Under consideration for publication in Formal Aspects of Computing

HYPE: Hybrid modelling by composition of flows

Vashti Galpin[1], Luca Bortolussi[2] and Jane Hillston[1]

[1]Laboratory for Foundations of Computer Science, University of Edinburgh
[2]Department of Maths and Computer Science, University of Trieste

Abstract. Hybrid systems are manifest in both the natural and the engineered world, and their complex nature, mixing discrete control and continuous evolution, make it difficult to predict their behaviour. In recent years several process algebras for modelling hybrid systems have appeared in the literature, aimed at addressing this problem. These all assume that continuous variables in the system are modelled monolithically, often with differential equations embedded explicitly in the syntax of the process algebra expression. In HYPE an alternative approach is taken which offers finer-grained modelling with each flow or influence affecting a variable modelled separately. The overall behaviour then emerges as the composition of flows.

In this paper we give a detailed account of the HYPE process algebra, its semantics, and its use for verification of systems. We establish both syntactic conditions (well-definedness) and operational restrictions (well-behavedness) to ensure reasonable behaviour in HYPE models. Furthermore we consider how the equivalence relation defined for HYPE relates to other relations previously proposed in the literature, demonstrating that our fine-grained approach leads to a more discriminating notion of equivalence. We present the HYPE model of a standard hybrid system example, both establishing that our approach can reproduce the previously obtained results and demonstrating how our compositional approach supports variations of the problem in a straightforward and flexible way.

Keywords: hybrid systems; process algebra; flows; compositionality; bisimulation

1. Introduction

Predominantly, process algebras have been used to model discrete systems. Their style of modelling, in terms of agents and actions, matches well with a discrete event view of the world. This form of modelling encompasses a wide variety of systems as witnessed by the wide-range of applications which have emerged for process algebra modelling. Nevertheless there are situations in which this discrete view is not entirely satisfactory. For example there may be inherent properties of the system which vary continuously or because a fluid abstraction (in which discrete variables are treated as continuous) of some aspects of the system will bring benefits for modelling, or analysis, or both. We term a *hybrid system* one which exhibits both discrete

Correspondence and offprint requests to: Vashti Galpin, Informatics Forum, 10 Crichton Street, Edinburgh EH8 9AB. e-mail: Vashti.Galpin@ed.ac.uk and continuous behaviour. Given the success and attractiveness of process algebra modelling it is appealing to investigate whether the same compositional approach can be applied to such hybrid systems.

Hybrid behaviour arises in a variety of systems, both engineered and natural. Consider a thermostatically controlled heater. The continuous variable is air temperature, and the discrete events are the switching on and off of the heater by the thermostat in response to the air temperature. Another example would be a genetic regulatory network, such as the Repressilator [EL00], in which genes can be switched on or off by interactions with their environment (more precisely, with transcription factor proteins). The behaviour of such systems can be regarded as a collection of sets of ODEs, the discrete events shifting the dynamic behaviour from the control of one set of ODEs to another and possibly modifying the values of the variables immediately after the event. This is the approach taken with hybrid automata [Hen96].

HYPE is a novel process algebra for modelling hybrid systems. A hybrid system consists of values which change continuously over time with respect to specific dynamics. Discrete events can cause discontinuous jumps in these values after which different dynamics may come into effect. These events can be triggered by conditions on the continuously changing values. When modelling a hybrid system we need to capture both the dynamics and the events which cause the system to move between different dynamics. The novelty of HYPE lies in how it captures the continuous dynamics of a system, its modular style of definition and the separation of a discrete controller considered in parallel to the system under study. Unlike previous process algebras for hybrid systems, HYPE captures behaviour at a fine-grained level, composing distinct flows or influences. The dynamic behaviour then emerges, via the semantics of the language, from these compositional elements. For the continuous aspects, we are inspired by the fluid flow semantics of PEPA models [Hil05, TGH10] which approximates the behaviour of large numbers of discrete components with a set of ordinary differential equations (ODEs). For the discrete aspects of the model, the controller imposes sequencing on events and having this separated readily supports investigations into how the same system may behave under the influence of distinct controllers. Note that although the controller is specified as a distinct component of the system to aid modularity of definition, its events are triggered by the continuous values of the system.

