
5	<i>University of California, Berkeley</i>	<i>UCB</i>	<i>Contractor</i>
6	<i>Vanderbilt University</i>	<i>VU</i>	<i>Contractor</i>
7	<i>PARADES, Rome</i>	<i>PARADES</i>	<i>Subcontractor</i>

**DOCUMENT HISTORY**

<b>Release</b>	<b>Date</b>	<b>Reason of change</b>	<b>Status</b>	<b>Distribution</b>
0.1	08/10/04	Final draft	Draft	Partners
0.2	08/16/04		Final	All

## TABLE of CONTENTS

<b>1. LIST OF DUE DELIVERABLES FOR WPMM.....</b>	<b>4</b>
<b>2. SUMMARY OF RESULTS.....</b>	<b>5</b>
2.1 Recalling the Objectives.....	5
2.2 Meta-Model of a Reactive Synchronous Formalism with Dataflow and Clock Aspects.....	5
2.3 Compositional Executable Semantics of the Formalism.....	6
<b>3. META-MODEL.....</b>	<b>6</b>
3.1 Introduction.....	6
3.2 Architectural Aspect.....	7
3.3 Dataflow Aspect.....	8
3.4 Clock Aspect.....	8
<b>4. CONCRETE SEMANTICS.....</b>	<b>9</b>
4.1 A Petri Net Based Concurrent Semantics.....	9
4.2 Composition of Nets by Identification of Places and Transitions.....	11
4.3 Semantics of Dataflow Aspects.....	11
4.4 Semantics of Clock Aspects.....	12
<b>5. REFERENCES.....</b>	<b>13</b>

## 1. LIST OF DUE DELIVERABLES FOR WPMM

---

Id	Deliverable	Respons. Partner	Original due date/ milestone	Revised due date/ milestone	Actual delivery	Status / Comments
DMM1	Report on formal framework of meta-models	INRIA	31/12/03		23/01/04	
DMM2	Report on use of meta-model for hybrid systems.	INRIA	30/6/04		30/06/04	

**Note:** Deliverables DMM1 and DMM2 have been merged into a single document. The reason for producing a single deliverable is that the materials to be included in the two deliverables are complementary, with tight inter-dependencies: Deliverable DMM1 would have contained the definition of the meta-model (the abstract syntax of a reactive synchronous formalism, taking ideas from aspect oriented programming, *cf.* Section mm), while DMM2 would have contained the definition of the concrete semantics of the meta-model (executable semantics, based on one-safe Petri nets, *cf.* Section cs). Clearly, a semantics of a specification formalism can be defined only when its syntax has been defined. Conversely, the syntax of the formalism has been chosen so that the decomposition into two aspects (*dataflow* and *clock*) could be reflected into the concrete semantics, namely that the semantics of a specification consisting in two aspects is the composition of the semantics of the aspects. This explains why the deliverables have been merged into a single document.

## 2. SUMMARY OF RESULTS

---

### 2.1 Recalling the Objectives

In providing a specification, one needs a capability of describing the following three aspects: actions, constraints, and their *refinement*. This is true whether it is a specification of a system (behaviour or architecture) or of the environment. The objective of this work package WPMM is the development of meta-models that have precisely this capability. The ultimate goal is to contribute to a meta-modelling based open tool integration framework, which enables the rapid composition of domain-specific design tools using reusable tool components.

This deliverable investigates the design of a domain-specific synchronous reactive formalism, dedicated to the specification of real-time embedded systems, to be integrated in a meta-modelling tool, such as GME Entrée de bibliographie, developed at Vanderbilt University. Its aim is to bridge the gap between theoretical models proposed in WPTA and state of the art software engineering methodologies. In particular, it integrates concepts inherited from *aspect oriented programming* Entrée de bibliographie: The formalism allows a complete separation of concerns between the specification of dataflow paths and the specification of control.

A particular effort has been put on the definition of a formal executable semantics, expressed as compositions of simple one-safe Petri nets. This semantics can be used to build interactive simulators, perform code generation, or prove the correctness of syntactic transformations at the meta-model level. The striking property of the semantics is that it is compositional, and that it reflects in the semantic domain the separation of concerns allowed by the aspect-oriented features of the formalism.

### 2.2 Meta-Model of a Reactive Synchronous Formalism with Dataflow and Clock Aspects

Synchronous reactive systems are systems that perform (logically) instantaneous computations at discrete points in time, possibly (but not necessarily) in response to some input. These systems are often required to be deterministic and compute using a bounded amount of memory. Synchronous reactive languages Entrée de bibliographie only allow to express programs that have these properties. Compilers of these languages reject programs which do not have these properties. As one may imagine, these properties are not easy to check, and even, for some of them, their verification can not be automated. Compilers, usually accept only programs for which checking these properties is automated. The consequence is that programs may be rejected even if they satisfy (non-trivially) the required properties of the synchronous programming paradigm.

