An Institution of Hybrid Systems

Hugo Lourenço and Amílcar Sernadas

CMA, Dep. Matemática,
IST, Av. Rovisco Pais, Lisboa, Portugal
{hlouren,acs}@math.ist.utl.pt

Abstract. Hybrid systems are systems that intermix discrete and continuous behavior. Interest in hybrid systems has grown in recent years [3], mainly because of their relation with embedded systems (systems that interact with the *continuously* changing real-world). In order to deal with complex problems, compositional specification and verification methods are needed. Using the specification language of HyTech[5] as a starting point, an institution of hybrid systems is established. Generalizing the results in [9], free aggregation, interconnection and abstraction of hybrid systems are presented as categorial constructions within the proposed institution, at both the specification and the model levels. The HyTech parallel composition constructor is shown to be a particular case of interconnection. Compositional model checking of a complex system is shown to be possible by exploiting its categorial structure. A class of properties that can be verified with HyTech is identified.

1 Introduction

A *hybrid system* is a system that intermixes discrete and continuous components (typically a digital computer interacting with the analog physical world). Hybrid system are everywhere: they arise in air traffic control, automobiles, robotics, consumer electronics (e.g. VCR, microwave oven, heater)...

Hybrid systems exhibit a phased evolution: discrete steps alternate with continuous evolution. We can describe such a system in terms of its (real-valued) attributes, modes of operation and action symbols. Each mode of operation describes how the attributes evolve and the actions label discrete transitions.

While a discrete transition does not occur the hybrid system remains in the same mode of operation and the attributes values change continuously according to the evolution law of the mode of operation. When a discrete transition takes place both the mode of operation and the attributes values may change. Each discrete transition is (possibly) labelled with an action symbol. Action symbols are useful for interconnecting hybrid systems.

* This work was partially supported by *Fundação para a Ciência e a Tecnologia*, the PRAXIS XXI Projects PCEX/P/MAT/46/96 ACL, PRAXIS/P/MAT/10002/1998 ProbLog and 2/2.1/TIT/1658/95 LogComp, as well as by the ESPRIT IV Working Groups 22704 ASPIRE and 23531 FIREworks.

As an example, consider an heater. A possible attribute is the current temperature. We can consider an "heating" mode of operation during which the heater is on and the temperature is continuously increasing, and a "cooling" mode of operation during which the heater is off and temperature decreases. Also, we can act on the heater by turning it on or off. A mode of operation can also be seen as a particular valuation of a set of boolean attributes. In the previous example, we would say that the heater is in an "heating" mode whenever "heating=true". At every instant, the state of the heater is described by a pair of values: the temperature and the current mode of operation.

Interest in hybrid systems has grown in recent years, mainly because of their relation with embedded systems. These systems often arise in safety critical situations, so formal verification methods for hybrid systems have been the subject of intense research [1, 2, 4, 6].

However, in order to deal with real world problems compositional verification methods are known to be needed. We thus seek to construct an institution of hybrid system and characterize categorially some forms of combining hybrid systems. Namely, we seek to characterize free aggregation as a product, restriction as a cartesian lifting and abstraction as a cocartesian lifting. Interconnection is seen as a two-step process: free aggregation followed by restriction. These constructions extend to hybrid systems the well known definitions of aggregation and interconnection of transition systems as presented in [10].

The paper is organized as follows. First, we introduce the institution of hybrid computations, from which we build the institution of hybrid systems (preliminary work on the semantic component of the institution can be found in [8, 7]). Then we present the already referred forms of combining hybrid systems. Finally, we look at model checking procedures for hybrid systems and show how the categorial structure of a complex system can be used to ease the verification process.

We assume familiarity with some basic concepts of category theory (namely, (co)products and (co)cartesian morphisms - see [9]).

2 The Institution of Hybrid Computations

In this section we establish the institution $I^b = \langle Sig, Sen, Int^b, \Vdash^b \rangle$ of hybrid computations. In the sequel, *Fin* denotes the category of finite sets and maps and *PFin* denotes the category of finite sets and partial maps.

2.1 The Signature Category

A hybrid system is described in terms of its attributes, modes of operation and action symbols. We thus have the objects for *Sig*. As for the signature morphisms, lets look at what we want to get as signature coproducts.

If we take the signatures of two systems, we want their coproduct to correspond to the signature of the free aggregation of the two systems. So we want to have all the attributes together but keeping the distinction between attributes with the same name, a mode of operation will be a pair of modes of operation, and an action of the resulting system can either be an action from only one of the components or a pair of actions (again, actions with the same name are kept distinct).

Definition 2.1. *Sig* is the category $Fin \times Fin^{op} \times PFin^{op}$.

Given a signature $\langle X, V, A\rangle$, X is a set of real-valued attributes, V is a set of vertices and A is a set of action symbols. Each vertex corresponds to a mode of operation. The action symbols are used to label discrete transitions. Together, a vertex $v \in V$ and a valuation $\alpha \in \mathbb{R}^X$ define a state of the hybrid system. We use St_Σ for the set $V \times \mathbb{R}^X$ (of Σ-states). A morphism $\sigma : \Sigma \to \Sigma'$ has the form $\langle \overrightarrow{\sigma}_{\rm att} : X \to X', \overleftarrow{\sigma}_{\rm vtx} : V' \to V, \overleftarrow{\sigma}_{\rm act} : A' \rightharpoonup A\rangle$.[1] For a signature op-morphism $f : \Sigma' \to \Sigma$ we reverse the arrows: $\langle \overleftarrow{f}_{\rm att} : X \to X', \overrightarrow{f}_{\rm vtx} : V' \to V, \overrightarrow{f}_{\rm act} : A' \rightharpoonup A\rangle$.

Example 2.2. Consider an heater. The heater can be either on *or* off. *When it is on, the temperature T rises according to the law $\mathbf{D}T = 4 - T/20$; when it is off, T follows $\mathbf{D}T = -T/20$. A possible signature for the heater is $\Sigma_h = \langle X_h, V_h, A_h\rangle$ with $X_h = \{T\}$, $V_h = \{\text{on}, \text{off}\}$ and $A_h = \{h, c\}$ (h = heat, c = cool).*

Example 2.3. If we hide the attribute T in the preceding system, we retain only the discrete behavior of the heater. A possible signature for this simplified system is $\Sigma_{h'} = \langle X_{h'}, V_{h'}, A_{h'}\rangle$ with $X_{h'} = \emptyset$, $V_{h'} = \{\text{on}, \text{off}\}$ and $A_{h'} = \{h, c\}$.

Example 2.4. Consider a controller for the preceding heater. For safety reasons, the heater cannot be turned on if the temperature is over 40*. Also, the heater cannot stay on for more than* 10 *time units, so this controller has a clock C to measure how long the heater has been on. If the clock reaches* 10 *without the heater being turned off in the meantime, then the controller forces the heater to turn off. A possible signature for the controller is $\Sigma_c = \langle X_c, V_c, A_c\rangle$ with $X_c = \{T, C\}$, $V_c = \{\text{on}, \text{off}\}$ and $A_c = \{h, c\}$. A simplified version c' of c can be defined as we did for the heater. In this case, $\Sigma_{c'} = \Sigma_{h'}$.*

2.2 The Semantic Functor

A hybrid system works by alternating *continuous phases* with *discrete jumps*.