Process algebras have the advantage of being compositional hence models are built out of subcomponents and we fully exploit this feature in HYPE. Existing process algebras for hybrid systems include ACP^{srt}_{hs} [BM05], hybrid χ [vBMR$^+$06], ϕ-calculus [RS03] and HyPA [CR05]. In [Kha06], Khadim shows substantial differences in the approaches taken by these process algebras relating to syntax, semantics, discontinuous behaviour, flow-determinism, theoretical results and availability of tools. However, they are all similar in their approach in that the dynamic behaviour of each subcomponent must be fully described with the ODEs for the subcomponent given explicitly in the syntax of the process algebra, before the model can be constructed. Furthermore an evaluation of continuous variables is inherent in the notion of state.

We aim for a finer-grained approach where each subcomponent is built up from a number of flows and hence the ODEs are only obtained once the model is constructed. By flow, we mean something that has an influence on a quantity of interest. For example, in a tank with two inlets and an outlet, both the inlets and the outlet influence the tank level, hence here we would identify three separate flows. The continuous part of the system is represented by the appropriate variables and the change over time of a given variable is determined by a number of active influences which represent flows and are additive in nature. Our approach also differs in that we explicitly require a controller that consists only of events.

We believe that the use of flows as the basic elements of model construction has advantages such as ease and simplification of modelling. This approach assists the modeller in allowing them to identify smaller or local descriptions of the model and then to combine these descriptions to obtain the larger system. The explicit controller also helps to separate modelling concerns.

In this paper we present a full account of the HYPE language, including both operational and hybrid semantics. The two are distinct because in HYPE the *state* of the system does not include an evaluation of the continuously varying variables; instead it captures the influences which are currently in operation within the system. Thus each state corresponds to one dynamic regime, captured by a set of ODEs in the hybrid semantics. We also establish an equivalence semantics for the language and consider how this compares with the existing bisimulations for hybrid process algebras and hybrid automata which have appeared in the literature. In particular we consider the bisimulations defined in [BM05] for ACP^{srt}_{hs} and the U-bisimulation of [AMP$^+$03] for hybrid automata, showing that the notion of bisimulation in HYPE is more discriminating than that for hybrid automata.

This paper is a revised and extended version of the paper which appeared in [GBH09]. Here we have substantially revised the definition of what it means for a HYPE model to be well-defined and furthermore considered when a model may be considered to be *well-behaved* and how to check for such a property. A model which is not well-behaved may exhibit *instantaneous Zeno behaviour*, completing an infinite number of events in a single time instant. The work on comparing bisimulations is extended and here, additionally, we compare the expressive power of HYPE with that of hybrid automata at the level of composition of models. Finally a new example of HYPE modelling that considers a train gate controller [AHH96], is presented.

The structure of the rest of the paper is as follows. In the next section we introduce our syntax for hybrid systems, explaining its components. In the following sections, we present the operational semantics and the hybrid semantics, explaining how we go from the notion of state to the ODEs which describe the system dynamics. We then identify the subclass of HYPE models which we consider to be well-defined and discuss how to check that a HYPE model is well-behaved. We also compare composition of models in HYPE and hybrid automata. We define a notion of bisimulation in Section 8 and compare it with previous bisimulations for hybrid languages from the literature. To illustrate the power of HYPE, in Section 10 we present an example, a train gate controller. The remaining sections discuss related work, conclusions and future research.

2. HYPE Definition

This section will present HYPE by way of a running example of the temperature control system of an orbiting spacecraft. As the orbiter travels around the earth, it needs to regulate its temperature to remain within operational limits. It has insulation but needs to use a heater at low temperatures and at high temperatures it can erect a shade to reduce temperatures. The modelling spirit of HYPE focusses on flows. In our example, we identify four flows affecting the temperature. One is due to thermodynamic cooling, one is due to the heater, one is due to the heating effect of the sun and one is due to the cooling effect of the shade. The strength and form of a flow are modified by events. We first define the heater which can be on or off.

$$\mathit{Heat} \stackrel{\mathit{def}}{=} \underline{\mathrm{on}} : (h, r_h, \mathit{const}).\mathit{Heat} + \underline{\mathrm{off}} : (h, 0, \mathit{const}).\mathit{Heat} + \underline{\mathrm{init}} : (h, 0, \mathit{const}).\mathit{Heat}$$

This is a *summation* of prefixes. Each prefix consists of two actions. *Events* ($\underline{\mathrm{a}} \in \mathcal{E}$) are actions which happen instantaneously and trigger discrete changes. They can be caused by a controller or happen randomly and can depend on the global state of the system, specifically values of variables. In the example, the events are $\underline{\mathrm{on}}$, switching on; $\underline{\mathrm{off}}$, switching off and $\underline{\mathrm{init}}$, the initialisation event.

