Combining Hybrid Systems

Hugo Lourenço and Amílcar Sernadas

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

Corresponding author:
Hugo Miguel Ramos Lourenço,
Secção de Ciência da Computação,
Dep. Matemática, IST,
Av. Rovisco Pais, Lisboa, Portugal
Tel. +351-1-8417144, Fax. +351-1-8499242
http://www.cs.math.ist.utl.pt/s84.www/cs/hlouren.html

Abstract. The problem of combining hybrid systems is addressed within a categorial setting. A notion of hybrid system with a clear identification of the signature component is introduced. A suitable notion of morphism is presented, leading to a categorial characterization of free aggregation, restriction, abstraction and interconnection as universal constructions. The HyTech [4] parallel composition constructor is shown to be a particular case of interconnection.

1 Introduction

A hybrid system is a system that exhibits a phased evolution, intermixing continuous and discrete behavior. One can describe such a system in terms of its (real-valued) attributes, modes of operation and available actions. Each mode of operation describes how the attributes (continuously) change.

At certain time instants a transition between modes of operations occurs. During this (discrete) transition the attributes values can change abruptly. Furthermore, the decision to undertake the transition can be made based on the attributes values.

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.

As an example, consider an heater. A possible attribute is the current temperature. We can consider an "heating" mode of operation during which the

* 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.

temperature is continuously increasing. 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 finite 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 hybrid system is described by two components: the temperature and the current mode of operation.

Interest in hybrid systems has grown in recent years, mainly because of their direct relation with embedded systems (systems that interact with the *continuously* changing real-world). These systems often arise in safety critical situations, so formal verification methods have been the subject of intense research [1–3, 5].

However, in order to deal with real world problems compositional verification methods are known to be needed. Having this in mind, we tackle the problem of combining hybrid systems from a categorial point of view (preliminary work can be found in [7].) 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 automata the well known definitions of aggregation and interconnection of transition systems as presented in [11].

Herein, we only deal with the semantic aspects of the combination of hybrid systems. The syntactical counterpart of the constructions presented here can be found in [6]. For more information on compositional specification see [8, 10, 9].

The paper is organized in two parts. In the first part we present our model of a hybrid system. In the second part we investigate some forms of combining hybrid systems and provide a categorial characterization of the considered constructions.

A bit of category theory is needed to fully understand the paper. For the sake of completeness, all the relevant concepts are presented in the appendix.

2 Hybrid Computation Systems

In this section we present our adopted model of hybrid systems. We regard a hybrid system as a set of (hybrid) computations that correspond to the possible evolutions of the system.

In the sequel, *Fin* denotes the category of finite sets and maps between finite sets and *PFin* denotes the category of finite sets and partial maps between finite sets (as defined in the appendix.)

2.1 The Signature Category

We start by presenting the category of hybrid signatures, *Sig*. From what has been said before, we already know that we talk about a hybrid system in terms of its attributes, modes of operation and action symbols. We thus have the objects for our category of signatures; the signature is the *interface* of the system. In order to motivate the choice of signature morphisms, lets look at what we want to get as signature coproducts.

If we take the signatures of two hybrid 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 of the component systems; 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.)

Is is thus natural to adopt the following definition for the signature category.

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

Given a signature $\langle X, V, A\rangle$, X is a set of real-valued attributes, V is set of vertices and A is a set of action symbols. Together, a vertex $v \in V$ and a valuation $\alpha \in \mathbb{R}^X$ define a state of the hybrid system. Clearly, each vertex corresponds to a mode of operation. The action symbols are used to label discrete transitions and play a major role in the interconnection of hybrid systems.

A morphism $\sigma : \Sigma \to \Sigma'$ has the form $\langle \overrightarrow{\sigma}_{\mathrm{att}} : X \to X', \overleftarrow{\sigma}_{\mathrm{vtx}} : V' \to V, \overleftarrow{\sigma}_{\mathrm{act}} : A' \rightharpoonup A\rangle$. If $f : \Sigma' \to \Sigma$ is a signature op-morphism, then we reverse the arrows: $\langle \overleftarrow{f}_{\mathrm{att}} : X \to X', \overrightarrow{f}_{\mathrm{vtx}} : V' \to V, \overrightarrow{f}_{\mathrm{act}} : A' \rightharpoonup A\rangle$.

