Towards a formal language for systemic requirements

Yann Hourdel

LIX, École Polytechnique, 91128 Palaiseau Cedex, France,
yann.hourdel@polytechnique.edu

Abstract. This work is an attempt to contribute to the field of systems architecture. More precisely, it deals with complex[1] engineered systems analysis. Many informal methods for architecting such systems have been described over the past decade, but a lot of specific points still need to be clarified by a precise (mathematically formalized) definition. In particular, the languages used to manipulate system properties during systemic analysis are one big issue to be tackled. Our approach is the following: we take the framework described in [6] and reviewed in [4] as a starting point, and build a formal language to express functional (behaviour) requirements on models. The result is a formal language that allows architects to manipulate precise constraints on their models and, more importantly, translate them across subsequent systemic levels.

Keywords: Systems modeling, Systems architecture, Systems Engineering, Architecture framework

Introduction

In this work, systems are seen as black boxes. Their behaviour is only functional, so we will only express functional constraints on them. This kind of constraints is one perfect thing to be formalized, since it is very close to mathematical notions. Moreover, let us precise some important points:

- This work only deals with deterministic systems, for which we are able to describe the set of possible executions.
- In this work, time is considered discrete. That allows us to speak about the *previous* or *next* instant of an t.

[1] large, integrated, dense and heterogeneous

Notations

In the present work, the concatenation of two vectors A and B will be noted $A \otimes B$.
More generally, we will note $f \otimes g : A \otimes C \to B \otimes D$ the concatenation of $f : A \to B$ and $g : C \to D$.

1 Preliminary definitions

In this section, we recall the definitions introduced in [3] and reviewed in [4] to formalize the notion of *system*, a timed extension of Mealy machines to model heterogeneous integrated systems and their integration.

Definition 1 (Type). *The notion of* ***type*** *will be the classical "set of values" one.*

1.1 Time

Time is an underlying, yet very important, point of our formal approach. Indeed, real-life systems are naturally described according to various types of "times". As a result, we need to deal uniformaly with both continuous and discrete times. While this problem has been shown hard by [2], some solutions have been found in studies like [5] to introduce formal models for mixed discrete-continuous systems. We give here a set of definitions to handle such systems.

Informally, as very well expressed in [3], time is "a linear quantity composed of ordered moments, pairs of which define durations".

Definition 2 (Time reference). *A* **time reference** *is an infinite set T together with an internal law $+^T : T \times T \to T$ and a pointed subset $(T^+, 0^T)$ satisfying the following conditions:*

- *upon T^+:*
  - $\forall a, b \in T^+,\ a +^T b \in T^+$ — *closure (Δ_1)*
  - $\forall a, b \in T^+,\ a +^T b = 0^T \implies a = 0^T \wedge b = 0^T$ — *initiality (Δ_2)*
  - $\forall a \in T^+,\ 0^T +^T a = a$ — *neutral to left (Δ_3)*
- *upon T:*
  - $\forall a, b, c \in T,\ a +^T (b +^T c) = (a +^T b) +^T c$ — *associativity (Δ_4)*
  - $\forall a \in T,\ a +^T 0^T = a$ — *neutral to right (Δ_5)*
  - $\forall a, b, c \in T,\ a +^T b = a +^T c \implies b = c$ — *cancelable to left (Δ_6)*
  - $\forall a, b \in T,\ \exists c \in T^+,\ (a +^T c = b) \vee (b +^T c = a)$ — *linearity (Δ_7)*

Definition 3 (Time scale). *A* **time scale** *is any subset $\mathbb{T}$ of a time reference T such that:*

- *$\mathbb{T}$ has a minimum $m^{\mathbb{T}} \in \mathbb{T}$*
- *$\forall t \in T,\ \mathbb{T}_{t+} = \{t' \in \mathbb{T} \mid t \prec t'\}$ has a minimum called $succ^{\mathbb{T}}(t)$*

– $\forall t \in T$, *when* $m^{\mathbb{T}} \prec t$, *the set* $\mathbb{T}_{t-} = \{t' \in \mathbb{T} \mid t' \prec t\}$ *has a maximum called* $pred^{\mathbb{T}}(t)$
– *the principle of induction*[2] *is true on* $\mathbb{T}$.