Definition 2.5. Let $\Sigma = \langle X, V, A\rangle \in |Sig|$. A *$\Sigma$-phase* is a triple $p = \langle v, h, Dh\rangle$ with $v \in V$ and $h, Dh : I \to \mathbb{R}^X$, where $I \subseteq \mathbb{R}_0^+$ is a closed interval and such that (i) $0 \in I$, (ii) for each $x \in X$, $Dh^x \stackrel{\rm def}{=} \lambda t.Dh(t)(x)$ is continuous and (iii) for each $x \in X$, $h^x \stackrel{\rm def}{=} \lambda t.h(t)(x)$ satisfies $h^x(t) = h^x(0) + \int_0^t Dh^x(t)dt$.

We use I_p for denoting the interval I and P_Σ for the set of Σ-phases.

In the following definition, a range $0..n$ denotes the set $\{i \in \mathbb{N}_0 \mid i \leq n\}$ (note that $+\infty \notin 0.. + \infty$).

[1] Sometimes we omit the subscripts in σ.

Definition 2.6. Let $\Sigma = \langle X, V, A\rangle$ be a signature. A *Σ-computation* is a triple $\langle n, p, a\rangle$ with $n \in \overline{\mathbb{N}}_0$, $p : 0..n \to P_\Sigma$ and $a : 0..(n-1) \rightharpoonup A$ such that (i) if $I_{p_i} = \mathbb{R}_0^+$ then $i = n$ and (ii) $\sum length(I_{p_i}) = +\infty$.

We denote the length of I_{p_i} by d_i, and use Ξ_Σ for the set of Σ-computations.

The number of phases in the computation is given by $n+1$, the phases by p and the action symbols (possibly) labelling the end of the phases by a.[2] We can have a finite number of phases, but in that case the last phase is of infinite length. All computations are infinite in time. We usually refer to the phase p_i by $\langle v_i, h_i, Dh_i\rangle$.

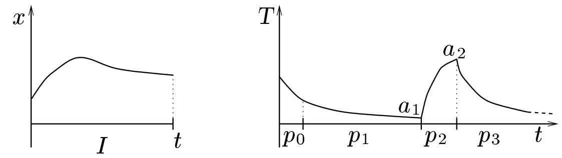

Fig. 1. A phase and a computation

Definition 2.7. A morphism $\sigma : \Sigma \to \Sigma'$ induces the maps $\overleftarrow{\sigma}_{\mathrm{asg}} : \mathbb{R}^{X'} \to \mathbb{R}^X$, $\overleftarrow{\sigma}_{\mathrm{st}} : St_{\Sigma'} \to St_\Sigma$, $\overleftarrow{\sigma}_{\mathrm{ph}} : P_{\Sigma'} \to P_\Sigma$ and $\overleftarrow{\sigma}_{\mathrm{cmp}} : \Xi_{\Sigma'} \to \Xi_\Sigma$ defined as follows:

- $\overleftarrow{\sigma}_{\mathrm{asg}}(\alpha') = \alpha' \circ \overrightarrow{\sigma}_{\mathrm{att}}$;
- $\overleftarrow{\sigma}_{\mathrm{st}}(\langle v', \alpha'\rangle) = \langle \overleftarrow{\sigma}_{\mathrm{vtx}}(v'), \overleftarrow{\sigma}_{\mathrm{asg}}(\alpha'\rangle$;
- $\overleftarrow{\sigma}_{\mathrm{ph}}(\langle v', h', Dh'\rangle) = \langle \overleftarrow{\sigma}_{\mathrm{vtx}}(v'), \overleftarrow{\sigma}_{\mathrm{asg}} \circ h', \overleftarrow{\sigma}_{\mathrm{asg}} \circ Dh'\rangle$;
- $\overleftarrow{\sigma}_{\mathrm{cmp}}(\langle n', p', a'\rangle) = \langle n', \overleftarrow{\sigma}_{\mathrm{ph}} \circ p', \overleftarrow{\sigma}_{\mathrm{act}} \circ a'\rangle$.

Prop/Definition 2.8. Let Σ, Σ' be signatures and $\xi \in \Xi_\Sigma$, $\xi' \in \Xi_{\Sigma'}$. A computation morphism $f : \xi' \to \xi$ is a signature morphism $f : \Sigma \to \Sigma'$ such that $\overrightarrow{f}_{\mathrm{cmp}}(\xi') = \xi$. Computations and their morphisms constitute a category *Comp*.

In order to establish the semantic functor $Int^b : Sig \to Cat^{op}$ we start by defining a cofibration $cS : Comp \to Sig^{op}$ that has a splitting cocleavage. Then we obtain Int^b in a canonical way (see [9] for details).

Prop/Definition 2.9. We obtain a (forgetful) functor $cS : Comp \to Sig^{op}$ by defining that $cS(\xi) = \Sigma$ for every $\xi \in \Xi_\Sigma$ and $cS(f : \xi \to \xi') = f$.

We have that cS is a cofibration: given $\sigma : \Sigma' \to \Sigma$ in Sig^{op} and $\xi' \in \Xi_{\Sigma'}$, the computation morphism $\sigma : \xi' \to \overrightarrow{\sigma}_{\mathrm{cmp}}(\xi')$ is cocartesian by cS for σ on ξ'. Moreover, cS has a splitting cocleavage.

We write $Comp_\Sigma$ for $cS^{-1}(\Sigma)$ (the fiber of *Comp* over Σ).

Prop/Definition 2.10. We obtain the functor $Int^b : Sig \to Cat^{op}$ from sS. Namely, $Int^b(\Sigma) = Comp_\Sigma$ and $Int^b(\sigma : \Sigma \to \Sigma')(\xi') = \overleftarrow{\sigma}_{\mathrm{cmp}}(\xi')$.

[2] The possibility to have discrete jumps without labels will become clear when we discuss the aggregation of hybrid systems

2.3 The Syntactic Functor

Definition 2.11. Let $\Sigma = \langle X, V, A\rangle$ be a signature. The set T_Σ of Σ-terms is inductively defined as follows:

- $q, x, \mathbf{D}x \in T_\Sigma$ provided $q \in \mathbb{Q}, x \in X$;
- $\mathbf{N}t_1, (t_1 + t_2), (t_1 \times t_2) \in T_\Sigma$ provided $t_1, t_2 \in T_\Sigma$.

The term $\mathbf{D}x$ denotes the derivative of x, and $\mathbf{N}t$ denotes the value of t at the next point in time.[3]

Definition 2.12. Let $\Sigma = \langle X, V, A\rangle$ be a signature. The set L_Σ of Σ-formulae is inductively defined as follows:

- $t_1 \leq t_2 \in L_\Sigma$ provided $t_1, t_2 \in T_\Sigma$;
- $@W, \mathbf{N}@W \in L_\Sigma$ provided $W \subseteq V$;
- $\nabla B \in L_\Sigma$ provided $B \subseteq A$;
- $(\neg\,\varphi), (\varphi \Rightarrow \psi) \in L_\Sigma$ provided $\varphi, \psi \in L_\Sigma$;
- $(\varphi \mathbf{U}^{\alpha..\beta}\psi), (\varphi \mathbf{S}^{\alpha..\beta}\psi) \in L_\Sigma$ provided $\varphi, \psi \in L_\Sigma$, $\alpha \in \mathbb{Q}_0^+$ and $\beta \in \overline{\mathbb{Q}_0^+}$.