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 $T' = 2 - T$; when it is off, the temperature decreases according to the law $T' = -T$.*[1] *The signature $\Sigma_h = \langle X_h, V_h, A_h\rangle$ with $X_h = \{T\}$, $V_h = \{\mathrm{on}, \mathrm{off}\}$ and $A_h = \{heat, cool\}$ is a possible signature one for this heater.*

Example 2.3. Consider a controller for the preceding heater: for safety reasons, the controller will turn off the heater within .25 *time units if the temperature rises above* 1/2*. A possible signature for such a controller is $\Sigma_{c1} = \langle X_{c1}, V_{c1}, A_{c1}\rangle$ with $X_{c1} = \{T\}$, $V_{c1} = \{v\}$ and $A_{c1} = \{cool\}$.*

Example 2.4. The controller presented in the preceding example is fairly abstract. A real controller would have a clock C to measure the time elapsed since the temperature rose above 1/2*. If the clock reaches* .25 *without the heater being turned off in the meantime, then the controller turns the heater off. A possible signature this controller is $\Sigma_{c2} = \langle X_{c2}, V_{c2}, A_{c2}\rangle$ with $X_{c2} = \{T, C\}$, $V_{c2} = \{\mathrm{c\_on}, \mathrm{c\_off}\}$ and $A_{c2} = \{cool, start\}$.*

2.2 Hybrid Computations

A hybrid system works by intermixing *continuous phases* with *discrete jumps*. We must then start by defining what a phase is.

Definition 2.5. Let $\Sigma = \langle X, V, A\rangle$ be a signature. A *Σ-history* is a function $h : I \to \mathbb{R}^X$, where $I \subseteq \mathbb{R}_0^+$ is a closed interval such that $0 \in I$. In the case

[1] We are not concerned with the units T is expressed on.

that $I \neq \{0\}$, we require that for each $x \in X$, $h^x \stackrel{\text{def}}{=} \lambda t.h(t)(x)$ be differentiable and define the maps $Dh^x : I \to \mathbb{R}$, $Dh : I \to \mathbb{R}^X$ by $Dh(t) = \frac{dh^x}{dt}(t)$ and $Dh(t)(x) = Dh^x(t)$. We use I_h when referring to the interval I, and denote by H_Σ the set of Σ-histories.

Example 2.6. Two possible histories for the heater are $h_{\text{on}}, h_{\text{off}} : \mathbb{R}_0^+ \to \mathbb{R}^{X_h}$ with $h_{\text{on}}^T(t) = 2 - 2e^{-t}$ and $h_{\text{off}}^T(t) = 2e^{-t}$. Two possible histories for the second controller are $h_{c\_on} : [0, .25] \to \mathbb{R}^{X_{c2}}$, $h_{c\_off} : \mathbb{R}_0^+ \to \mathbb{R}^{X_{c2}}$ with $h_{\text{c\_on}}^C(t) = t$ and $h_{\text{c\_off}}^C(t) = 0$.[2]

Definition 2.7. Let $\Sigma = \langle X, V, A\rangle$ be a signature and $h, h' \in H_\Sigma$. We say that h is a *sub-history* of h' (denoted by $h \subseteq h'$) iff (1) $I_h \subseteq I_{h'}$ and (2) there is some $t_0 \in \mathbb{R}_0^+$ such that for all $t \in I_h$, $t_0 + t \in I_{h'}$ and $h(t) = h'(t_0 + t)$. We define $Sub(h') = \{h \in H_\Sigma \mid h \subseteq h'\}$ as the set of h' sub-histories.

Definition 2.8. Let $\Sigma = \langle X, V, A\rangle$ be a signature. A *Σ-phase* is a pair $p = \langle v, h\rangle$ with $v \in V$ and $h \in H_\Sigma$. We use I_p when referring to the interval I_h, and denote by P_Σ the set of Σ-phases.

Definition 2.9. Let $\Sigma = \langle X, V, A\rangle$ be a signature and $p = \langle v, h\rangle$, $p' = \langle v', h'\rangle$ Σ-phases. We say that p is a *sub-phase* of p' (denoted by $p \subseteq p'$) iff $v = v'$ and $h \subseteq h'$. We define $Sub(p') = \{p \in P_\Sigma \mid p \subseteq p'\}$ as the set of p' sub-phases.