The set of all time scales on T *is noted* $Ts(T)$.

1.2 Dataflows

Together with this unified definition of time, we need a definition of data that allows to handle the heterogeneity of data among real-life systems. We rely on the previous definitions to describe data carried by dataflows.

Definition 4 (ϵ-alphabet). *A set* D *is an* ***ϵ-alphabet*** *if* $\epsilon \in D$. *For any set* B, *we can define an* ϵ*-alphabet by* $\overline{B} = B \cup \{\epsilon\}$.

Definition 5 (System dataset). *A* ***system dataset****, or* ***dataset****, is a pair* $\mathcal{D} = (D, \mathcal{B})$ *such that:*

– D *is an* ϵ*-alphabet*
– $\mathcal{B}$, *called* ***data behavior****, is a pair* (r, w) *with* $r : D \to D$ *and* $w : D \times D \to D$ *such that*[3]*:*
  - $r(\epsilon) = \epsilon$ *(R1)*
  - $r\big(r(d)\big) = r(d)$ *(R2)*
  - $r\big(w(d, d')\big) = r(d')$ *(R3)*
  - $w\big(r(d'), d\big) = d$ *(W1)*
  - $w\big(w(d, d'), r(d')\big) = w(d, d')$ *(W2)*

Definition 6 (Dataflow). *Let* $\mathbb{T}$ *be a time scale. A* **dataflow** *over* $(\mathcal{D}, \mathbb{T})$ *is a mapping* $X : \mathbb{T} \to D$.

Definition 7 (Sets of dataflows). *The set of all dataflows over* $(\mathcal{D}, \mathbb{T})$ *is noted* $\mathcal{D}^{\mathbb{T}}$. *The set of all dataflows over* $\mathcal{D}$ *with any possible time scale on time reference* T *is noted* $\mathcal{D}^T = \bigcup_{\mathbb{T} \in Ts(T)} \mathcal{D}^{\mathbb{T}}$.

1.3 Systems and integration operators

Given the previous definitions, we are now able to give a mathematical definition of systems. Informally, our definition is very similar to timed Mealy machines with two important differences: the set of states may be infinite and the transfer function transforms dataflows. The key point is to see those systems as black boxes that just behave the way they are supposed to.

Definition 8 (System). *A* ***system*** *is a 7-tuple* $\int = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ *where:*

[2] For $A \subset \mathbb{T}$, $\big(m^{\mathbb{T}} \in A \;\&\; \forall t \in A, succ^{\mathbb{T}}(t) \in A\big) \Rightarrow A = \mathbb{T}$.
[3] These axioms give a relevant semantics and are necessary to define consistent projections of dataflows on time scales.

– $\mathbb{T}$ *is a time scale,*
– X, Y *are input and output datasets,*
– Q *is a nonempty* ϵ*-alphabet* [4] *of states,*
– q_0 *is an element of* Q*, called initial state,*
– $\mathcal{F} : X \times Q \times \mathbb{T} \to Y$ *describes a functional behavior,*
– $\delta : X \times Q \times \mathbb{T} \to Q$ *describes a state behavior.*

Figure 1 illustrates this definition.