A formula $@W$ is intended to mean that the current mode of operation (or vertex) is in W (we omit the braces when W is a singleton), whereas $\mathbf{N}@W$ means that at the next point in time the vertex will be in W. A formula ∇B states that an action in B is occurring. In order to be able to specify precise timing constraints, the "until" ($\mathbf{U}$) and "since" ($\mathbf{S}$) operators have a time bound. The usual operators arise if we use $0..+\infty$ as the time bound.

We use the usual abbreviations: $\mathbf{F}\varphi \stackrel{\text{def}}{=} (\varphi \vee \neg\,\varphi)\ \mathbf{U}\ \varphi$ for "sometime in the future", $\mathbf{G}\varphi \stackrel{\text{def}}{=} \neg(\mathbf{F}(\neg\,\varphi))$ for "always in the future", etc. The following formula is true precisely at the points where the system is behaving continuously:

$$\mathbf{Cnt}_\Sigma \stackrel{\text{def}}{=} (\bigwedge_{x\in X} x = \mathbf{N}x) \wedge (\bigwedge_{x\in X} \mathbf{D}x = \mathbf{N}\mathbf{D}x) \wedge (\bigwedge_{v\in V} @v \Leftrightarrow \mathbf{N}@v)$$

Example 2.13. Recall the systems described in examples 2.2, 2.3 and 2.4. Let us now see a possible specification of such systems:

- $\Gamma_h = \{\ @on \Rightarrow \mathbf{D}T = 4 - T/20,\quad @off \Rightarrow \mathbf{D}T = -T/20,$
  $\nabla h \Rightarrow (@off \wedge \mathbf{N}@on \wedge \mathbf{N}T = T),\quad (\neg\,\nabla A_h) \Rightarrow \mathbf{Cnt}_{\Sigma_h},$
  $\nabla c \Rightarrow (@on \wedge \mathbf{N}@off \wedge \mathbf{N}T = T)\}$
- $\Gamma_{h'} = \{\ \nabla h \Rightarrow (@off \wedge \mathbf{N}@on),\quad (\neg\,\nabla A_{h'}) \Rightarrow \mathbf{Cnt}_{\Sigma_{h'}},$
  $\nabla c \Rightarrow (@on \wedge \mathbf{N}@off)\}$
- $\Gamma_c = \{\ @on \Rightarrow \mathbf{D}C = 1,\quad @off \Rightarrow \mathbf{D}C = 0,$
  $\nabla h \Rightarrow (@off \wedge T \leq 40 \wedge \mathbf{N}@on \wedge \mathbf{N}C = 0),\ \nabla c \Rightarrow (@on \wedge \mathbf{N}@off),$
  $(\neg\,\nabla A_c) \Rightarrow \mathbf{Cnt}_{\Sigma_c},\quad @on \Rightarrow (0 \leq C \leq 10)\}$

A signature morphism $\sigma : \Sigma \to \Sigma'$ induces the maps $\overrightarrow{\sigma}_{\text{trm}} : T_\Sigma \to T_{\Sigma'}$ and $\overrightarrow{\sigma}_{\text{lng}} : L_\Sigma \to L_{\Sigma'}$ defined as expected. Namely, $\overrightarrow{\sigma}_{\text{lng}}(@W) = @\overleftarrow{\sigma}_{\text{vtx}}^{-1}(W)$, $\overrightarrow{\sigma}_{\text{lng}}(\mathbf{N}@W) = \mathbf{N}@\overleftarrow{\sigma}_{\text{vtx}}^{-1}(W)$ and $\overrightarrow{\sigma}_{\text{lng}}(\nabla B) = \nabla\overleftarrow{\sigma}_{\text{act}}^{-1}(B)$. We thus have:

Prop/Definition 2.14. The (syntactic) functor $\mathit{Sen} : \mathit{Sig} \to \mathit{Set}$ is defined by $\mathit{Sen}(\Sigma) = L_\Sigma$ and $\mathit{Sen}(\sigma : \Sigma \to \Sigma') = \overrightarrow{\sigma}_{\text{lng}}$.

[3] Because we are dealing with a continuous time domain, it may seem strange to talk about a "next point in time". Later on, the meaning will become clear.

2.4 The Satisfaction Relation

In order to define the satisfaction relation, we start by defining the term interpretation map. Terms are interpreted at "points" in computations.

Definition 2.15. Let $\xi = \langle n, p, a\rangle$ be a computation. The *point space* associated with ξ is the set $spa_\xi = \{\langle i, t\rangle \in \mathbb{N}_0 \times \mathbb{R}_0^+ \mid i \leq n, 0 \leq t \leq d_i\}$. For $\langle i, t\rangle \in spa_\xi$, i corresponds to number of the phase and t to time elapsed since the beginning of the phase. We obtain a total ordering over spa_ξ by defining that $\langle i, t\rangle \leq \langle i', t'\rangle$ iff $i < i'$ or ($i = i'$ and $t \leq t'$), and write $\xi(\langle i, t\rangle)$ for the Σ-state $\langle v_i, h_i(t)\rangle$.

Also, a function $T_\xi : spa_\xi \to \mathbb{R}_0^+$ mapping points in spa_ξ to the real-time axis can be defined by $T_\xi(\langle i, t\rangle) = t + \sum_{j<i} d_j$.

Definition 2.16. Let $\Sigma = \langle X, V, A\rangle$ be a signature, $\xi \in \Xi_\Sigma$ a computation and $\langle i, t\rangle \in spa_\xi$. Then the *term interpretation map* $[\![\_]\!]_\Sigma^{\xi,\langle i,t\rangle} : T_\Sigma \to \mathbb{R}$, mapping Σ-terms to their value, is inductively defined as follows:

– $[\![x]\!]_\Sigma^{\xi,\langle i,t\rangle} = h_i^x(t)$; – $[\![\mathbf{N}t]\!]_\Sigma^{\xi,\langle i,t\rangle} = \begin{cases} [\![t]\!]_\Sigma^{\xi,\langle i+1,0\rangle} & \text{pt } t = d_i \\ [\![t]\!]_\Sigma^{\xi,\langle i,t\rangle} & \text{otherwise} \end{cases}$;

– $[\![\mathbf{D}x]\!]_\Sigma^{\xi,\langle i,t\rangle} = Dh_i^x(t)$; – etc

Note the interpretation of the term $\mathbf{N}t$. Here it is clear what we mean by "next point in time": if we are at the end of a phase, then the next point in time is the first point of the *next* phase; otherwise, it is just the current point.