Activities ($\alpha \in \mathcal{A}$) are *influences* on the evolution of the continuous part of the system and define flows. An activity is defined as a triple and can be parameterised by a set of variables, $\alpha(\mathcal{W}) = (\iota, r, I(\mathcal{W}))$. This triple consists of an *influence name* ι, a rate of change (or *influence strength*) r and an *influence type name* $I(\mathcal{W})$ which describes how that rate is to be applied to the variables involved, or the actual form of the flow[1]. In *Heat*, there are two distinct activities, (h, r_h, const) and $(h, 0, \mathit{const})$. The first one gives the effect of the heater being on: the influence name is h which represents the influence from the heater on the orbiter's temperature, r_h is the strength of the heater, and it is associated with the function called *const*. The second captures the effect of the heater being off. It again affects influence h, it has strength 0 and the form it takes is *const*.

The interpretation of influence types will be specified separately, so that experimentation with different functional forms of the heating flow can occur without modifying the subcomponent. Hence, in HYPE we separate the description of the logical structure of flows from their mathematical interpretation. We now describe the other flows for the example.

$$\begin{aligned}
\mathit{Shade} &\stackrel{\mathit{def}}{=} \underline{\mathrm{up}} : (d, -r_d, \mathit{const}).\mathit{Shade} + \underline{\mathrm{down}} : (d, 0, \mathit{const}).\mathit{Shade} + \underline{\mathrm{init}} : (d, 0, \mathit{const}).\mathit{Shade} \\
\mathit{Sun} &\stackrel{\mathit{def}}{=} \underline{\mathrm{light}} : (s, r_s, \mathit{const}).\mathit{Sun} + \underline{\mathrm{dark}} : (s, 0, \mathit{const}).\mathit{Sun} + \underline{\mathrm{init}} : (s, 0, \mathit{const}).\mathit{Sun} \\
\mathit{Cool}(K) &\stackrel{\mathit{def}}{=} \underline{\mathrm{init}} : (c, -1, \mathit{linear}(K)).\mathit{Cool}(K)
\end{aligned}$$

The only event in the last definition is the initialisation event, as once cooling is in effect it does not change. The influence name is c and its strength is -1. The influence type $\mathit{linear}(K)$ will be interpreted as a linear function of its variable K which represents the temperature of the system. We also need to model the change in sunlight. We do this by keeping track of time with the following component (which is kept simple for

[1] For convenience, we will use I for $I(\mathcal{W})$ when $\mathcal{W}$ can be inferred.

reasons of space).

$$\mathit{Time} \stackrel{\mathit{def}}{=} \underline{\mathrm{light}} : (t, 1, \mathit{const}).\mathit{Time} + \underline{\mathrm{dark}} : (t, 1, \mathit{const}).\mathit{Time} + \underline{\mathrm{init}} : (t, 1, \mathit{const}).\mathit{Time}$$

These subcomponents can be combined to give the overall uncontrolled system[2].

$$\mathit{Sys} \stackrel{\mathit{def}}{=} (((\mathit{Heat} \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Shade}) \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Sun}) \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Cool}(K)) \underset{\{\underline{\mathrm{init}},\underline{\mathrm{light}},\underline{\mathrm{dark}}\}}{\bowtie} \mathit{Time}$$

Here $\underset{L}{\bowtie}$ represents parallel cooperation. L is the set of events over which synchronisation must occur. Events not in L can occur independently. *Sys* is called the *uncontrolled system* because all events are possible and no causal or temporal constraints have been imposed yet. For instance, we need to specify that the heater can only be switched off after it has been switched on. We now give controllers/sequencers for the heater, the shade and the effect of the sun. Note that these correspond to the starting states set by $\underline{\mathrm{init}}$ events. For example, in *Heat*, $\underline{\mathrm{init}}$ sets the heater to off (the influence strength is 0), therefore $\underline{\mathrm{on}}$ is the next event to occur.