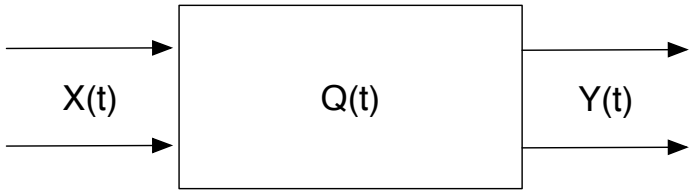
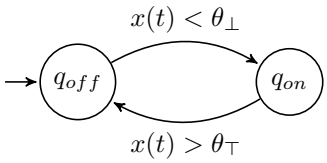


Fig. 1. Illustration of a system

Example 1. A very basic radiator with an internal thermostat placed in a room can be modeled as a system $S = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ with:

– $T \sim \mathbb{N}$
– $X =$ room temperature $\sim R$
– $Y = \{heat, nothing\}$
– $Q = \{q_{on}, q_{off}\}$
– $q_0 = q_{off}$
– $\mathcal{F}(x(t), q(t)) = \begin{cases} heat \text{ if } q(t) = q_{on} \\ \varnothing \text{ otherwise} \end{cases}$
– δ is a follows:



It is important to understand here that at each time instant of the time scale, the state of the system changes instantly and before $\mathcal{F}$ computes the resulting output. At $m^{\mathbb{T}_s}$, the beginning of the time scale, the state of the system is q_0. But as soon as the first input data arrives, at $succ^{\mathbb{T}}(m^{\mathbb{T}_s})$, the state of $\int$ changes so that the functional behaviour ignores q_0. Figure 2 illustrates this behaviour and we give the following formal definition for timed executions of systems.

[4] Defining Q as an ϵ-alphabet (therefore containing ϵ) and not just as a set will make it possible to define a dataflow of states, which is convenient.

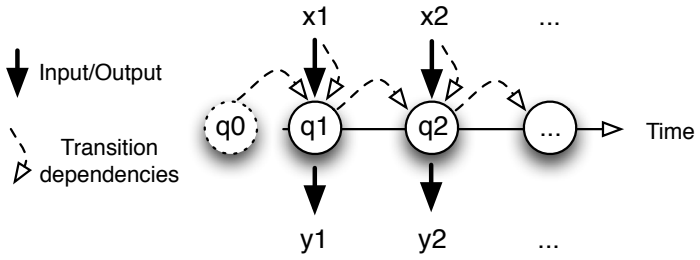


Fig. 2. Transitions of a system throughout its time scale

Definition 9 (Execution of a system). *Let $\int = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system. Let $In \in X^T$ be an input dataflow for $\int$ and $\tilde{In} = In_{\mathbb{T}}$. The **execution of $\int$ on the input dataflow** In is the 3-tuple (In, S, Out) where:*

- *$S \in Q^{\mathbb{T}}$ is recursively defined by:*
  - *$S(m^{\mathbb{T}}) = \delta\big(\tilde{In}(m^{\mathbb{T}}), q_0, m^{\mathbb{T}}\big)$*
  - *$\forall t \in \mathbb{T},\ S(t^+) = \delta\big(\tilde{In}(t^+), S(t), t^+\big)$ where $t^+ = succ^{\mathbb{T}}(t)$*
- *$Out \in Y^{\mathbb{T}}$ is defined by:*
  - *$Out(m^{\mathbb{T}}) = \mathcal{F}\big(\tilde{In}(m^{\mathbb{T}}), q_0, m^{\mathbb{T}}\big)$*
  - *$\forall t \in \mathbb{T},\ Out(t^+) = \mathcal{F}\big(\tilde{In}(t^+), S(t), t^+\big)$ where $t^+ = succ^{\mathbb{T}}(t)$*

In, S and Out are respectively input, state and output dataflows.
We note $exec(\int)$ the set of possible executions of $\int$.