Definition 2.17. Let $\Sigma = \langle X, V, A\rangle$ be a signature. The satisfaction relation $\Vdash^b \subseteq \Xi_\Sigma \times L_\Sigma$ is defined by $\xi \Vdash_\Sigma^b \varphi$ iff $\xi, \langle i, t\rangle \Vdash_\Sigma^b \varphi$ for all $\langle i, t\rangle \in spa_\xi$ where:

– $\xi, \langle i, t\rangle \Vdash_\Sigma t_1 \leq t_2$ iff $[\![t_1]\!]_\Sigma^{\xi,\langle i,t\rangle} \leq [\![t_2]\!]_\Sigma^{\xi,\langle i,t\rangle}$
– $\xi, \langle i, t\rangle \Vdash_\Sigma @W$ iff $v_i \in W$
– $\xi, \langle i, t\rangle \Vdash_\Sigma \mathbf{N}@W$ iff $(t = d_i, v_{i+1} \in W)$ or $(t < d_i, v_i \in W)$
– $\xi, \langle i, t\rangle \Vdash_\Sigma \nabla B$ iff $t = d_i$, a_i is defined and $a_i \in B$
– $\xi, \langle i, t\rangle \Vdash_\Sigma (\varphi \mathbf{U}^{\alpha..\beta} \psi)$ iff for some $\langle i', t'\rangle \geq \langle i, t\rangle$ $T_\xi(\langle i', t'\rangle) - T_\xi(\langle i, t\rangle) \in [\alpha, \beta]$, $\xi, \langle i', t'\rangle \Vdash_\Sigma \psi$ and $\xi, \langle i'', t''\rangle \Vdash_\Sigma \varphi$ for all $\langle i, t\rangle \leq \langle i'', t''\rangle < \langle i', t'\rangle$

2.5 The Satisfaction Condition

Proposition 2.18. Let $\sigma : \Sigma \to \Sigma'$ be a signature morphism, $\xi' \in \Xi_{\Sigma'}$ and $\varphi \in L_\Sigma$. Then $\xi' \Vdash_{\Sigma'}^b \overrightarrow{\sigma}_{\text{lng}}(\varphi)$ iff $\overleftarrow{\sigma}_{\text{cmp}}(\xi') \Vdash_\Sigma^b \varphi$.

3 The Institution of Hybrid Systems

Capitalizing on the previous institution I^b, we build the institution of hybrid systems, $I = \langle Sig, Sen, Int, \Vdash\rangle$.

We define a hybrid systems to be a set of computations (with some additional properties). Thus I is easily obtained: the Sig category and the Sen functor are just as in I^b; the Int functor is built upon Int^b; the satisfaction relation $\Vdash$ is just an extension of $\Vdash^b$; finally, the satisfaction condition is a consequence of the satisfaction condition for I^b.

3.1 The Semantic Functor

A hybrid system consists of a set of computations. However, not every set of computations will do, because our logic cannot distinguish every two sets of computations. We thus start by considering a notion of *pre-computation system* that is then refined into the desired notion of *computation system*.

Definition 3.1. Let Σ be a signature. A *Σ-pre-computation system* is a pair $\langle \Sigma, \Xi \rangle$ where $\Xi \subseteq \Xi_\Sigma$.

Given $\Xi \subseteq \Xi_\Sigma$, we can determine the set of formulae satisfied by Ξ. Reciprocally, given $\Gamma \subseteq L_\Sigma$, we can determine the set of computations that satisfy Γ. These notions are captured in the following definition where, as usual, we define that $\Xi \Vdash^b_\Sigma \varphi$ iff $\xi \Vdash^b_\Sigma \varphi$ for every $\xi \in \Xi$ and $\xi \Vdash^b_\Sigma \Gamma$ iff $\xi \Vdash^b_\Sigma \varphi$ for every $\varphi \in \Gamma$.

Definition 3.2. Let Σ be a signature. We define the maps $S_\Sigma : 2^{L_\Sigma} \to 2^{\Xi_\Sigma}$ and $T_\Sigma : 2^{\Xi_\Sigma} \to 2^{L_\Sigma}$ by $S_\Sigma(\Gamma) = \{\xi \in \Xi_\Sigma \mid \xi \Vdash^b_\Sigma \Gamma\}$ and $T_\Sigma(\Xi) = \{\varphi \in L_\Sigma \mid \Xi \Vdash^b_\Sigma \varphi\}$. Given $\Xi \subseteq \Xi_\Sigma$, we write $\Xi^\equiv$ for $S_\Sigma(T_\Sigma(\Xi))$. Thus $\Xi^\equiv$ consists of all computations that satisfy the same formulae that are satisfied by Ξ.

Definition 3.3. Let Σ be a signature. A *Σ-computation system* is a Σ-pre-computation system $\langle \Sigma, \Xi \rangle$ such that $\Xi = \Xi^\equiv$.

We denote by $|Sys_\Sigma|$ the set of Σ-computation systems.

Prop/Definition 3.4. A computation system morphism $f : \langle \Sigma', \Xi' \rangle \to \langle \Sigma, \Xi \rangle$ is a signature morphism $f : \Sigma \to \Sigma'$ such that $\overrightarrow{f}_{\text{cmp}}(\Xi') \subseteq \Xi$. Computation systems and their morphisms constitute a category Sys.

A signature morphism $\sigma : \Sigma \to \Sigma'$ induces a map $\overleftarrow{\sigma}_{\text{cs}} : |Sys_{\Sigma'}| \to |Sys_\Sigma|$ defined by $\overleftarrow{\sigma}_{\text{cs}}(\langle \Sigma', \Xi' \rangle) = \langle \Sigma, \overleftarrow{\sigma}_{\text{cmp}}(\Xi') \rangle$.

Prop/Definition 3.5. We obtain a (forgetful) functor $sS : Sys \to Sig^{op}$ by defining that $sS(\langle \Sigma, \Xi \rangle) = \Sigma$ and $sS(f : s \to s') = f$.

We have that sS is a cofibration: given $\sigma : \Sigma' \to \Sigma$ in Sig^{op} and s' in $|Sys_{\Sigma'}|$, the computation system morphism $\sigma : s' \to \overrightarrow{\sigma}_{\text{cs}}(s')$ is cocartesian by sS for σ on s'. Moreover, sS has a splitting cocleavage.

We write Sys_Σ for $sS^{-1}(\Sigma)$ (the fiber of Sys over Σ).

Prop/Definition 3.6. The functor $Int : Sig \to Cat^{op}$ is obtained from sS. Namely, $Int(\Sigma) = Sys_\Sigma$ and $Int(\sigma : \Sigma \to \Sigma')(s') = \overleftarrow{\sigma}_{\text{cs}}(s')$.

3.2 The Satisfaction Relation

Definition 3.7. Let Σ be a signature. The relation $\Vdash_\Sigma \subseteq Sys_\Sigma \times L_\Sigma$ is defined by $\langle \Sigma, \Xi \rangle \Vdash_\Sigma \varphi$ iff $\Xi \Vdash^b_\Sigma \varphi$.

3.3 The Satisfaction Condition

Proposition 3.8. For every $\sigma : \Sigma \to \Sigma'$, $s' \in |Sys_{\Sigma'}|$ and $\varphi \in L_\Sigma$, $\overleftarrow{\sigma}_{\mathrm{cs}}(s') \Vdash_\Sigma \varphi$ iff $s' \Vdash_{\Sigma'} \overrightarrow{\sigma}_{\mathrm{lng}}(\varphi)$.

Corollary 3.9. System morphisms reflect properties, that is, for $f : s' \to s$ in *Sys*, with $s \in |Sys_\Sigma|$, and $\varphi \in L_\Sigma$, if $s \Vdash_\Sigma \varphi$ then $s' \Vdash_{\Sigma'} \overleftarrow{f}_{\mathrm{lng}}(\varphi)$.

4 Theories versus Interpretation Structures

Let Σ be a signature, $\Gamma \subseteq L_\Sigma$ and $\varphi \in L_\Sigma$. As usual, $\Gamma \models_\Sigma \varphi$ states that for every Σ-computation system s, if $s \Vdash_\Sigma \Gamma$ then $s \Vdash_\Sigma \varphi$. Also, we use $\Gamma^{\models}$ for denoting the set $\{\varphi \in L_\Sigma \mid \Gamma \models_\Sigma \varphi\}$ (the semantic closure of Γ).