Example 2.10. Some possible sets of phases for the heater are $P_{\text{on}} = Sub(\langle \text{on}, h_{\text{on}}\rangle)$ and $P_{\text{off}} = Sub(\langle \text{off}, h_{\text{off}}\rangle)$. Some possible sets of phases for the second controller are $P_{\text{c\_on}} = \{\langle \text{c\_on}, h\rangle \mid h \in Sub(h_{\text{c\_on}})$ and $h^C(0) = 0\}$, $P_{\text{c\_off}} = Sub(\langle \text{c\_off}, h_{\text{c\_off}}\rangle)$.

A computation is an alternating sequence of phases and discrete jumps. Each discrete jump is possibly labelled with an action symbol.[3] We are thus lead to the following definition, where a range $0..n$ denotes the set $\{i \in \mathbb{N}_0 \mid i \leq n\}$.

Definition 2.11. 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 (1) $I_{p_i} = \mathbb{R}_0^+$ iff $n \neq +\infty$ and $i = n$ and (2) $\sum length(I_{p_i}) = +\infty$.

We denote the length of I_{p_i} by d_i (the duration of phase p_i), and use Ξ_Σ for the set of Σ-computations. Also, note that each p_i is of the form $\langle v_i, h_i\rangle$, and so we freely refer to v_i,h_i when needed.

The number of phases in the computation is given by n, the phases by p and the action symbol labelling the end of the phases by a. For technical reasons we only consider computations that span over all of the real time-line.

Example 2.12. The set Ξ_h of the heater's computations consists of all computations $\xi = \langle n, p, a\rangle \in \Xi_{\Sigma_h}$ such that for every i:

[2] Note that no restrictions is imposed on $h_{c\_on}^T$ or $h_{\text{c\_off}}^T$.

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

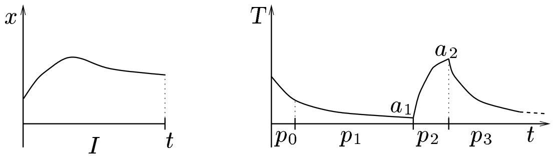

Fig. 1. A phase and a computation

- $p_i \in (P_{on} \cup P_{off})$, $h_i(d_i) = h_{i+1}(0)$ *and* a_i *is defined;*
- *if* $a_i =$ *heat then* $p_i \in P_{off}$ *and* $p_{i+1} \in P_{on}$ *and*
- *if* $a_i =$ *cool then* $p_i \in P_{on}$ *and* $p_{i+1} \in P_{off}$.

Thus the heater alternates between on and off phases, and the temperature is always continuous.

The set Ξ_{c1} *of the first controller's computations consists of all computations* $\xi = \langle n, p, a\rangle \in \Xi_{\Sigma_{c1}}$ *such that for every* i:

- $h_i(d_i) = h_{i+1}(0)$ *and* a_i *is defined and*
- *if* $d_i \neq 0$ *and there is* $t \leq d_i$ *such that* $h_i^T(t) = .5$ *and* $Dh_i^T(t) > 0$ *then* $t + .25 \leq d_i$.

Thus all controller phases end with the occurrence of the cool action. Also, if the temperature rises above .5 then the phase ends within .25 time units.[4]

The set Ξ_{c2} *of the second controller's computations consists of all computations* $\xi = \langle n, p, a\rangle \in \Xi_{\Sigma_{c2}}$ *such that for every* i:

- $p_i \in (P_{c\_on} \cup P_{c\_off})$, $h_i^T(d_i) = h_{i+1}^T(0)$ *and* a_i *is defined;*
- *if* $a_i =$ *cool then* $p_{i+1} \in P_{c\_off}$;
- *if* $a_i =$ *start then* $p_i \in P_{c\_off}$ *and* $p_{i+1} \in P_{c\_on}$;
- *if* $p_i \in P_{c\_off}$, $d_i \neq 0$ *and there is* $t \leq d_i$ *such that* $h_i^T(t) = .5$ *and* $Dh_i^T(t) > 0$ *then* $t = d_i$ *and* $a_i =$ *start.*

Thus the controller starts the clock whenever the temperature rises above .5; then within .25 time units the cool action is issued (recall the definition of P_{c_on}: *all phases where the clock is on last at most .25 time units.)*

Definition 2.13. A morphism $\sigma : \Sigma \to \Sigma'$ induces the maps $\overleftarrow{\sigma}_{\mathrm{asg}} : \mathbb{R}^{X'} \to \mathbb{R}^{X}$, $\overleftarrow{\sigma}_{\mathrm{hst}} : H_{\Sigma'} \to H_{\Sigma}$, $\overleftarrow{\sigma}_{\mathrm{ph}} : P_{\Sigma'} \to P_{\Sigma}$ and $\overleftarrow{\sigma}_{\mathrm{cmp}} : \Xi_{\Sigma'} \to \Xi_{\Sigma}$ defined as follows:

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

Thus a signature morphism induces a *translation* between computations.