Definition 10 (Product of systems on a time scale). *Let $(\int^i)_i = (\mathbb{T}, X_i, Y_i, Q_i, q_{0i}, \mathcal{F}_i, \delta_i)_i$ be n systems of time scale $\mathbb{T}$. The **product** $\int^1 \otimes \cdots \otimes \int^n$ is the system $\big(\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta\big)$ where:*

- *$X = X_1 \otimes \cdots \otimes X_n$ and $Y = Y_1 \otimes \cdots \otimes Y_n$*
- *$Q = Q_1 \times \cdots \times Q_n$ and $q_0 = (q_{01}, \ldots, q_{0n}) = q_{01\ldots n}$*
- *$\mathcal{F}\big(x_{1\ldots n}, q_{1\ldots n}, t\big) = \big(\mathcal{F}_1(x_1, q_1, t), \ldots, \mathcal{F}_n(x_1, q_1, t)\big)$*
- *$\delta\big(x_{1\ldots n}, q_{1\ldots n}, t\big) = \big(\delta_1(x_1, q_1, t), \ldots, \delta_n(x_1, q_1, t)\big)$*

Remark 1. This definition can be extended to systems that do not share a time scale, thanks to a technical operator introduced in [3]. This operator builds a timed-extension of a system, which is a system that has an equivalent input-output behaviour as the original system, but on a wider time scale. Figure 3 illustrates this idea.

Definition 11 (Feedback of a system). *Let $\int = \big(\mathbb{T}, (D \times In, \mathcal{I}), (D \times Out, \mathcal{O}), Q, q_0, \mathcal{F}, \delta\big)$ be a system such that there is no instantaneous influence of dataset D from*

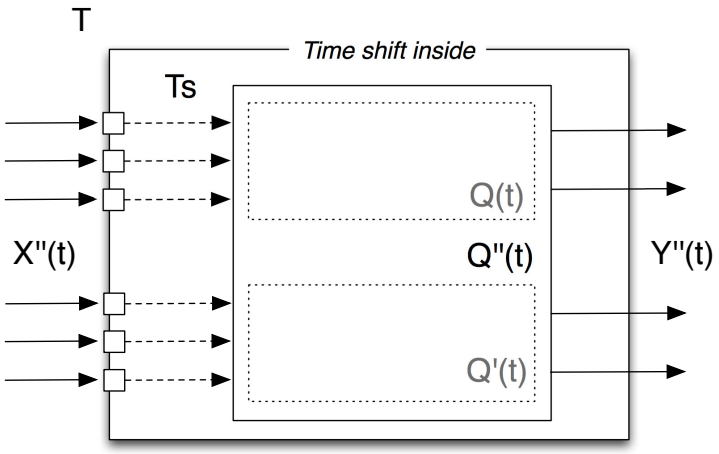


Fig. 3. Product of systems

the input to the output[5]*, i.e.* $\forall t \in \mathbb{T}, \forall x \in In,\ \forall d \in D,\ \mathcal{F}\big((d,x),q,t\big)_D = \mathcal{F}\big((\epsilon,x),q,t\big)_D$*. The* ***feedback of*** D ***in*** $\int$ *is the system* $\int_{FB(D)} = \big(\mathbb{T},(In,\mathcal{I}'),(Out,\mathcal{O}'),Q,q_0,\mathcal{F}',\delta'\big)$ *where:*

- $\mathcal{I}'$ *is the restriction of* $\mathcal{I}$ *to* In*, and* $\mathcal{O}'$ *is the restriction of* $\mathcal{O}$ *to* Out
- $\mathcal{F}'(x \in In, q \in Q, t) = \mathcal{F}\big((d_{x,q,t},x),q,t\big)_{Out}$
- $\delta'(x \in In, q \in Q, t) = \delta\big((d_{x,q,t},x),q,t\big)$

where $d_{x,q,t}$ *stands for* $\mathcal{F}\big((\epsilon,x),q,t\big)_D$*.*
Figure 4

Definition 12 (Abstraction of a transfer function). *Let* $F : X^T \to Y^{\mathbb{T}}$ *be a transfer function. Let* $A_x : X^{\mathbb{T}} \to {Y_a}^{\mathbb{T}_a}$ *be an abstraction for input dataflows and* $A_y : Y^{\mathbb{T}} \to {Y_a}^{\mathbb{T}_a}$ *an abstraction for output dataflows. The* ***abstraction of*** F *for input and output abstractions* (A_x, A_y) *with events* $\mathcal{E}$ *is the new transfer function*

$$F_a : (X_a \otimes \mathcal{E})^T \to {Y_a}^{\mathbb{T}_a}$$

defined by:

$$\forall x \in X^T,\ \exists e \in \mathcal{E}^{\mathbb{T}_a},\ F_a\big(A_x(x_{\mathbb{T}}) \otimes e\big) = A_y\big(F(x)\big)$$

Figure 5 illustrates this definition.