Two styles of synchronous languages have emerged: The imperative languages, like Esterel Entrée de bibliographie, and the declarative languages, like Lustre Entrée de bibliographie. Imperative synchronous languages combine the concepts of concurrency, instantaneous broadcast communication between processes, and preemption operators. A language like Esterel is imperative because the control flow (*eg.*, state transitions) is directly related to the syntax of the language. In fact, there is a close and intuitive relationship between concurrent finite state machines and Esterel. This relationship is best understood on the SyncCharts Entrée de bibliographie language, a StateCharts Entrée de bibliographie derivative, with preemption operators and a synchronous semantics. This close relationship to state machines emphasizes the power of imperative languages: Imperative languages succinctly specify control intensive applications. However, there is a price to be paid for specifying systems with imperative synchronous languages, namely that the underlying data path is difficult to comprehend and analyze. Each state or transition in the state machine performs some computation on data, and tracking the flow of data turns out to be difficult.

The declarative languages like Lustre and Signal Entrée de bibliographie take a control engineering approach. From this perspective, a synchronous system is a dynamical system defined by a system of equations. These equations focus attention to the underlying global data path, and in fact, the graphical notation for Lustre is a dataflow diagram where vertices indicate primitive operation (*eg.*, arithmetic operators, boolean connectors) and edges indicate the flow of data. These dataflow operators process streams of data, which are compound sequences consisting of a partial sequence of values and a

sequence of booleans. The boolean sequence is called the *clock*, and the  $n^{\text{th}}$  boolean indicates if the value sequence takes on a value at the  $n^{\text{th}}$  instant in time. The operators impose constraints on the allowed clocks, and verifying that all constraints are satisfied amounts to guaranteeing determinism. These languages are declarative because the equations express *what* has to be done, but they do not explicitly define the control flow (*ie.*, state machine) that defines *how* it is to be done. In fact, generating this control flow requires complex algorithms. The focus on dataflow makes specifying data intensive systems a natural process. However, the declarative nature of the control flow means that describing and debugging complex, state dependent control can be difficult.

The concept of *aspects* Entrée de bibliographie, introduced by the software engineering community, allows to specify separately properties of a system or component that are not inter-related. This concept has opened the way to the development of powerful design methodologies (*aspect oriented system design*, AOSD) and tools (for instance, AspectJ). Specifications in our formalism can be decomposed in several aspects, with a clear separation of concerns between aspects. For instance, it is possible to describe the possible data-paths of a component or system, independently of the specification of its synchronization properties.

This deliverable collects original ideas by Ethan Jackson, Vanderbilt University Entrée de bibliographie and Benoit Caillaud, IRISA/INRIA Rennes. The aim of this deliverable is two-fold: In Section mm, a specification formalism based on the synchronous paradigm is introduced. This formalism is designed to be integrated within the GME meta-modelling tool Entrée de bibliographie, developed at Vanderbilt University. The formalism allows for a strict separation between three aspects: a clock aspect (specification of synchronization properties, Section mmc) in which control can be expressed with state machines, a dataflow aspect (specification of data-paths, Section mmd) in which data dependencies can be expressed with a graphical notation very similar to SCADE/Lustre Entrée de bibliographie, and an architectural aspect (Section mma) in which the hierarchical architecture of systems can be described. Section cs defines a concrete executable semantics of the meta-model previously defined. The aim and principle of this executable semantics is explained below.

## 2.3 Compositional Executable Semantics of the Formalism

Section cs details a concrete executable semantics of the meta-model. It is a compositional true-concurrency semantics, based on one-safe Petri nets Entrée de bibliographie. In other words, the semantics of a model consisting in several components is the composition of the semantics of its components. Moreover, the semantics respects the decomposition into dataflow and clock aspects. Indeed, the semantics of a component is the composition of the semantics of its dataflow aspect (Section csd) with that of its clock aspect (Section csc). The semantics is expressed in terms of one-safe Petri nets, which allow involved composition operators (composition by identification of transitions or places). These composition operators extend strictly the well-known, all purpose, *partially synchronized product of transition systems* Entrée de bibliographie Entrée de bibliographie, and it would have been technically difficult to obtain a compositional semantics based only on partially synchronized products of transition systems.

The semantics can be used to build interactive simulators, perform code generation, or prove the correctness of syntactic transformations at the meta-model level.

## 3. META-MODEL

---

### 3.1 Introduction

A specification formalism based on the synchronous paradigm is introduced. This formalism allows for a strict separation between three aspects: a clock aspect (Section mmc) in which control can be expressed with state machines, a dataflow aspect (Section mmd) in which data dependencies can be expressed with a graphical notation very similar to SCADE/Lustre Entrée de bibliographie, and an architectural aspect (Section mma) in which the hierarchical architecture of systems can be described.