Definition 2.14. 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 $\overrightarrow{f}_{\mathrm{cmp}}(\xi') = \xi$.

Prop/Definition 2.15. Computations and their morphisms constitute a category *Comp*.

[4] Of course, the condition $T = .5$ and $T' > 0$ does not cover all the situations in which the temperature is just rising above 1/2...

2.3 Hybrid Computation Systems

Capitalizing on the previous section, we build the category of of hybrid computation systems, *Sys*.

A hybrid computation system consists of a set of computations. However, not every set of computations will do: there are pairs of computations that we do not wish to distinguish, and so they should both be present or absent in a given computation system.

We then start by considering a notion of *pre-computation system* that is then refined into the desired notion of *computation system*.

Definition 2.16. Let Σ be a signature. A *Σ-pre-computation system* is a pair $\langle \Sigma, \Xi \rangle$ where $\Xi \subseteq \Xi_\Sigma$.

We do not want to distinguish computations which differ only by the occurrence of unlabelled discrete transitions that do not introduce discontinuities. To make clear this concept, we start by defining when can histories and phases be "fused".

Definition 2.17. Let Σ be a signature and $h, h' \in H_\Sigma$ such that, letting $d = length(I_h)$ and $d' = length(I_{h'})$, we have (1) $d_i \neq +\infty$ (i.e., I_h has finite length), (2) $h(d) = h'(0)$ and (3) if $d, d' \neq 0$ then $Dh(d) = Dh'(0)$. We then say that h can be *fused* with h' and denote by $h; h'$ the history $(I; I') \to \mathbb{R}^X$ where

$$I; I' = \begin{cases} \mathbb{R}_0^+ & \text{if } d' = +\infty \\ [0, d + d'] & \text{otherwise} \end{cases} \qquad (h; h')(t) = \begin{cases} h(t) & \text{if } t \leq d \\ h'(t - d) & \text{otherwise} \end{cases}$$

Definition 2.18. Let Σ be a signature and $p = \langle v, h \rangle$, $p' = \langle v', h' \rangle$ be Σ-phases such that $v = v'$ and h can be fused with h'. We then say that p can be *fused* with p' and denote by $p; p'$ the phase $\langle v, h; h' \rangle$.

We can now associate with a computation a "reduced" version of it where we omit the unlabelled discrete transitions that do not introduce discontinuities.

Definition 2.19. Let Σ be a signature and $\xi = \langle n, p, a \rangle$ a Σ-computation. A *segment* of ξ starting at k is a tuple $sgm = \langle n', p', a' \rangle$ such that

- $k \in \mathbb{N}$ and $k \leq n$;
- $n' \in \overline{\mathbb{N}}_0$ and $n' \leq n - k$;
- $p' : 0..n' \to P_\Sigma$ is such that $p'(i) = p(i + k)$;
- $a' : 0..(n' - 1) \rightharpoonup A$ is such that $a'(i) = a(i + k)$.

We say that the segment can be *fused* into a unique phase iff for every $i < n'$ we have that (1) $p'(i)$ can be fused with $p'(i + 1)$ and (2) $a'(i)$ is undefined, and denote by $[sgm]$ the phase resulting from fusing all phases in sgm. Also, we say that the segment is *maximally fuse-able* if it is fuse-able and cannot be extended by considering more phases at its beginning or end.

Note that a segment of ξ is a subsequence of contiguous phases of ξ. The fact that all computations diverge in time is essential to guarantee that $[sgm]$ exists. Moreover, this is the only point where the time divergence restriction is needed. We can lift it by providing other (more cumbersome?) definition of equivalence.

Definition 2.20. Let Σ be a signature and $\xi \in \Xi_\Sigma$. We denote by ξ^R the computation obtained by substituting each maximally fuse-able segment sgm of ξ by $[sgm]$.