A Σ-specification is a pair $\langle \Sigma, \Gamma \rangle$ with $\Gamma \subseteq L_\Sigma$, and a Σ-theory is a Σ-specification $\langle \Sigma, \Gamma \rangle$ such that $\Gamma = \Gamma^{\models}$. A theory morphism $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ is a signature morphism $\sigma : \Sigma \to \Sigma'$ such that $\overrightarrow{\sigma}_{\mathrm{lng}}(\Gamma) \subseteq \Gamma'$. Theories and their morphisms constitute a category *Th*. We obtain a (forgetful) functor $tS : Th \to Sig$ by defining that $tS(\langle \Sigma, \Gamma \rangle) = \Gamma$ and $tS(\sigma : th \to th') = \sigma$.

Theories and interpretation structures can be related by the two following functors (recall the maps S_Σ and T_Σ presented in def. 3.2.)

Prop/Definition 4.1. We obtain a functor $S : Th^{op} \to Sys$ (the *semantics* functor) by defining that $S(\langle \Sigma, \Gamma \rangle) = \langle \Sigma, S_\Sigma(\Gamma) \rangle$ and $S(f : th \to th') = f$.

Prop/Definition 4.2. We obtain a functor $T : Sys \to Th^{op}$ (the *theory* functor) by defining that $T(\langle \Sigma, \Xi \rangle) = \langle \Sigma, T_\Sigma(\Xi) \rangle$ and $S(f : s \to s') = f$.

Proposition 4.3. The pair $\langle T, S \rangle$ is a concrete isomorphism over Sig^{op} wrt $\langle tS, sS \rangle$. Thus T and S preserve (co)limits and (co)cartesian morphisms.

5 Combining Hybrid Systems

In this section we present several forms of combining hybrid systems. All these operations can be expressed as categorial constructions.

In the following examples we will refer recurrently to the heater, its simplified version and the controller. Let $th_h = \langle \sigma_h, \Gamma_h^{\models} \rangle$, $th_{h'} = \langle \sigma_{h'}, \Gamma_{h'}^{\models} \rangle$ and $th_c = \langle \sigma_c, \Gamma_c^{\models} \rangle$ be the theories obtained from the already presented specifications, and $h = S(th_h)$, $h' = S(th_{h'})$ and $c = S(th_c)$ be the corresponding maximal models.

5.1 Free Aggregation

In this section we characterize free aggregation of two hybrid systems as a product in *Sys* and a coproduct in *Th*. By "free aggregation" we mean the composition of two systems without any form of interaction.

Because *Th* and *Sys* are concrete over *Sig*, it is useful to start by looking at coproducts in *Sig*. Coproducts in *Sig* are straightforward obtained from the (co)products in *Fin* and *PFin*.

Proposition 5.1. Let $\Sigma' = \langle X', V', A' \rangle$, $\Sigma'' = \langle X'', V'', A'' \rangle$ be signatures. The triple $\langle \Sigma' + \Sigma'', i', i'' \rangle$ where $\Sigma' + \Sigma'' = \langle X' + X'', V' \times V'', A' \times A'' \rangle$[4] and $i' : \Sigma' \to \Sigma' + \Sigma''$, $i'' : \Sigma' \to \Sigma' + \Sigma''$ are obtained from the injection/projection morphism in *Fin*/*PFin* is a coproduct of Σ' and Σ'' in *Sig*.

Example 5.2. We have that $\Sigma_{h\times c} = \langle X_{h\times c}, V_{h\times c}, A_{h\times c} \rangle$, *with* $X_{h\times c} = \{T_1, T_2, C_2\}$, $V_{h\times c} = \{\langle on, on \rangle, \langle off, on \rangle, \langle on, off \rangle, \langle off, off \rangle\}$ *and* $A_{h\times c} = \{h_1, c_1, h_2, c_2, \langle h, c \rangle, \langle c, c \rangle, \langle h, h \rangle, \langle c, h \rangle\}$, *is the signature of the free aggregation of* h *and* c.

Proposition 5.3. Let $th' = \langle \Sigma', \Gamma' \rangle$ and $th'' = \langle \Sigma'', \Gamma'' \rangle$ bet theories. The triple $\langle th' + th'', i', i'' \rangle$ where $th' + th'' = \langle \Sigma' + \Sigma'', (\overrightarrow{i'}_{\rm lng}(\Gamma') \cup \overrightarrow{i''}_{\rm lng}(\Gamma''))^{\models} \rangle$ and i', i'' are the injection morphisms defined in 5.1 is a coproduct of th' and th'' in *Th*.

Example 5.4. We have that $\langle \Sigma_{h\times c}, \Gamma_{h\times c} \rangle$ *with*

$$\begin{aligned}\Gamma_{h\times c} = \{\ & @\{tupleon, on, \langle on, off \rangle\} \Rightarrow \mathbf{D}\, T_1 = 4 - T_1/20, ...,\\ & @\{\langle on, on \rangle, \langle off, on \rangle\} \Rightarrow (0 \le C_2 \le 10)\}\end{aligned}$$

is a specification of the free aggregation of h *and* c.

Proposition 5.5. Let $s' = \langle \Sigma', \Xi' \rangle$, $s'' = \langle \Sigma'', \Xi'' \rangle$ be computation systems and $s' \times s'' = \langle \Sigma' + \Sigma'', \{\xi \in \Xi_{\Sigma'+\Sigma''} \mid \overrightarrow{i'}_{\rm cmp}(\xi) \in \Xi', \overrightarrow{i''}_{\rm cmp}(\xi) \in \Xi''\} \rangle$. The triple $\langle s' \times s'', i', i'' \rangle$ where i', i'' are the injection morphisms defined in 5.1 is a product of s' and s'' in *Sys*.

The products in *Sys* are what we wanted: a computation ξ is in $s' \times s''$ iff we can extract from ξ a computation $\overrightarrow{i'}_{\rm cmp}(\xi)$ of s' and a computation $\overrightarrow{i''}_{\rm cmp}(\xi)$ of s''. So, we can think of the computations of $s' \times s''$ as being pairs of computations from s' and s''. Coproducts in *Th* correspond to products in *Sys* (and vice-versa), so $th_h + th_c$ is a specification of $h \times c$.[5]

5.2 Restriction

Consider two systems s' and s'' that we want to interconnect. Two typical forms of interconnection are attribute and action sharing. These can be expressed as restrictions on the signature of $s' \times s''$. If we use a signature morphism $\sigma : \Sigma' + \Sigma'' \to \Sigma$ where Σ is the desired signature, then we can obtain the desired system in a canonical way. The same is true for theories.