This formalism allows to describe all possible data-paths, without having to consider the different modes of operation of the system. For instance an aircraft can measure its airspeed thanks to several pressure sensors positioned in various locations of the airframe. The selection of the pressure sensor

that gives the most accurate data depends not only on the configuration in which the aircraft is flying (incidence, bank, symmetry, air-speed, position of flaps, etc.), but also on the possible failures of the sensors. The possible airspeed data-paths can be described independently of the specification of the source selection algorithm. Moreover, this can be done in a single diagram: The dataflow aspect (Section mmd). The same argument applies to the clock aspect: It is possible to describe the source selection algorithm, independently of the actual data-path, in a single clock aspect diagram (Section mmc). This illustrates the clear separation of concerns between dataflow and clock aspects.

Section cs will reveal that this syntactic separation into aspects is reflected on the semantics of the formalism. More precisely, the semantics of the dataflow and clock aspects can be defined independently of one-another. The semantics of a complete specification is then some composition of the semantics of the independent clock and dataflow aspects contained in the specification. The architectural aspect is of a different nature, as it is used to define how components and clock or dataflow aspects should be composed.

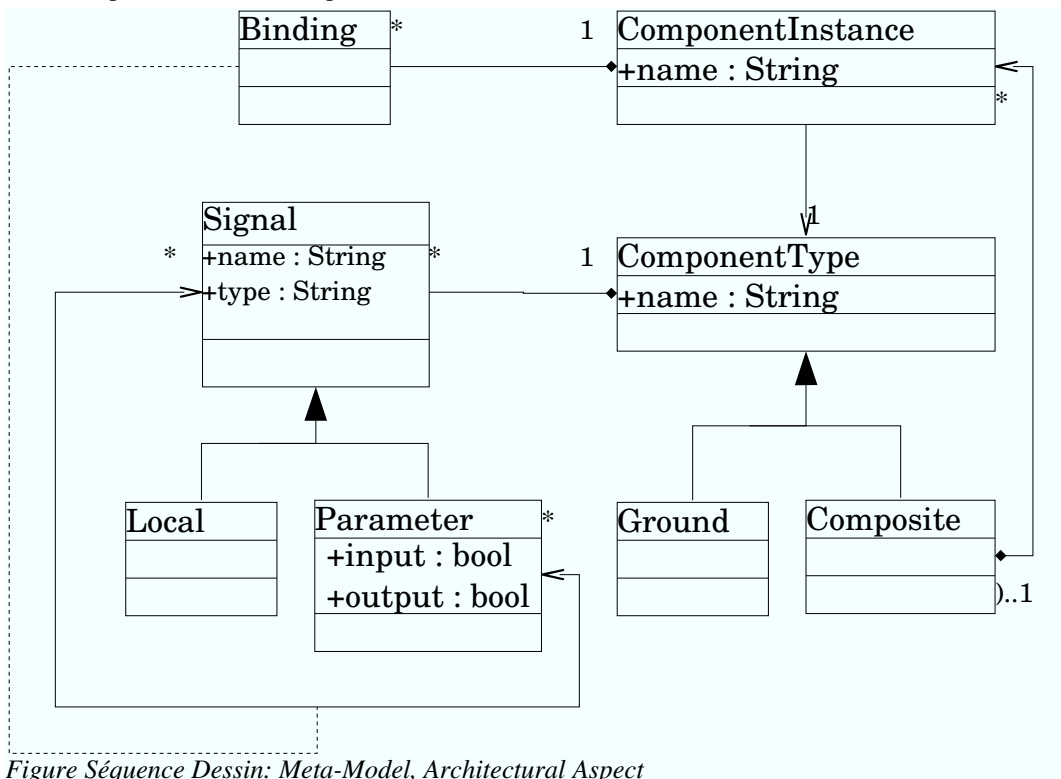


Figure Séquence Dessin: Meta-Model, Architectural Aspect

### 3.2 Architectural Aspect

The architectural aspect allows to describe the architecture of a system as a hierarchy of components, where each component is an instance of a component type. A component type contains local signals, input and output signal, and sub-components. The leaves of this tree of components are instances of component types taken from a set of pre-defined component types (*ground component types*). The architectural aspect serves as a substrate for the other two aspects. Its abstract syntax is given as a UML class diagram, in Figure Dessin.