Definition 2.21. Let Σ be a signature and $\xi, \xi' \in \Xi_\Sigma$. We say that ξ, ξ' are *equivalent*, denoted by $\xi \equiv \xi'$, iff $\xi^R = \xi'^R$. Given $\Xi \subseteq \Xi_\Sigma$, we denote by $\Xi^\equiv$ the set $\{\xi \in \Xi_\Sigma \mid \xi \equiv \xi' \text{ for some } \xi' \in \Xi\}$.

Note that $\equiv$ is an equivalence relation. An important property of a non-empty set of computations $\Xi^\equiv$ is that it is *open for interaction*, in the following sense: for each point in time t that we might consider, there is always a computation $\xi \in \Xi^\equiv$ such that a discrete transition occurs in ξ at time t. This property is essential to ensure that two hybrid systems can be put together with no interference.

We now define a computation system as a pre-computation system $\langle \Sigma, \Xi \rangle$ that is closed for equivalent computations.

Definition 2.22. Let Σ be a signature. A *Σ-computation system* is a Σ-pre-computation system $\langle \Sigma, \Xi \rangle$ such that $\Xi = \Xi^\equiv$.

Example 2.23. We can define the following computation systems for the heater and the two possible controllers: $h = \langle \Sigma_h, \Xi_h^\equiv \rangle$, $c1 = \langle \Sigma_{c1}, \Xi_{c1}^\equiv \rangle$, $c2 = \langle \Sigma_{c2}, \Xi_{c2}^\equiv \rangle$.

Definition 2.24. A computation system morphism $f : \langle \Sigma', \Xi' \rangle \to \langle \Sigma, \Xi \rangle$ is a signature morphism $f : \Sigma \to \Sigma'$ such that $\overrightarrow{f}_{\mathrm{cmp}}(\Xi') \subseteq \Xi$.

As for computations, we obtain a category for computation systems.

Prop/Definition 2.25. Computation systems and their morphisms constitute a category *Sys*.

3 Combining Hybrid Systems

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

The categories *Sys* and *Sig* can be related by a forgetful functor, which extracts only the signature component from *Sys*' objects and morphisms. This functor is useful for lifting constructions from *Sig* to *Sys*.

Prop/Definition 3.1. We obtain a functor $S : Sys \to Sig^{op}$ by defining that $S(\langle \Sigma, \Xi \rangle) = \Sigma$ and $S(f : s \to s') = f$.

3.1 Free Aggregation

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

Because *Sys* is built over Sig^{op}, it is useful to start by looking at coproducts in *Sig*. Coproducts in *Sig* are straightforwardly obtained from the (co)products in *Fin* and *PFin*.

Proposition 3.2. The triple $\langle \Sigma' + \Sigma'', i', i'' \rangle$ where

$$\Sigma' + \Sigma'' = \langle X' + X'', V' \times V'', A' \times A'' \rangle^5$$

and $i' : \Sigma' \to \Sigma' + \Sigma''$, $i'' : \Sigma' \to \Sigma' + \Sigma''$ are obtained from the injection/projection morphisms in *Fin*/*PFin* is a coproduct of $\Sigma' = \langle X', V', A' \rangle$ and $\Sigma'' = \langle X'', V'', A'' \rangle$ in *Sig*.

Example 3.3. We have that $\Sigma_h + \Sigma_{c1} = \langle X_{h+c1}, V_{h+c1}, A_{h+c1} \rangle$ is the signature of the free aggregation of h and c1, where $X_{h+c1} = \{T_1, T_2\}$, $V_{h+c1} = \{\langle \mathrm{on}, v \rangle, \langle \mathrm{off}, v \rangle\}$ and $A_{h+c1} = \{heat_1, cool_1, cool_2, \langle heat, cool \rangle, \langle cool, cool \rangle\}$.

Proposition 3.4. The triple $\langle s' \times s'', i', i'' \rangle$ where

$$s' \times s'' = \langle \Sigma' + \Sigma'', \{\xi \in \Xi_{\Sigma'+\Sigma''} \mid \overrightarrow{i'}_{\mathrm{cmp}}(\xi) \in \Xi', \overrightarrow{i''}_{\mathrm{cmp}}(\xi) \in \Xi''\} \rangle$$