Proposition 5.6. Let $\sigma : \Sigma \to \Sigma'$ be a morphism in *Sig*, $th = \langle \Sigma, \Gamma \rangle \in |Th_\Sigma|$ and $s = \langle \Sigma, \Xi \rangle \in |Sys_\Sigma|$. Also, let $\overrightarrow{\sigma}_{\rm thr}(th) = \langle \Sigma', \overrightarrow{\sigma}_{\rm lng}(\Gamma)^{\models} \rangle$ and $\overrightarrow{\sigma}_{\rm cs}^{-1}(s) = \langle \Sigma', \overrightarrow{\sigma}_{\rm cmp}^{-1}(\Xi) \rangle$. Then the *Th* morphism $\sigma : th \to \overrightarrow{\sigma}_{\rm thr}(th)$ is cocartesian by tS for σ on th and the *Sys* morphism $\sigma : \overrightarrow{\sigma}_{\rm cs}^{-1}(s) \to s$ is cartesian by sS for σ on s:

$$\begin{array}{c} \langle \Sigma, \Gamma \rangle \xrightarrow{\ \sigma\ } \langle \Sigma', \overrightarrow{\sigma}_{\rm lng}(\Gamma)^{\models} \rangle \\ \langle \Sigma, \Xi \rangle \xleftarrow{\ \sigma\ } \langle \Sigma', \overrightarrow{\sigma}_{\rm cmp}^{-1}(\Xi) \rangle \end{array} \qquad \Sigma \xrightarrow{\ \sigma\ } \underline{\Sigma'}$$

[4] Note that the product $A' \times A''$ is computed in *PFin*.
[5] Or, reciprocally, $h \times c$ is a model of $th_h + th_c$.

Thus, given a system s (theory th) with signature Σ and a desired signature Σ', we have a canonical way of obtaining from s (th) a system s' (theory th') with signature Σ' by resorting to a signature morphism $\sigma : \Sigma \to \Sigma'$.

Definition 5.7. Let $\Sigma' = \langle X', V', A'\rangle$ and $\Sigma'' = \langle X'', V'', A''\rangle$ be signatures. We then define $\Sigma'||\Sigma''$, $\sigma^{\Sigma'+\Sigma''}_{\Sigma'||\Sigma''} : \Sigma' + \Sigma'' \to \Sigma'||\Sigma''$, $\sigma^{\Sigma'}_{\Sigma'||\Sigma''} : \Sigma' \to \Sigma'||\Sigma''$ and $\sigma^{\Sigma''}_{\Sigma'||\Sigma''} : \Sigma'' \to \Sigma'||\Sigma''$ as follows (using σ for $\sigma^{\Sigma'+\Sigma''}_{\Sigma'||\Sigma''}$):

- $\Sigma'||\Sigma'' = \langle X' \cup X'', V' \times V'', A' \cup A''\rangle$;
- $\overrightarrow{\sigma}_{\mathrm{att}}(\overrightarrow{i'}_{\mathrm{att}}(x')) = x'$ for every $x' \in X'$ (similarly for $x'' \in X''$);
- $\overleftarrow{\sigma}_{\mathrm{vtx}} = id_{V' \times V''}$;
- $\overleftarrow{\sigma}_{\mathrm{act}}(a') = \overleftarrow{i'}_{\mathrm{act}}(a')$ for every $a' \in A' \setminus A''$ (similarly for $a'' \in A'' \setminus A'$);
- $\overleftarrow{\sigma}_{\mathrm{act}}(a) = \langle a, a\rangle$ for every $a \in A' \cap A''$;
- $\sigma^{\Sigma'}_{\Sigma'||\Sigma''} = \sigma \circ i'$, $\sigma^{\Sigma''}_{\Sigma'||\Sigma''} = \sigma \circ i''$;

where i', i'' are the injection morphisms of the coproduct of Σ' and Σ'' (see prop. 5.1). Thus in $\Sigma'||\Sigma''$ we share attributes and actions.[6]

Example 5.8. Consider again h and c. Suppose that we want to share the attribute T and actions h,c, thus obtaining a new system $h||c$ *and specification* $th_h||th_c$*. This system has signature* $\Sigma_{h||c} = \Sigma_h||\Sigma_c = \langle\{T, C\}, V_{h\times c}, \{h, c\}\rangle$*. Then* $\sigma = \sigma^{\Sigma h\times c}_{\Sigma_{h||c}} : \Sigma_{h\times c} \to \Sigma_{h||c}$ *is such that* $\overrightarrow{\sigma}_{\mathrm{att}} = \{T_1 \mapsto T, T_2 \mapsto T, C_2 \mapsto C\}$, $\overleftarrow{\sigma}_{\mathrm{vtx}} = id_{V_{h\times c}}$ *and* $\overleftarrow{\sigma}_{\mathrm{act}} = \{h \mapsto \langle h, h\rangle, c \mapsto \langle c, c\rangle\}$*. We have that* $h||c = \overrightarrow{\sigma}_{\mathrm{cs}}^{-1}(h \times c)$ *and* $th_h||th_c = \overrightarrow{\sigma}_{\mathrm{thr}}(th_h + th_c)$*. Namely:*

$$\begin{aligned}\Gamma_{h||c} = \{\ & @\{\langle on, on\rangle, \langle on, off\rangle\} \Rightarrow \mathbf{D}T = 4 - T/20, ...,\\ & @\{\langle on, on\rangle, \langle off, on\rangle\} \Rightarrow (0 \le C \le 10)\}\end{aligned}$$

5.3 Abstraction

In some cases it is desirable to abstract away certain details of a system. For instance, we may be interested in hiding attributes and actions or representing a set of operation modes by a more general mode of operation. This construction can be carried out in a way similar to the restriction operation.

Proposition 5.9. Let $\sigma : \Sigma \to \Sigma'$ be a morphism in *Sig*, $th' = \langle\Sigma', \Gamma'\rangle \in |Th_{\Sigma'}|$ and $s' = \langle\Sigma', \Xi'\rangle \in |Sys_{\Sigma'}|$. Also, let $\overrightarrow{\sigma}_{\mathrm{thr}}^{-1}(th') = \langle\Sigma, \overrightarrow{\sigma}_{\mathrm{lng}}^{-1}(\Gamma')\rangle$ and $\overrightarrow{\sigma}_{\mathrm{cs}}(s') = \langle\Sigma, \overrightarrow{\sigma}_{\mathrm{cmp}}(\Xi')^{\equiv}\rangle$. Then the *Th* morphism $\sigma : \overrightarrow{\sigma}_{\mathrm{thr}}^{-1}(th') \to th'$ is cartesian by *tS* for σ on th' and the *Sys* morphism $\sigma : s' \to \overrightarrow{\sigma}_{\mathrm{cs}}(s')$ is cocartesian by *sS* for σ on s':

$$\begin{array}{ccc}\langle\Sigma, \overrightarrow{\sigma}_{\mathrm{lng}}^{-1}(\Gamma')\rangle \xrightarrow{\ \sigma\ } \langle\Sigma', \Gamma'\rangle & & \\ & & \underline{\Sigma} \xrightarrow{\ \sigma\ } \Sigma' \\ \langle\Sigma, \overrightarrow{\sigma}_{\mathrm{cmp}}(\Xi')^{\equiv}\rangle \xleftarrow{\ \sigma\ } \langle\Sigma', \Xi'\rangle & & \end{array}$$

[6] This amounts to the HyTech parallel composition operation.

Thus, given a system s' (theory th') with signature Σ' and a desired signature Σ, we have a canonical way of obtaining from s' (th') a system s (theory th) with signature Σ by resorting to a signature morphism $\sigma : \Sigma \to \Sigma'$.