### 3.3 Dataflow Aspect

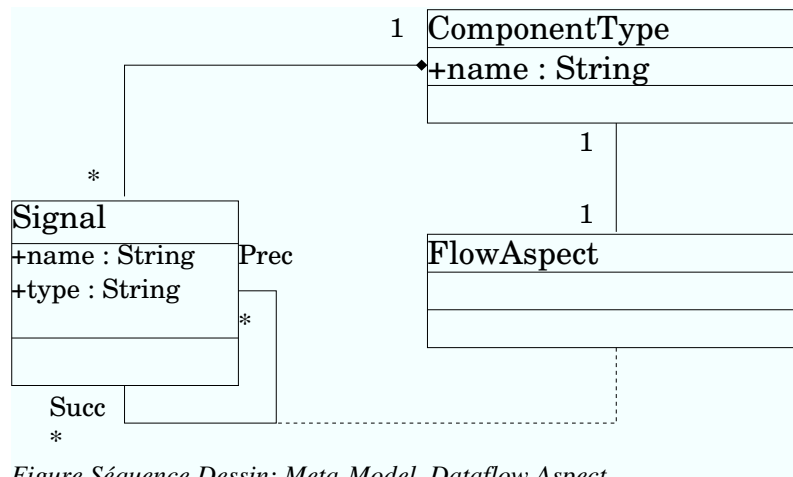


Figure Séquence Dessin: Meta-Model, Dataflow Aspect

The purpose of the dataflow aspect (syntax defined in Figure Dessin) is to allow the specification of all possible data-paths in a system. These data-paths are materialized by causality edges between signals. Therefore, an instance of the dataflow aspect is a graph which vertices are the local and interface signals of a component type, and the edges are causality dependencies between these signals. However, unlike many dataflow formalism, a causality dependence edge is meaningful only when both signals connected by the edge are active (*cause* and *consequence* signals). If either signal is inactive (not to be computed in a reaction), then this causality dependence does not need to be enforced. In particular the consequence signal can be computed independently of its cause, once it is known that the cause is not active. A signal may have from zero to an arbitrary number of causes. The existence of cycles in the dataflow graph does not mean that the specification is incorrect, as its interpretation depends on the synchronization constraints put on the signals. Synchronization constraints between signals are described in the clock aspect, and the expected separation of concerns between the two aspects fully justifies this weak semantics of the dataflow diagram. The semantics of the dataflow aspect is defined formally in Section csd of this deliverable.

### 3.4 Clock Aspect

The abstract syntax of the clock aspect is given in Figure Dessin. Instances of the clock aspect are state-machines, where states describe modes of operation of a component. In this respect, the clock aspect is similar to mode-automata. However, the clock aspect and mode-automata differ on the expression of conditions under which a mode is selected. In mode-automata, a system remains in a mode until a transition from the active mode to a new mode of operation is enabled. In the clock-aspect, a mode is selected for every reaction of the system, according to the current clock-mode and the set of successor clock-modes.

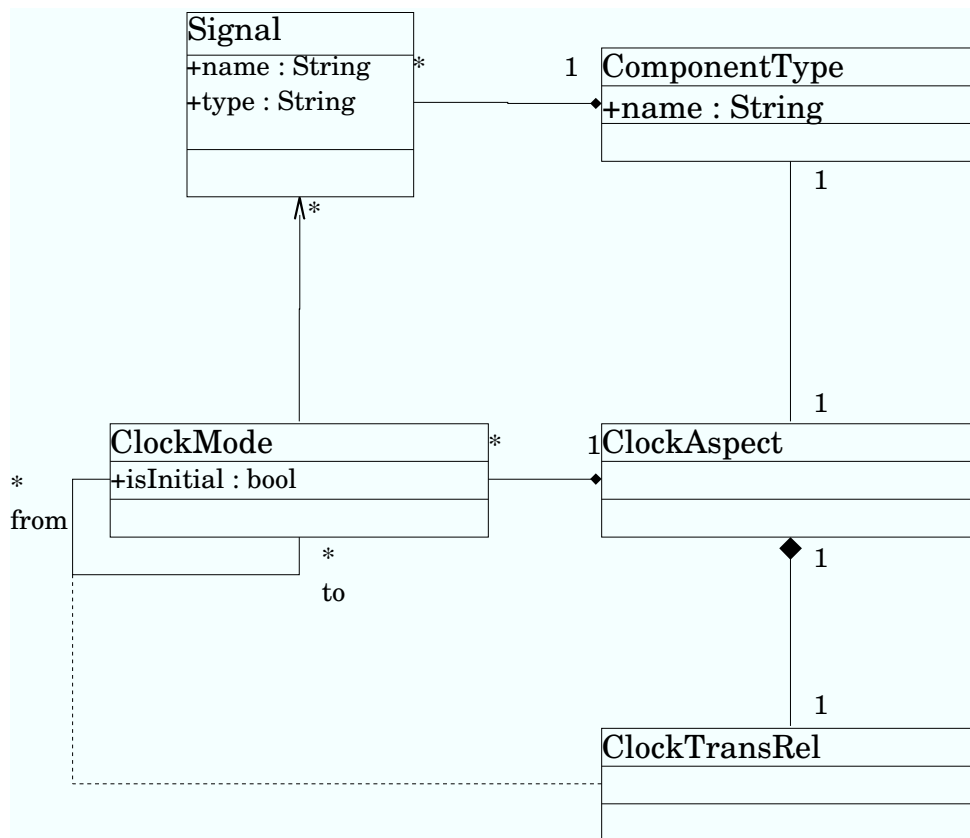


Figure Séquence Dessin: Meta-Model, Clock Aspect

Each clock-mode describes the set of signals that must be evaluated. Note that it is possible to control the activation of a sub-component, thanks to an enabling input signal.