$$\begin{aligned}
\mathit{Con}_h &\stackrel{\mathit{def}}{=} \underline{\mathrm{on}}.\underline{\mathrm{off}}.\mathit{Con}_h\\
\mathit{Con}_d &\stackrel{\mathit{def}}{=} \underline{\mathrm{up}}.\underline{\mathrm{down}}.\mathit{Con}_d\\
\mathit{Con}_s &\stackrel{\mathit{def}}{=} \underline{\mathrm{light}}.\underline{\mathrm{dark}}.\mathit{Con}_s\\
\mathit{Con} &\stackrel{\mathit{def}}{=} \mathit{Con}_h \underset{\emptyset}{\bowtie} \mathit{Con}_d \underset{\emptyset}{\bowtie} \mathit{Con}_s
\end{aligned}$$

Controllers only have event prefixes. Their behaviour is affected by the state of the system through event conditions which determine when events occur. The controlled system is constructed from cooperation of the controller and the uncontrolled system and the controller must be prefixed by the initialisation event $\underline{\mathrm{init}}$. For the example, the controlled system is described by

$$\mathit{TempCtrl} \stackrel{\mathit{def}}{=} \mathit{Sys} \underset{M}{\bowtie} \underline{\mathrm{init}}.\mathit{Con} \quad \text{with} \quad M = \{\underline{\mathrm{init}}, \underline{\mathrm{on}}, \underline{\mathrm{off}}, \underline{\mathrm{up}}, \underline{\mathrm{down}}, \underline{\mathrm{light}}, \underline{\mathrm{dark}}\}.$$

This has defined the structure of our system but we require additional definitions to capture further details. We need to link each influence with an actual variable. This is done using the function iv. For the example, $iv(h) = iv(s) = iv(d) = iv(c) = K$ where K is the actual variable for the temperature of the orbiter, and $iv(t)=T$, the variable for time. Note that an influence can only be associated with one variable, in agreement with the interpretation of influences as flows. This does not mean that only one variable can be affected by an event. For another variable Y that was also affected by the heater being on (power consumption, say), we could define a subcomponent with a prefix $\underline{\mathrm{on}}$:(p, r, I) and set $iv(p) = Y$.

We define the influence types as $[\![\mathit{const}]\!] = 1$ and $[\![\mathit{linear}(X)]\!] = X$. The influence types are used to describe how influences are affected by variables in the system. The type *const* is used when there is no dependency and $\mathit{linear}(X)$ is used when the influence depends linearly on the value of the variable X. Non-linear dependencies can also be defined through this mechanism. Examples of non-linear ODEs occur in the description of the motion of a pendulum and when mass action is used in biological modelling.

Finally, we define what triggers an event, and how it affects variables, with the function ec. Each event condition consists of an activation condition and a reset.

The activation condition is a positive boolean formula (one without negation) containing equalities and nonstrict inequalities on system variables or the symbol $\perp$. We choose this specific restriction since it guarantees that the set represented by such a formula is a closed set, being the finite union and intersection of closed sets. We require closed sets so that the first instant in which a guard becomes true is defined in a consistent way.

The variable reset is a conjunction of equality predicates of the form $V' = f(\mathcal{V})$ where V' denotes the new value that V will have after the reset, while V denotes the previous value. Resets of the form $V' = V$ can be left implicit, and the identity reset is denoted *true*. Event conditions are associated with events by the function ec. The example has the following event conditions.

[2] Note that in HYPE uncontrolled systems all possible controllable behaviour is exhibited, awaiting a suitable controller to instigate appropriate triggers. This is in contrast to, say, an embedded system, where the uncontrolled system would be aspects of behaviour outside the scope of the controller.

$$\begin{array}{rclrcl} ec(\underline{\text{init}}) & = & (true, (K' = k_0 \wedge T' = 0)) & & & \\ ec(\underline{\text{off}}) & = & ((K \geq k_1), true) & ec(\underline{\text{on}}) & = & (K \leq k_2, true) \\ ec(\underline{\text{up}}) & = & ((K \geq k_3), true) & ec(\underline{\text{down}}) & = & (K \leq k_4, true) \\ ec(\underline{\text{light}}) & = & ((T = 12), true) & ec(\underline{\text{dark}}) & = & (T = 24, T' = 0) \end{array}$$

where the $k_i (1 \leq i \leq 4)$ are fixed temperature values. Most events are urgent – the event must occur as soon as its event condition is satisfied. For events that can happen randomly such as breakdowns, we introduce a special event condition $\perp$ which means that the event can happen nondeterministically in time (sometimes termed lazy jumps)[3]. In the example, the $\underline{\text{init}}$ event has an associated event condition of *true* and so this must happen immediately and $\underline{\text{light}}$ happens when 12 hours have passed. The event $\underline{\text{init}}$ has a reset that defines the values of the variables and $\underline{\text{on}}$ has a reset of *true* meaning that no values are changed. Figure 1 presents all components of the HYPE model of the orbiter in one place. For a comparison of different process algebras for modelling hybrid systems (other than HYPE) see [Kha06, TCT01] and we discuss this further in Section 9.