and i', i'' are the injection morphisms defined in 3.2 is a product of $s' = \langle \Sigma', \Xi' \rangle$ and $s'' = \langle \Sigma'', \Xi'' \rangle$ 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 $\overrightarrow{i'}_{\mathrm{cmp}}(\xi)$ of s' and a computation $\overrightarrow{i''}_{\mathrm{cmp}}(\xi)$ of s''. So, we can think of the computations of $s' \times s''$ as being pairs of computations from s' and s''. Moreover, if both s' and s'' have at least one computation, then we can recover s' and s'' from $s' \times s''$.[6] Note also the relevance of the partial labelling and the closure for equivalence: $\overrightarrow{i'}_{\mathrm{cmp}}(\xi)$ and $\overrightarrow{i''}_{\mathrm{cmp}}(\xi)$ have precisely the same temporal structure, in the sense that they have discrete transitions at the same points. But we know that (if $\Xi \neq \emptyset$ then) we can always pick ξ' from Ξ' with discrete transitions everywhere we want (the same applies to Ξ''.)

Remark 3.5. In the system h $\times$ c1 we have two attributes T_1, T_2 that evolve independently of each other. Also, the actions $cool_1$ and $cool_2$ have no relation whatsoever, and so may happen together or alone.

[5] Note that the product $A' \times A''$ is computed in *PFin*.
[6] If we had non time divergent computations, this might not be the case.

3.2 Restriction

Consider two systems s' and s'' that we want to interconnect. In the system $s' \times s''$ there is no interaction between the two component systems. Two typical forms of interaction are attribute and action sharing. These can be expressed as restrictions on the signature of $s' \times s''$. If we express these restrictions by 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.

Proposition 3.6. Let $\sigma : \Sigma \to \Sigma'$ be a signature morphism and $s = \langle \Sigma, \Xi \rangle$ be a Σ-computation system. Then the *Sys* morphism $\sigma : \overrightarrow{\sigma}_{\text{cs}}^{-1}(s) \to s$ with $\overrightarrow{\sigma}_{\text{cs}}^{-1}(s) = \langle \Sigma', \overrightarrow{\sigma}_{\text{cmp}}^{-1}(\Xi) \rangle$ is cartesian by S for σ on s:

$$\langle \Sigma, \Xi \rangle \xleftarrow{\sigma} \langle \Sigma', \overrightarrow{\sigma}_{\text{cmp}}^{-1}(\Xi') \rangle \qquad\qquad \Sigma \xrightarrow{\sigma} \underline{\Sigma'}$$

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

Example 3.7. Consider again the free aggregation of the heater and the first controller. Suppose that we want to share the attribute T and the cool action, thus obtaining a new system $h||c1$.[7] *That is, we want a system with signature $\Sigma_{h||c1} = \langle \{T\}, V_{h+c1}, \{heat, cool\} \rangle$. If we consider a signature morphism $\sigma : \Sigma_h + \Sigma_{c1} \to \Sigma_{h||c1}$ such that $\overrightarrow{\sigma}_{\text{att}} = \{T_1 \mapsto T, T_2 \mapsto T\}$, $\overleftarrow{\sigma}_{\text{vtx}} = id_{V_{h+c1}}$ and $\overleftarrow{\sigma}_{\text{act}} = \{heat \mapsto heat_1, cool \mapsto \langle cool, cool \rangle\}$ then we have that $h||c1$ is $\overrightarrow{\sigma}_{\text{cs}}^{-1}(h \times c1)$.*

In this system, the T attributes of the heater and the controller are seen as just one attribute. Only the phases of h and $c1$ where T follows the same history appear in $h||c1$; the others phases are ruled out. Also, the two systems synchronize on the cool action; thus, the orders issued by the controller are followed by the heater.

Note that in prop. 3.6 no restriction is imposed on σ. Namely, the restriction operation can also be applied to the set of vertices.

3.3 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 represented at the signature level and carried to the semantic level in a similar way to the restriction operation.

Proposition 3.8. Let $\sigma : \Sigma \to \Sigma'$ be a signature morphism and $s' = \langle \Sigma', \Xi' \rangle$ be a Σ'-computation system. Then the *Sys* morphism $\sigma : s' \to \overrightarrow{\sigma}_{\text{cs}}(s')$ with $\overrightarrow{\sigma}_{\text{cs}}(s') = \langle \Sigma, \overrightarrow{\sigma}_{\text{cmp}}(\Xi')^{\equiv} \rangle$ is cocartesian by S for σ on s':