## 4. CONCRETE SEMANTICS

### 4.1 A Petri Net Based Concurrent Semantics

The aim of this section is the definition of an executable semantics for the dataflow and clock aspects defined in the previous sections. This semantics should be compatible with the SIGNAL semantics Entrée de bibliographie, as it is implied by the POLYCHRONY tool Entrée de bibliographie. Moreover, its definition should be compositional, dealing with the two aspects independently. Synchronous programs are inherently concurrent: variable assignments can occur independently of one-another. This concurrency should be reflected into the concrete semantics.

Petri nets are best suited to express the behaviour of concurrent discrete-event systems. It benefits from a rigorous mathematical semantics, a powerful theory, and flexible composition operators. For all these reasons, one-safe Petri nets have been chosen to express the concurrent semantics of the clock and dataflow aspects. The semantics of an aspect, component or complete specification is defined as the composition of the semantics (eg., one-safe nets) of its most elementary syntactic constructs: clock modes (for clock aspects), data or control dependencies (for dataflow aspects). The composition of the semantics all dataflow and clock aspects that appear in a specification is parametrized by the architectural aspects of the specification.

The semantics defined below is a *micro-step* semantics, in which elementary transitions (steps) correspond to the assignment of not more than one variable. This is in contrast with *macro-step* semantics, where complete reactions are evaluated in a single transition. This micro-step semantics bear similarities with the constructive semantics of Esterel Entrée de bibliographie. The two major differences is that the operational semantics proposed below is not defined by logical rules, but rather by mapping instances of the meta-model to one-safe Petri nets. The second difference is that the semantics considers only those firing sequences of the nets that do not lead to deadlock. More precisely, the one-safe nets are composed of two sorts of places: *macro-places* (represented by two

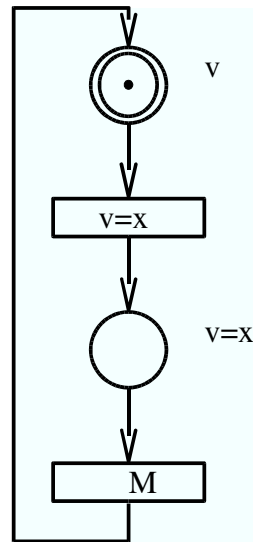


Figure Séquence Dessin: Semantics of a Variable Assignment

concentric circles) and *micro-places* (single circle). A reaction is considered to be *in progress* whenever a micro-place is marked. Said differently, a net is in a *synchronization marking* when only

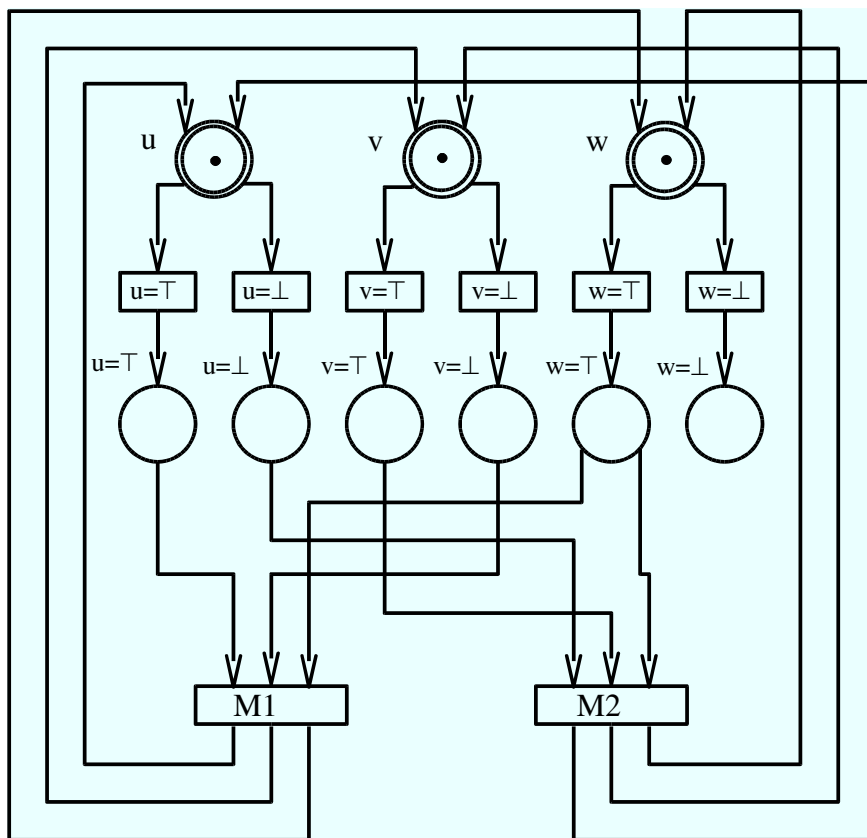


Figure Séquence Dessin: Semantics of a Simple Clock Aspect, Synchronization of Assignments

macro-places are marked. It is possible for the nets to deadlock in a marking that is not a synchronization marking. The semantics retains only those firing sequences that reach markings from which a synchronization marking can be reached. Said differently, any behaviour can be extended in a behaviour where all reactions have been completed.