In the preceding informal discussion, we have introduced the main constituents of a HYPE model including the combination of flow components with a controller component, variables, association between influences and variables, conditions that specify when events occur, and definitions for the influence type functions. To understand the dynamics of this system, we need to derive ODEs to describe how the variables change over time. To do this we present operational semantics that define the behaviour of our controlled system. Before that we present the formal definition of HYPE. In the rest of the paper, $\mathcal{V}$ is a set or tuple of variables with $\mathcal{W} \subseteq \mathcal{V}$ denoting an arbitrary subset of $\mathcal{V}$.

Definition 2.1. A *controlled system* is constructed as follows.

- *Subcomponents* are defined by $C_s(\mathcal{W}) \stackrel{\text{def}}{=} S$, where C_s is the *subcomponent name* and S satisfies the grammar $S' ::= \underline{\text{a}} : \alpha.C_s(\mathcal{W}) \mid S' + S'$ ($\underline{\text{a}} \in \mathcal{E}$, $\alpha \in \mathcal{A}$), with the free variables of S in $\mathcal{W}$.
- *Components* are defined by $C(\mathcal{W}) \stackrel{\text{def}}{=} P$, where C is the *component name* and P satisfies the grammar $P' ::= C_s(\mathcal{W}) \mid P' \underset{L}{\bowtie} P'$, with the free variables of P in $\mathcal{W}$ and $L \subseteq \mathcal{E}$.
- An *uncontrolled system* Σ is defined according to the grammar $\Sigma' ::= C_s(\mathcal{W}) \mid C(\mathcal{W}) \mid \Sigma' \underset{L}{\bowtie} \Sigma'$, where $L \subseteq \mathcal{E}$ and $\mathcal{W}$ is a set of system variables.
- *Controllers* only have events: $M ::= \underline{\text{a}}.M \mid 0 \mid M + M$ with $\underline{\text{a}} \in \mathcal{E}$ and $Con ::= M \mid Con \underset{L}{\bowtie} Con$ with $L \subseteq \mathcal{E}$.
- A *controlled system* is $ConSys ::= \Sigma \underset{L}{\bowtie} \underline{\text{init}}.Con$ where $L \subseteq \mathcal{E}$. The set of controlled systems is $\mathcal{C}_{Sys}$.

A controlled system together with the appropriate sets and functions, gives a HYPE model.

Definition 2.2. A *HYPE model* is a tuple $(ConSys, \mathcal{V}, IN, IT, \mathcal{E}, \mathcal{A}, ec, iv, EC, ID)$ where

- *ConSys* is a controlled system as defined above.
- $\mathcal{V}$ is a finite set of variables[4].
- *IN* is a set of influence names and *IT* is a set of influence type names.
- $\mathcal{E}$ is a set of events of the form $\underline{\text{a}}$ and $\underline{\text{a}}_i$.
- $\mathcal{A}$ is a set of activities of the form $\alpha(\mathcal{W}) = (\iota, r, I(\mathcal{W})) \in (IN \times \mathbb{R} \times IT)$.
- $ec : \mathcal{E} \to EC$ maps events to event conditions. Event conditions are pairs of formulas, the first with free variables in $\mathcal{V}$ and the second with free variables in $\mathcal{V} \cup \mathcal{V}'$.
- $iv : IN \to \mathcal{V}$ maps influence names to variable names.
- EC is a set of event conditions.
- ID is a collection of definitions consisting of a real-valued function for each influence type name $[\![I(\mathcal{W})]\!] = f(\mathcal{W})$ where the variables in $\mathcal{W}$ are from $\mathcal{V}$.
- $\mathcal{E}$, $\mathcal{A}$, IN and IT are pairwise disjoint.

[3] We have not done so here but it is possible to have events that are probabilistic rather than nondeterministic in time. Tuffin *et al* [TCT01] use a value drawn from a exponential distribution for the time until the next event and we consider the use of stochastic delays in [BGH10a].