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

$$\langle \Sigma, \overrightarrow{\sigma}_{\text{cmp}}(\Xi')^{\equiv} \rangle \xleftarrow{\sigma} \langle \Sigma', \Xi' \rangle \qquad\qquad \underline{\Sigma} \xrightarrow{\sigma} \Sigma'$$

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

Example 3.9. The first controller is an abstraction of the second one (or, dually, the second controller is a refinement or implementation of the first one). If we consider a signature morphism $\sigma : \Sigma_{c1} \to \Sigma_{c2}$ such that $\overrightarrow{\sigma}_{\text{att}} = \{T \mapsto T\}$ (C is hidden,) $\overleftarrow{\sigma}_{\text{vtx}} = \{on \mapsto v, off \mapsto v\}$ (c_on and c_off are collapsed) and $\overleftarrow{\sigma}_{\text{act}} = \{cool \mapsto cool\}$ ($\overleftarrow{\sigma}_{\text{act}}(start)$ is undefined, that is, $start$ is hidden) then we have that $c1 = \overrightarrow{\sigma}_{\text{cs}}(c2)$.

In this case, we hide the auxiliary attribute C but retain the dynamic behavior of the system. Namely, the controller still turns off the heater within .25 time units if the temperature rises above 1/2.

4 Concluding remarks

The main goal of this paper has been achieved: we were able to provide a categorial characterization of free aggregation, restriction and abstraction 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 symbol sets.

Also, 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.

This work forms the semantic basis upon which a categorial theory of compositional specification and verification of hybrid systems can be constructed (results of work in this direction can be found in [6].)

A Appendix

Definition A.1. A *category* is a tuple $\langle \mathcal{O}, \mathcal{M}, \circ, id \rangle$ where

- $\mathcal{O}$ is a class;
- $\mathcal{M} = \{\mathcal{M}_{AB}\}_{A,B \in \mathcal{O}}$ is a family of classes (for $f \in \mathcal{M}_{AB}$ we write $f : A \to B$);
- $\circ = \{\circ_{ABC}\}_{A,B,C} \in \mathcal{O}$ with $\circ_{ABC} : \mathcal{M}_{AB} \times \mathcal{M}_{BC} \to \mathcal{M}_{AC}$;
- $id = \{id_A\}_{A \in \mathcal{O}}$ with $id_A \in \mathcal{M}_{AA}$;

and such that[8]

[8] Subscripts in "$\circ$" have been omitted for clarity.

- $(f \circ g) \circ h = f \circ (g \circ h)$ for each $h : A \to B, g : B \to C, f : C \to D$;
- $f \circ id_A = id_B \circ f = f$ for each $f : A \to B$.

The elements of $\mathcal{O}$ are called *objects*, $\mathcal{M}_{AB}$ is the class of *morphisms* from A to B and id_A is the *identity morphism* for object A. We also use $|\mathcal{C}|$ for denoting the class $\mathcal{O}$.

Example A.2. Fin is the category where $\mathcal{O}$ is the class of finite sets, $\mathcal{M}_{AB}$ is the set of maps from A to B, $g \circ h$ is the usual composition of maps and id_A is the identity map from A to A. PFin is the same as Fin, except for the fact that $\mathcal{M}_{AB}$ is the set of partial maps from A to B (a partial map f from A to B is represented as $f : A \rightharpoonup B$.)

Prop/Definition A.3. Let $\mathcal{C}, \mathcal{C}'$ be categories. Then $\mathcal{C} \times \mathcal{C}' = \langle \mathcal{O} \times \mathcal{O}', \mathcal{M}^\times, \circ^\times, id^\times \rangle$ with $\mathcal{M}^\times_{\langle A,A'\rangle\langle B,B'\rangle} = \mathcal{M}_{AB} \times \mathcal{M}'_{A'B'}$, $\langle f, f'\rangle \circ^\times \langle g, g'\rangle = \langle f \circ g, f' \circ' g'\rangle$ and $id^\times_{AA'} = \langle id_A, id'_{A'}\rangle$ is also a category (called the product of $\mathcal{C}$ and $\mathcal{C}'$.)

Prop/Definition A.4. Let $\mathcal{C}$ be a category. Then $\mathcal{C}^{op} = \langle \mathcal{O}, \mathcal{M}^{op}, \circ^{op}, id\rangle$ where $\mathcal{M}^{op}_{BA} = \mathcal{M}_{AB}$ and $f \circ^{op} g = g \circ f$ is also a category (called the *dual* of $\mathcal{C}$.)