## 4.2 Composition of Nets by Identification of Places and Transitions

Composition of Petri nets by identification of transitions is the most well-known composition operator in the theory of Petri-nets. This operator is the adjoint of the partially-synchronized product of transition system. Indeed, the marking graph of the composition of two nets, by identification of some of its transition, is a partially synchronized product of the marking graphs of the nets to be composed. In set theoretic terms, the composition by identification of transitions consists in computing a fibered product of the sets of transitions of the nets to be composed.

Composition of Petri-nets by identification of places is another composition operator that can be found in the literature. Not only it has been studied from a theoretical point of view, but it has also been used as a powerful composition operator in Petri net-based specification formalisms, such as the *Petri module* formalism. Formally, the composition by identification of places amounts to compute an amalgamated sum of the sets of places of the nets to be composed.

The composition operator used for the definition of the semantics of a meta-model consists in identifying places with identical labels (amalgamated sums of sets of places) and identifying transitions with identical labels (fibered product of sets of transitions). Unlabelled places or transitions are copied, without identification with another place/transition.

Given a set of names  $\Lambda$ , a *labelled one-safe net* is a tuple  $N=(P,T,\lambda,F)$  where  $P$  is a set of places,  $T$  is a set of transitions, disjoint from  $P$ ,  $\lambda : P \cup T \rightarrow \Lambda$  is a partial labelling function, that is injective when restricted to  $P$  or  $T$ , and  $F \subseteq (P \times T) \cup (T \times P)$  is the flow relation.

Given three sets  $A, B, C$  and two injective partial function  $f : A \rightarrow C$  and  $g : B \rightarrow C$ , the *fibered product* of  $A$  and  $B$  is the set  $A \times_g B = \{ (a,b) \mid a \in A, b \in B, f(a)=g(b) \} \oplus \{ a \mid a \in A, f(a) \text{ is undefined or } f(a) \notin g(B) \} \oplus \{ b \mid b \in B, g(b) \text{ is undefined or } g(b) \notin f(A) \}$ , where  $\oplus$  is the disjoint union of sets. The amalgamated sum of  $A$  and  $B$ , denoted  $A \uparrow_g B$ , is the quotient of  $A \oplus B$  by the least equivalence relation  $\sim$ , such that  $\forall a \in A, \forall b \in B, f(a)=g(b) \Rightarrow a \sim b$ . The equivalence class of  $x$  is denoted  $[x]$ .

Given two nets  $N=(P,T,\lambda,F)$  and  $N'=(P',T',\lambda',F')$ , the composition of two nets, is the net  $N * N' = (P'',T'',\lambda'',F'')$ , with:

- $P'' = P \uparrow_{\lambda \cup \lambda'} P'$ ,
- $T'' = T \times_{\lambda \cup \lambda'} T'$ ,
- $\lambda'' = \varphi^{-1 \circ} (\lambda \cup \lambda')$ ,
- $F'' = \varphi^{-1 \circ} (F \cup F')$ ,

Where  $\varphi \subseteq (P'' \cup T'') \times ((P \cup T) \oplus (P' \cup T'))$  is the least relation such that:

- $\forall p \in P \oplus P' \quad ([p], p) \in \varphi$ ,
- $\forall (t, t') \in T \times T' \quad ((t, t'), (t, t')) \in \varphi$  and  $((t, t'), t') \notin \varphi$ ,
- $\forall t \in T, \lambda(t) \text{ is undefined or } \lambda(t) \notin \lambda'(T') \Rightarrow (t, t) \in \varphi$ ,
- $\forall t' \in T', \lambda'(t') \text{ is undefined or } \lambda'(t') \notin \lambda(T) \Rightarrow (t', t') \in \varphi$ .

This composition of nets will be used to define the semantics of a model (instance of the meta-model) by composing simple nets defining the semantics of elementary constructions of the formalism: causality edges, modes, and signal synchronizations.

## 4.3 Semantics of Dataflow Aspects

The semantics of a dataflow graph is defined by a one-safe net obtained by composition of one net per edge of the graph. Indeed a causality edge is meaningful if and only if both ends of the edge are present. The semantics of an edge “ $x \rightarrow y$ ” is defined below. This net ensures that assignments of variable “ $x$ ” precedes assignments of variable “ $y$ ”, when both variables are present. The formal semantics of a causality edge is defined in Figure Dessin.

### 4.4 Semantics of Clock Aspects

Each component encompasses a control state machine. The behaviour of this state machine is defined by a one-safe net, obtained by composition of several sub-nets: one net for each transition of the clock aspect and one net for each assertion on the presence or absence of a signal in a clock mode.