[5] As explained informally in [3], this condition makes it possible to define a unique feedback, i.e. without having to solve a fixed point equation that could lead to zero or multiple solutions

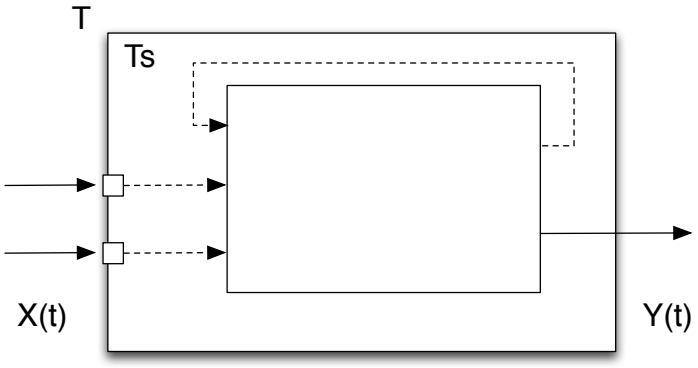


Fig. 4. Feedback of a system

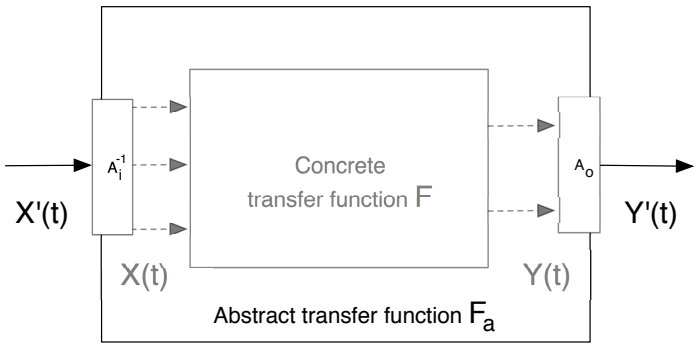


Fig. 5. Abstraction of a transfer function

Definition 13 (Abstraction of a system). *Let* $\int = \left(\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta\right)$ *be a system.* $\int' = \left(\mathbb{T}_a, X_a \otimes \mathcal{E}, Y_a, Q_a, q_{a0}, \mathcal{F}_a, \delta_a\right)$ *is an abstraction of* $\int$ *for input and output abstractions* (A_x, A_y) *if, and only if:* $\exists A_q : Q^{\mathbb{T}} \rightarrow {Q_a}^{\mathbb{T}_a}$*, for all execution* (x, q, y) *of* $\int$*,* $\exists E \in \mathcal{E}^{\mathbb{T}_a}$*,* $\left(A_x(x_{\mathbb{T}}) \otimes E, A_q(q), A_y(y)\right)$ *is an execution of* $\int'$*. Conversely,* $\int'$ *is a concretization of the system* $\int$*.*

A system captures the behavior of a system that can be observed (functional and states behavior, called together *systemic behavior*). From this definition, we can start expressing behaviour properties.

2 Systemic behaviour properties: a formal semantics

The goal of this section is to be able to describe the behaviour of such a system, in order to express properties and constraints on it. To do so, we will define a

semantics of systems and provide a formal definition of "properties". The idea here is that systems are described by executions, so we must use timed properties. We will first describe our property syntax language, then our property semantics.