Definition A.5. A *product* of two objects A' and A'' in a category $\mathcal{C}$ is a triple $\langle A, \pi', \pi''\rangle$, where $A \in \mathcal{O}$, $\pi' : A \to A'$ and $\pi'' : A \to A''$, such that for any object B and morphisms $f' : B \to A'$ and $f'' : B \to A''$ there is a unique morphism $h : B \to A$ such that $\pi' \circ h = f'$ and $\pi'' \circ h = f''$:

$$\begin{tikzcd}
A' & A \arrow[l, "\pi'"'] \arrow[r, "\pi''"] & A'' \\
& B \arrow[ul, "f'"] \arrow[u, dashed, "h"] \arrow[ur, "f''"'] &
\end{tikzcd}$$

By duality we obtain the notion of *coproduct*.

Example A.6. Let A, B be finite sets. The cartesian product $A \times B$ together with the usual projection maps $\pi_A : A \times B \to A$, $\pi_B : A \times B \to B$ is a product of A and B in Fin. The disjoint union $A + B$ together with the underlying injection maps $i_A : A \to A + B$, $i_B : B \to A + B$ is a coproduct of A and B in Fin. The set $A + B + A \times B$ together with the (partial) projection maps $\pi_A : (A + B + A \times B) \rightharpoonup A$, $\pi_B : (A + B + A \times B) \rightharpoonup B$ is a product of A and B in PFin.

Definition A.7. Given two categories $\mathcal{C}$ and $\mathcal{C}'$, a functor $F : \mathcal{C} \to \mathcal{C}'$ consists of a map $F^{\mathcal{O}} : \mathcal{O} \to \mathcal{O}'$ and a family $F^{\mathcal{M}} = \{F^{\mathcal{M}}_{AB}\}_{A,B \in \mathcal{O}}$ with $F^{\mathcal{M}}_{AB} : \mathcal{M}_{AB} \to \mathcal{M}'_{F^{\mathcal{O}}(A)F^{\mathcal{O}}(B)}$ such that:

- $F^{\mathcal{M}}_{AC}(f \circ g) = F^{\mathcal{M}}_{BC}(f) \circ' F^{\mathcal{M}}_{AB}(g)$ for each $g : A \to B, f : B \to C$;
- $F^{\mathcal{M}}(id_A) = id'_{F^{\mathcal{O}}(A)}$ for each $A \in \mathcal{O}$;

We usually omit the superscripts in $F^{\mathcal{O}}, F^{\mathcal{M}}$.

Definition A.8. Let $F : \mathcal{C} \to \mathcal{C}'$ be a functor, $f' : A' \to B'$ a morphism in $\mathcal{C}'$ and A an object of $\mathcal{C}$ such that $F(A) = A'$. A morphism $f : A \to B$ in $\mathcal{C}$ is said to be cartesian by F for f' on A iff $F(f) = f'$ and for each pair of morphisms $g : D \to B$ and $h' : D' \to A'$ such that $D' = F(D)$ and $F(g) = f' \circ h'$ there is a unique $h : D \to A$ such that $F(h) = h'$ and $f \circ h = g$:

$$\begin{tikzcd} A \arrow[r, "f"] & B \\ D \arrow[u, dashed, "h"] \arrow[ur, "g"'] & \end{tikzcd} \qquad \begin{tikzcd} A' \arrow[r, "f'"] & B' \\ D' \arrow[u, "h'"] \arrow[ur, "F(g)"'] & \end{tikzcd}$$

By duality we obtain the notion of *cocartesian morphism*.

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. 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.
4. 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.
5. 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.
6. H. Lourenço and A. Sernadas. An institution of hybrid systems. Research report, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1999. Presented at WADT'99. Submitted for publication.
7. 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.
8. A. Sernadas, J. F. Costa, and C. Sernadas. An institution of object behaviour. In H. Ehrig and F. Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 337–350. Springer-Verlag, 1994.
9. A. Sernadas, C. Sernadas, and C. Caleiro. Denotational semantics of object specification. *Acta Informatica*, 35:729–773, 1998.
10. A. Sernadas, C. Sernadas, and J. F. Costa. Object specification logic. *Journal of Logic and Computation*, 5(5):603–630, 1995.
11. Glynn Winskel and Mogens Nielsen. *Semantics and Logics of Computation*, chapter Categories in Concurrency, pages 299–354. Publications of the Newton Institute. 1997.