Example 5.10. *Consider an abstraction of h in which we hide T. This can be done by considering the signature morphism* $a_{h} : \Sigma_{h'} \to \Sigma_{h}$ *such that* $\overleftarrow{a_h}_{\mathrm{vtx}} = id_{V_h}$ *and* $\overleftarrow{a_h}_{\mathrm{act}} = id_{A_h}$*. Then* $\overrightarrow{a_h}^{-1}_{\mathrm{thr}}(th_h) = th_{h'}$ *and* $\overrightarrow{a_h}_{\mathrm{cs}}(h) = h'$*. Similarly, we can simplify c to get c' by hiding T and C, using a morphism* $a_c : \Sigma_{c'} \to \Sigma_c$*.*

5.4 Putting it All Together

Suppose we have two pairs of systems a, b and a', b' that we interconnect by using signature morphisms $q : \Sigma_a + \Sigma_b \to \Sigma$, $r : \Sigma_{a'} + \Sigma_{b'} \to \Sigma'$ thereby obtaining systems $s = \overrightarrow{q}^{-1}_{\mathrm{cs}}(a \times b)$ and $s' = \overrightarrow{k}^{-1}_{\mathrm{cs}}(a' \times b')$. Consider also that there are morphisms $h_a : a \to a'$, $h_b : b \to b'$. We might then ask if there is any relation between s and s'. First of all, because $h_a \circ p_a : a \times b \to a'$ and $h_b \circ p_b : a \times b \to b'$ (p_a, p_b are the product projection morphisms) we conclude that h_a, h_b uniquely determine a morphism $u : a \times b \to a' \times b'$:

$$\begin{tikzcd}
 & & a \arrow[r, "h_a"] & a' & & \\
s \arrow[r, "q"] & a \times b \arrow[ur, "p_a"] \arrow[dr, "p_b"'] \arrow[rrr, dotted, "u"] & & & a' \times b' \arrow[ul, "p_{a'}"'] \arrow[dl, "p_{b'}"] & s' \arrow[l, "r"'] \\
 & & b \arrow[r, "h_b"] & b' & &
\end{tikzcd}$$

Let $k = u \circ q$. *In the case* that there is $t : \Sigma' \to \Sigma$ such that $k = t \circ r$, then the universal property of the cartesian morphism $r : s' \to a' \times b'$ guarantees that $t : s \to s'$ is a morphism in *Sys*:[7]

$$\begin{tikzcd}
s' \arrow[r, "r"] & a' \times b' \\
s \arrow[u, dotted, "t"] \arrow[ur, "k"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\Sigma' \arrow[d, "t"'] & \Sigma_{a'} + \Sigma_{b'} \arrow[l, "r"'] \arrow[dl, "k"] \\
\Sigma
\end{tikzcd}$$

Example 5.11. *We have a morphism* $t : h||c \to h'||c'$ *obtained from* $q = \sigma^{\Sigma_{h \times c}}_{\Sigma_{h||c}}$ *and* $r = \sigma^{\Sigma_{h' \times c'}}_{\Sigma_{h'||c'}}$*:* $f = \langle \emptyset, id_{V_h \times V_c}, id_{A_h \cup A_c} \rangle$*.*

6 Compositional Model Checking

In this section we present the main concepts of a model checking procedure for hybrid systems. The main result here is that the model checking procedure of HyTech can be extended to take advantage of the categorial structure of a complex system.

[7] Informally speaking, the existence of t means that the restriction morphisms q and r are *compatible*.

6.1 Reachability

The reachability problem is the following: given a hybrid system s and two sets of states W and W', is it possible for s to reach a state in W' starting from a state in W? Certain interesting classes of properties (namely safety and some liveness properties) can be expressed as reachability problems.

Definition 6.1. Let $s = \langle \Sigma, \Xi \rangle$ be a computation system. Then s induces the (reachability) relation $\rightarrow_s \subset St_\Sigma \times St_\Sigma$ defined by $w \rightarrow_s w'$ iff $\xi(\langle i, t\rangle) = w$ and $\xi(\langle i', t'\rangle) = w'$ for some $\xi \in \Xi$ and $\langle i, t\rangle, \langle i', t'\rangle \in spa_\xi$ s.t. $\langle i, t\rangle \leq \langle i', t'\rangle$.[8]

Definition 6.2. Let $s = \langle \Sigma, \Xi \rangle$ be a computation system. We define the (forward and backward reachability) maps $\mathcal{F}_s, \mathcal{B}_s : 2^{St_\Sigma} \rightarrow 2^{St_\Sigma}$ as

- $\mathcal{F}_s(W) = \{w' \in St_\Sigma \mid w \rightarrow_s w' \text{ for some } w \in W\}$
- $\mathcal{B}_s(W') = \{w \in St_\Sigma \mid w \rightarrow_s w' \text{ for some } w' \in W'\}$

In general, these maps are not computable and so the reachability problem is undecidable. However, the reachability problem is semi-decidable for systems specified with a subset of our specification language (that corresponds to HyTech's specification language).

Proposition 6.3. Let Σ, Σ' be signatures, $s \in |Sys_\Sigma|$ and $s' \in |Sys_{\Sigma'}|$. Also, let $f : s \rightarrow s'$ be a morphism in Sys. Then f preserves reachability, that is, for every $w, w' \in S_\Sigma$ and $W \subseteq S_\Sigma$:

- if $w \rightarrow_s w'$ then $\vec{f}_{\text{st}}(w) \rightarrow_{s'} \vec{f}_{\text{st}}(w')$;
- $\vec{f}_{\text{st}}(\mathcal{F}_s(W)) \subseteq \mathcal{F}_{s'}(\vec{f}_{\text{st}}(W))$
- $\vec{f}_{\text{st}}(\mathcal{B}_s(W)) \subseteq \mathcal{B}_{s'}(\vec{f}_{\text{st}}(W))$

6.2 Verification

The notions presented in the previous subsection enable us to verify certain classes of properties. Because the notions presented are semantic ones, we start by defining their syntactic counterpart.

Definition 6.4. Let Σ be a signature. The set of *Σ-state terms*, sT_Σ, is the set of Σ-terms that can be written without the **D** and **N** operators. Likewise, the set of *Σ-state formulae*, sL_Σ, is the set of Σ-formulae that can be written without the **N**, ∇, **U** and **S** operators.

A state is all we need to interpret state formulae.

Definition 6.5. Let Σ be a signature. The *state formula interpretation map* $[\![\_]\!]_\Sigma : sL_\Sigma \rightarrow 2^{St_\Sigma}$ is defined by $[\![\varphi]\!]_\Sigma = \{w \in St_\Sigma \mid w \text{ satisfies } \varphi\}$.

[8] That is, w' can be reached from w iff there is some computation where w' occurs after w.

Proposition 6.6. Let $\Sigma \in |Sig|$, $s \in |Sys_\Sigma|$ and $\varphi_0, \varphi \in sL_\Sigma$. Then
$$s \Vdash_\Sigma \varphi_0 \Rightarrow \mathbf{G}\varphi \quad \text{iff} \quad \mathcal{F}_s([\![\varphi_0]\!]_\Sigma) \cap [\![\neg\varphi]\!]_\Sigma = \emptyset \quad \text{iff} \quad [\![\varphi_0]\!]_\Sigma \cap \mathcal{B}_s([\![\neg\varphi]\!]_\Sigma) = \emptyset$$

Example 6.7. We can prove that $h \Vdash 0 < T < 80 \Rightarrow \mathbf{G}(0 < T < 80)$, *from which we can conclude that* $h \Vdash 0 < T < 80 \Rightarrow \mathbf{G}(-4 < \mathbf{D}T < 4 \wedge \mathbf{D}T \neq 0)$.