Definition 14 (System property formulas). *Let $\int = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system.*
The set $P(\int)$ of property formulas over $\int$, similar to LTL[6]*, is inductively defined as follows.*
Here are the atomic formulas, where $(x, q, y) \in X \times Q \times Y$:

- *input(x), which means that the current input of $\int$ has to be x*
- *istate(q), which means that the current state of $\int$ has to be q*
- *ouput(y), which means that the current output of $\int$ has to be y*

And here are the operators, where $(\phi, \phi_1, \phi_2) \in P(\int)^3$:

- $\neg\phi$
- $\phi_1 \wedge \phi_2$
- $\bigcirc\phi$*, which means that ϕ has to hold at the next state of the execution*
- $\phi_1 \mathcal{U} \phi_2$*, which means that ϕ_2 eventually has to hold and until it does, ϕ_1 has to hold (ϕ_1 can hold further)*

Definition 15 (Other system property formulas). *Let $\int = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system.*
Let $(\phi, \phi_1, \phi_2) \in P(\int)^3$.
The previously defined language $P(\int)$ can be extended with the following operators:

- $\top$
- $\bot$
- $\phi_1 \vee \phi_2$
- $\phi_1 \Rightarrow \phi_2$ [7]
- $\Diamond\phi$*, which means that ϕ has to hold now or in a future state of the execution*
- $\Box\phi$*, which means that ϕ has to hold for the entire subsequent execution*
- $\phi_1 \mathcal{R} \phi_2$*, which means that ϕ_1 has to hold until and including a state when ϕ_2 holds, which is not forced to happen if ϕ_1 holds forever*

$P(\int)$ is the *syntax* of our properties language. It gives us a way to express properties on executions of $\int$. We are now able to define our semantics. The following rules describe the satisfaction of $P(\int)$ formulas using only the first set of operators, but such rules could be easily extended to the extended set of operators.

Definition 16 (system property satisfaction). *Let $\int$ be a system.*
*The **satisfaction** of a $P(\int)$ formula ϕ by an execution e of $\int$, written $e \vDash \phi$, is defined according to the following rules:*

[6] see [1]
[7] We can also accept $\phi_1 \otimes \phi_2$ or any other "usual" operators

$$\frac{}{\big((x_0, \_, \_), \dots\big) \vDash input(x_0)}\ [AI] \qquad \frac{}{\big((\_, q_0, \_), \dots\big) \vDash istate(q_0)}\ [AS]$$

$$\frac{}{\big((\_, \_, y_0), \dots\big) \vDash output(y_0)}\ [AO] \qquad \frac{[e \vDash \phi] \Rightarrow \bot}{e \vDash \neg\phi}\ [AN]$$

$$\frac{e \vDash \phi_1 \qquad e \vDash \phi_2}{e \vDash \phi_1 \wedge \phi_2}\ [AW] \qquad \frac{\big(e_1, e_2, \dots\big) \vDash \phi}{\big(\_, e_1, e_2, \dots\big) \vDash \bigcirc\phi}\ [AC]$$

$$\frac{\exists i \leq 0, \big[\big((e_i, e_{i+1}, \dots) \vDash \phi_2\big) \wedge \big(\forall k \in \{0, \dots, i\}, (e_k, e_{k+1}, \dots) \vDash \phi_1\big)\big]}{\big(e_0, e_1, \dots\big) \vDash \phi_1 \mathcal{U} \phi_2}\ [AU]$$

Definition 17 (system behaviour constraint). *Let $\int$ be a system. Let ϕ be a $P(\int)$ formula.*
We say that $\int$ satisfies ϕ, which is noted $\int \vDash \phi$, iif

$$\forall e \in exec(\int), e \vDash \phi$$

In this case, ϕ is said to be a ***behaviour constraint*** *on $\int$.*

Example 2. Unsing our previous example 1, with $\theta_\perp = 15$ and $\theta_\top = 20$, here are some behaviour constraints one would want to express on the system:

- $\int \vDash \neg\Diamond\big(istate(q_{off}) \wedge output(heat)\big)$
- $\int \vDash \neg\Diamond\big(input(10) \wedge \bigcirc istate(q_{off})\big)$

or, even more generally:

- $\forall \theta > \theta_\top, \int \vDash \neg\Diamond\big(input(\theta) \wedge \bigcirc istate(q_{on})\big)$

3 Computation rules over behaviour constraints

We give here a minimalist set of computation rules to establish proofs about system behaviours. A more advanced set of rules might be needed to ease such proofs.