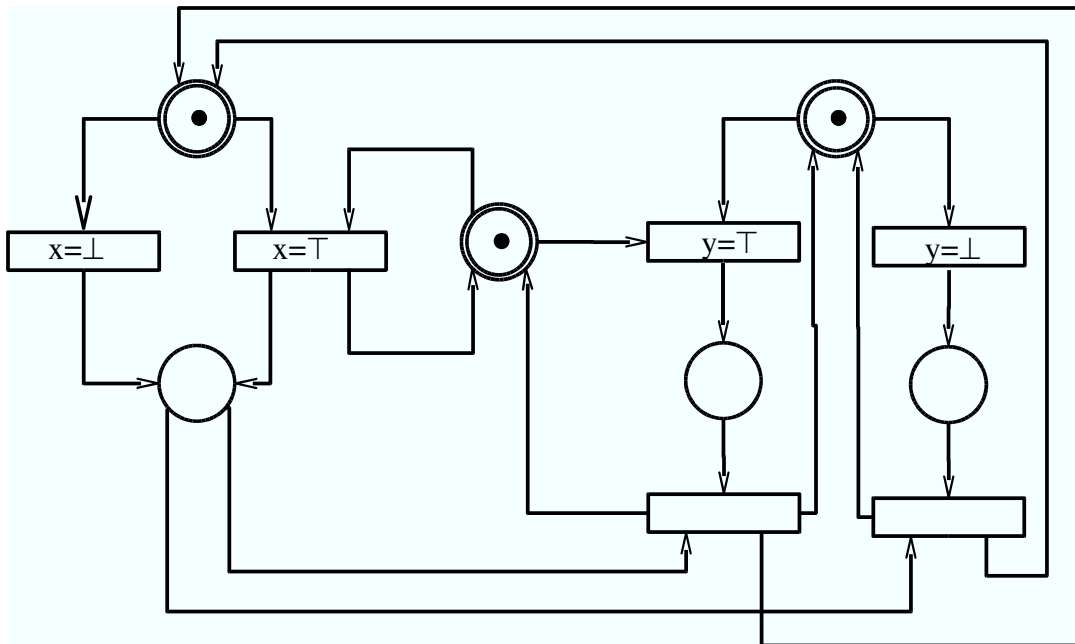


Figure Séquence Dessin: Semantics of a Causality Edge

**Semantics of a transition:** For each transition from mode  $M'$  to mode  $M$ , there is a net composed of three places ( $M'$ ,  $M-$ ,  $M$ ) and two transition, one silent, and one labelled  $M$ . The three places denote the three possible states of the system that have to be considered for the evaluation of a mode transition: mode  $M'$  completed (place  $M'$  is marked), mode  $M$  in progress (place  $M-$  is marked), and mode  $M$  completed (place  $M$  is marked). This net is shown in Figure Dessin. If a mode is initial, it is considered that there exists a transition from a dummy mode called INIT which corresponding place (labelled INIT) is initially marked. A complete example of state-machine semantics is detailed in Figure Dessin.

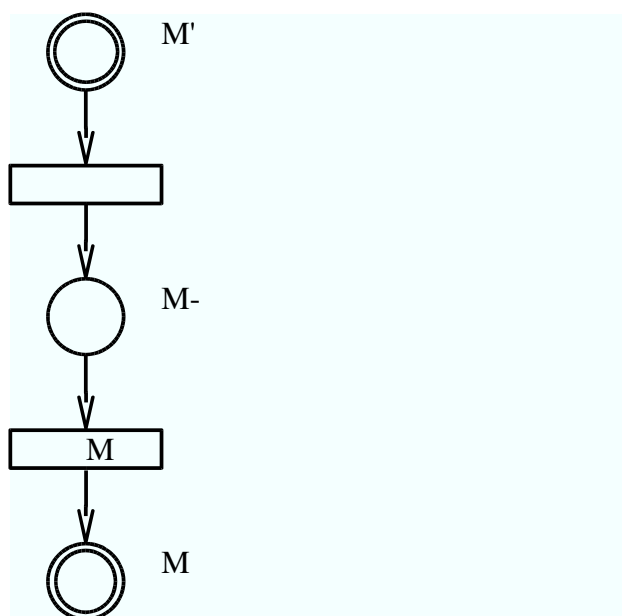


Figure Séquence Dessin: Semantics of a Mode Transition

**Semantics of an assignment:** In mode  $M$ , the assignment of signal  $v$  to value  $x \in \{\top, \perp\}$  is represented by a net consisting in two places ( $v$  and  $v=x$ ) and two transitions ( $v=x$  and  $M$ ). This net is shown in Figure Dessin.

A complete example of clock aspect is detailed in Figures Dessin and Dessin. The first figure contains the net corresponding to a state machine with two initial modes ( $M1$  and  $M2$ ). At each time, the machine may choose between one of the two modes. The second figure contains the net corresponding to the synchronization constraints on three signals ( $u, v$ , and  $w$ ) expressed in the two modes: In mode  $M1$ , signals  $u$  and  $w$  are present, and signal  $v$  is absent. In mode  $M2$ , signals  $v$  and  $w$  are present and signal  $u$  is absent.