Definition 6.8. A *monitor for a using C* is a system $m_{a,C} = S(\langle \Sigma_{a,C} \rangle, \Gamma^{\models}_{a,C})$ where

- $\Sigma_{a,C} = \langle \{C\}, \{n, y\}, \{a\} \rangle$
- $\Gamma_{a,C} = \{@n \Rightarrow \mathbf{D}C = 1, @y \Rightarrow \mathbf{D}C = 0, \nabla a \Rightarrow \mathbf{N}@y, (\neg \nabla a) \Rightarrow \mathbf{Cnt}_{\Sigma_{a,C}}\}$

We use $\star_{m_{a,C}}$ for the formula $C = 0 \wedge @n$.

Thus, if $m_{a,C}$ starts in a state satisfying $\star_{m_{a,C}}$, it will record the time of the first occurrence of a or keep running the clock forever if a never occurs.

Proposition 6.9. Let $\Sigma = \langle X, V, A \rangle$ be a signature, s a Σ-computation system, φ_0 a Σ-state formula and $C \notin X$. Then
$$s \Vdash \varphi_0 \Rightarrow \mathbf{F}^{\leq\delta}\nabla a \quad \text{iff} \quad s||m_{a,C} \Vdash (\varphi_0' \wedge \star_{m_{a,C}}') \Rightarrow \mathbf{G}(C \leq \delta)$$
$$s \Vdash \varphi_0 \Rightarrow \mathbf{G}^{\leq\delta}(\neg \nabla a) \quad \text{iff} \quad s||m_{a,C} \Vdash (\varphi_0' \wedge \star_{m_{a,C}}') \Rightarrow \mathbf{G}(y' \Rightarrow C > \delta)$$

where $\varphi_0' = \overrightarrow{\sigma}^{\Sigma}_{\Sigma||\Sigma_{a,C}}(\varphi)$, $\star_{m_{a,C}}' = \overrightarrow{\sigma}^{\Sigma_{a,C}}_{\Sigma||\Sigma_{a,C}}(\star_{m_{a,C}})$ and $y' = \overrightarrow{\sigma}^{\Sigma_{a,C}}_{\Sigma||\Sigma_{a,C}}(@y)$.

Example 6.10. We can prove[9] *that* $c \Vdash @on \Rightarrow \mathbf{F}^{\leq 10}\nabla c$, *from which we can conclude that* $c \Vdash \nabla h \Rightarrow (T \leq 40 \wedge \mathbf{F}^{\leq 10}\nabla c)$.

6.3 Compositional Verification

In this subsection we show (by means of some examples) that it is possible to take advantage of the categorial structure of a complex system to ease the verification of properties of the system.

Suppose we have two systems $s \in |Sys_\Sigma|$, $s' \in |Sys_{\Sigma'}|$ related by a morphism $f : s \to s'$ and a formula $\varphi \in L_\Sigma$. In some cases, we may prove that $s \Vdash \varphi$ by executing the following steps (this method is particularly advantageous if s' is "simpler" than s):

1. Find $\varphi' \in L_{\Sigma'}$ such that $\varphi = \overleftarrow{f}_{\mathrm{lng}}(\varphi')$.[10]
2. Prove that $s' \Vdash \varphi'$.
3. Conclude that $s \Vdash \varphi$ because morphisms in Sys reflect properties.

The three systems considered so far can be related in the following way (a_h, a_c are the morphism introduced in example 5.10, t is the morphism introduced in example 5.11, p_h and p_c are projection morphisms and $q = \sigma^{\Sigma_{h\times c}}_{\Sigma_{h||c}}$):

[9] Using HyTech!
[10] Of course, this is not always possible.

$$\begin{tikzcd} & & h \arrow[r, "ah"] & h\text{'} \\ h||c \arrow[r, "q"] & h \times c \arrow[ur, "p_h"] \arrow[dr, "p_c"'] & & \\ & & c \arrow[r, "ac"'] & c\text{'} \end{tikzcd} \qquad\qquad h||c \xrightarrow{t} h\text{'}||c\text{'}$$

Consider the formulae $\varphi_1 = @\{\langle on, on\rangle, \langle off, off\rangle\} \Rightarrow \mathbf{G}@\{\langle on, on\rangle, \langle off, off\rangle\}$ and $\varphi_2 = \nabla h \Rightarrow \mathbf{F}^{\leq 10}\nabla c$. Using the above strategy we can prove that $h||c \Vdash \varphi_i$ $(i = 1, 2)$:

1. Find $\varphi'_1 \in L_{\Sigma_{h'||c'}}$ such that $\varphi = \overleftarrow{t}_{\rm lng}(\varphi'_1)$. We can choose $\varphi'_1 = \varphi_1$.
2. Prove that $h'||c' \Vdash \varphi'_1$. We can use HyTech to prove this.
3. Conclude that $h||c \Vdash \varphi_1$.

and

1. Find $\varphi'_2 \in L_{\Sigma_c}$ such that $\varphi_2 = \overleftarrow{p_c \circ q}_{\rm ng}(\varphi'_2)$. We can choose $\varphi'_2 = \varphi_2$.
2. Prove that $c \Vdash \varphi'_2$. We have already seen that this is the case.
3. Conclude that $h||c \Vdash \varphi_2$.

7 Concluding Remarks

The main goal of this paper has been achieved: we were able to construct an institution of hybrid systems and provide a categorial characterization of free aggregation, restriction and abstraction of such systems, thus providing a basis for compositional specification and verification of hybrid systems.

Namely, we showed that in the category of hybrid systems free aggregation corresponds to a product, restriction to a cartesian lifting and abstraction to a cocartesian lifting. The lifting is done from the category of signatures, where the desired signatures and morphisms are chosen. The (co)cartesian lifting obtained are very general, not being restricted to inclusion of attributes or action symbols. Also, because of the isomorphism between theories and hybrid systems, these semantic constructions have a well defined syntactic counterpart.

Interconnection of two systems was shown to be possible by doing a restriction operation on the free aggregation of the systems. Standard interconnection constructions (such as HyTech's parallel composition) appear as special cases on our framework.

It was also shown (by the mean of some examples) that compositional verification of hybrid systems is possible. Namely, the categorial structure of complex systems can be used to ease the verification process. More work is ongoing in this direction.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

2. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.
3. P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems IV*. Lecture Notes in Computer Science 1273. Springer-Verlag, 1997.
4. L. de Alfaro, A. Kapur, and Z. Manna. Hybrid diagrams: A deductive-algorithmic approach to hybrid system verification. In R. Reischuk and M. Morvan, editors, *STACS'97*, volume 1200 of *Lecture Notes in Computer Science*, pages 153–164. Springer Verlag, 1997.
5. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.
6. A. Kapur, T.A. Henzinger, Z. Manna, and A. Pnueli. Proving safety properties of hybrid systems. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 431–454. Springer-Verlag, 1994.
7. H. Lourenço and A. Sernadas. Combining hybrid systems. Research report, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1096 Lisboa, Portugal, 1999. Submitted for publication.
8. H. Lourenço, A. Sernadas, and C. Sernadas. Aggregation and interconnection of hybrid automata: Categorial characterization. Research report, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1096 Lisboa, Portugal, 1998. Presented at FIREworks Meeting, Magdeburg, May 15-16.
9. A. Sernadas, C. Sernadas, and C. Caleiro. Denotational semantics of object specification. *Acta Informatica*, 35:729–773, 1998.
10. Glynn Winskel and Mogens Nielsen. *Semantics and Logics of Computation*, chapter Categories in Concurrency, pages 299–354. Publications of the Newton Institute. 1997.