Proposition 1 (Product of two systems). *Let $\int_1 = (\mathbb{T}, X_1, Y_1, Q_1, \_, \_, \_)$ and $\int_2 = (\mathbb{T}, X_2, Y_2, Q_2, \_, \_, \_)$ be two systems.*
Let $(x_1, x_2) \in X_1 \times X_2$, $(y_1, y_2) \in Y_1 \times Y_2$ and $(q_1, q_2) \in Q_1 \times Q_2$.

$$\frac{\int_1 \vDash input(x_1) \qquad \int_2 \vDash input(x_2)}{\int_1 \otimes \int_2 \vDash input(x_1 \otimes x_2)}\ [PI]$$

$$\frac{\int_1 \vDash output(y_1) \qquad \int_2 \vDash output(y_2)}{\int_1 \otimes \int_2 \vDash output(x_1 \otimes x_2)}\ [PO]$$

$$\frac{\int_1 \vDash istate(q_1) \qquad \int_2 \vDash istate(q_2)}{\int_1 \otimes \int_2 \vDash istate((q_1, q_2))}\ [PS]$$

Definition 18 (Composition of two systems). *Let* $\int_1 = (\mathbb{T}, X_1, Y_1, Q_1, \_, \_, \_)$ *and* $\int_2 = (\mathbb{T}, X_2, Y_2, Q_2, \_, \_, \_)$ *be two systems such that* $Y_1 = X_2$.
We note $\int_2 \circ \int_1$ *the composition of* $\int_1$ *and* $\int_2$*, obtained by "pluging" the output of* $\int_1$ *to the input of* $\int_2$.

Proposition 2 (Composition of two systems). *Let* $\int_1 = (\mathbb{T}_1, X_1, Y_1, Q_1, \_, \_, \_)$ *and* $\int_2 = (\mathbb{T}_2, X_2, Y_2, Q_2, \_, \_, \_)$ *be two systems such that* $\mathbb{T}_1 = \mathbb{T}_2$ *and* $Y_1 = X_2$.
Let $(x, y, q_1, q_2) \in X_1 \times Y_2 \times Q_1 \times Q_2)$.

$$\frac{\int_1 \vDash input(x)}{\int_2 \circ \int_1 \vDash input(x)}\ [WI]$$

$$\frac{\int_2 \vDash output(y)}{\int_2 \circ \int_1 \vDash output(y)}\ [WO]$$

$$\frac{\int_1 \vDash istate(q_1) \qquad \int_2 \vDash istate(q_2)}{\int_2 \circ \int_1 \vDash istate((q_1, q_2))}\ [WS]$$

Conclusion

In this work we built a semantics of systems and provided a formal definition of "properties" as timed behaviour constraints. The next step is to build a more advanced refinement computation language that could let modelers obtain a system from a set of such constraints.

References

1. LTL. `http://en.wikipedia.org/wiki/Linear_temporal_logic`. (last accessed on 31/10/2012).
2. O. Bournez and M.-L. Campagnolo. A survey on continuous time computations. *CoRR*, abs/0907.3117, 2009.
3. B. Golden, M. Aiguier, and D. Krob. Modeling of complex systems ii: A minimalist and unified semantics for heterogeneous integrated systems. *Applied Mathematics and Computation*, 218(16):8039–8055, 2012.
4. B. Golden and Y. Hourdel. A minimalist formal framework for systems design. 2013.
5. T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–, Washington, DC, USA, 1996. IEEE Computer Society.
6. D. Krob. Éléments de systémique - architecture des systèmes. 2012.