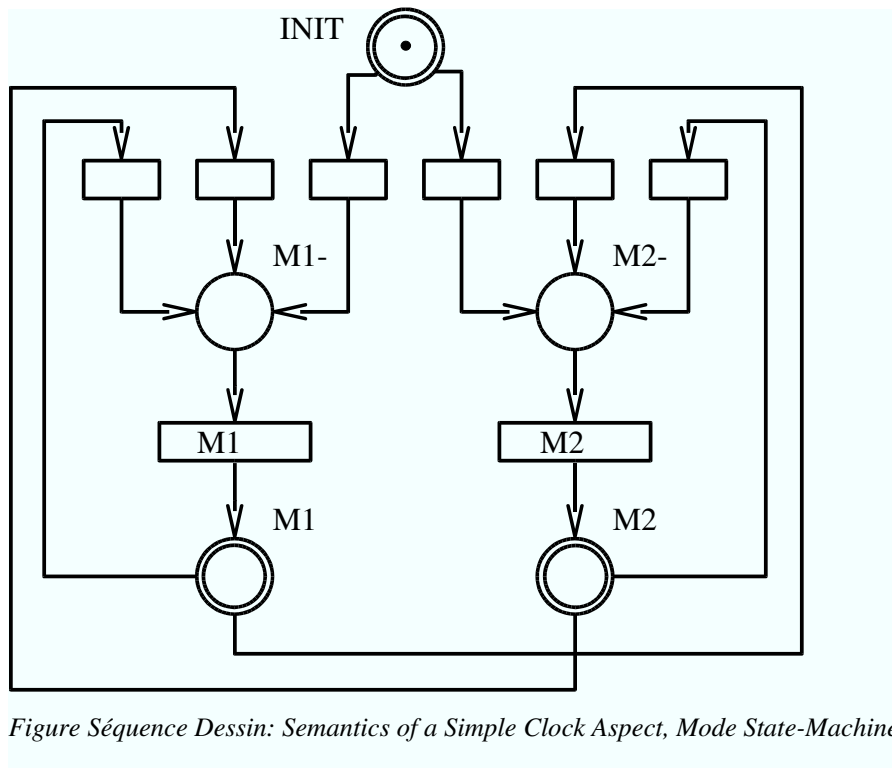


Figure Séquence Dessin: Semantics of a Simple Clock Aspect, Mode State-Machine

## 5. REFERENCES

- [KSLB03] G. Karsai, J. Sztipanovits, A. Ledeczi, T. Bapty, Model-Integrated Development of Embedded Software, Proceedings of the IEEE, vol. 91 , 2003.
- [KLMMVLLI97] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, L. Videira Lopes, J.-M. Loingtier, J. Irwin, Aspect-Oriented Programming, Proceedings of Object-Oriented Programming, 11th European Conference, ECOOP, 1997.
- [BCEHLGS03] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, R. de Simone, The Synchronous Languages Twelve Years Later, Proceedings of the IEEE, vol. 91 , 2003.
- [FS91] F. Boussinot, R. de Simone, The Esterel Language, Another Look at Real Time Programming, Proc. of the IEEE, vol. 79 , 1991.
- [HCRP91] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud, The synchronous data-flow programming language LUSTRE, Proceedings of the IEEE, vol. 79 , 1991.
- [Andre96] C. Andre, Representation and Analysis of Reactive Behaviors: A Synchronous Approach, CESA'96, IEEE-SMC, Computational Engineering in Systems Applications, 1996.
- [Harel87] D. Harel, Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming, vol. 8 , 1987.
- [Jack04] E. Jackson, A Fine-Grained Modal Semantics for Synchronous Reactivity, unpublished, 2004.

- [SCADE03] Esterel Technologies, Efficient Development of Airborne Software with SCADE Suite, 2003, <http://www.esterel-technologies.com/>
- [Reut90] C. Reutenauer, The mathematics of Petri nets, Prentice-Hall, 1990.
- [BFP02] L. Bernardinello, C. Ferigato, L. Pomello, Towards Modular Synthesis of Elementary Net Systems, Synthesis and Control of Discrete Event Systems, 2002.
- [BBCPP03] M. Bednarczyk, L. Bernardinello, B. Caillaud, L. Pomello, W. Pawlowski, Modular system development with pullbacks, Applications and Theory of Petri Nets 2003, 2003.
- [MR98] F. Maraninchi, Y. Remond, Mode-Automata: About Modes and States for Reactive Systems, European Symposium on Programming, ESOP'98, 1998.
- [BLGJ91] A. Benveniste, P. Le Guernic, C. Jacquemot, Synchronous programming with events and relations: The SIGNAL language and its semantics, Science of Computer Programming, vol. 16 , 1991.
- [Polychrony03] ESPRESSO team of IRISA, Signal / Polychrony, 2003, <http://www.irisa.fr/espresso/Polychrony/>
- [Berry99] G. Berry, The Constructive Semantics of Pure Esterel, draft version 3, 1999, <http://www.esterel-technologies.com/>
- [JL02] G. Juhas, R. Lorenz, Modelling with Petri Modules, Synthesis and Control of Discrete Event Systems, 2002.