[4] In [GBH09], a set of formal variables $\mathcal{X}$ was also defined but with experience, it became clear that these variables were not required and have been dropped.

$$\begin{array}{rcl}
\mathit{Heat} & \stackrel{\mathit{def}}{=} & \underline{\mathrm{on}} : (h, r_h, \mathit{const}).\mathit{Heat} + \underline{\mathrm{off}} : (h, 0, \mathit{const}).\mathit{Heat} + \underline{\mathrm{init}} : (h, 0, \mathit{const}).\mathit{Heat} \\
\mathit{Shade} & \stackrel{\mathit{def}}{=} & \underline{\mathrm{up}} : (d, -r_d, \mathit{const}).\mathit{Shade} + \underline{\mathrm{down}} : (d, 0, \mathit{const}).\mathit{Shade} + \underline{\mathrm{init}} : (d, 0, \mathit{const}).\mathit{Shade} \\
\mathit{Sun} & \stackrel{\mathit{def}}{=} & \underline{\mathrm{light}} : (s, r_s, \mathit{const}).\mathit{Sun} + \underline{\mathrm{dark}} : (s, 0, \mathit{const}).\mathit{Sun} + \underline{\mathrm{init}} : (s, 0, \mathit{const}).\mathit{Sun} \\
\mathit{Cool}(K) & \stackrel{\mathit{def}}{=} & \underline{\mathrm{init}} : (c, -1, \mathit{linear}(K)).\mathit{Cool}(K) \\
\mathit{Time} & \stackrel{\mathit{def}}{=} & \underline{\mathrm{light}} : (t, 1, \mathit{const}).\mathit{Time} + \underline{\mathrm{dark}} : (t, 1, \mathit{const}).\mathit{Time} + \underline{\mathrm{init}} : (t, 1, \mathit{const}).\mathit{Time}
\end{array}$$

$$\mathit{Sys} \stackrel{\mathit{def}}{=} (((\mathit{Heat} \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Shade}) \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Sun}) \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Cool}(K)) \underset{\{\underline{\mathrm{init}},\underline{\mathrm{light}},\underline{\mathrm{dark}}\}}{\bowtie} \mathit{Time}$$

$$\begin{array}{rcl}
\mathit{Con}_h & \stackrel{\mathit{def}}{=} & \underline{\mathrm{on}}.\underline{\mathrm{off}}.\mathit{Con}_h \\
\mathit{Con}_d & \stackrel{\mathit{def}}{=} & \underline{\mathrm{up}}.\underline{\mathrm{down}}.\mathit{Con}_d \\
\mathit{Con}_s & \stackrel{\mathit{def}}{=} & \underline{\mathrm{light}}.\underline{\mathrm{dark}}.\mathit{Con}_s \\
\mathit{Con} & \stackrel{\mathit{def}}{=} & \mathit{Con}_h \underset{\emptyset}{\bowtie} \mathit{Con}_d \underset{\emptyset}{\bowtie} \mathit{Con}_s
\end{array}$$

$$\mathit{TempCtrl} \stackrel{\mathit{def}}{=} \mathit{Sys} \underset{M}{\bowtie} \underline{\mathrm{init}}.\mathit{Con} \quad \text{with} \quad M = \{\underline{\mathrm{init}}, \underline{\mathrm{on}}, \underline{\mathrm{off}}, \underline{\mathrm{up}}, \underline{\mathrm{down}}, \underline{\mathrm{light}}, \underline{\mathrm{dark}}\}$$

$$\mathit{iv}(t) = T \qquad \mathit{iv}(h) = \mathit{iv}(s) = \mathit{iv}(d) = \mathit{iv}(c) = K$$

$$\begin{array}{rclrcl}
\mathit{ec}(\underline{\mathrm{init}}) & = & (\mathit{true}, (K' = k_0 \wedge T' = 0)) & & & \\
\mathit{ec}(\underline{\mathrm{off}}) & = & ((K \geq k_1), \mathit{true}) & \mathit{ec}(\underline{\mathrm{on}}) & = & (K \leq k_2, \mathit{true}) \\
\mathit{ec}(\underline{\mathrm{up}}) & = & ((K \geq k_3), \mathit{true}) & \mathit{ec}(\underline{\mathrm{down}}) & = & (K \leq k_4, \mathit{true}) \\
\mathit{ec}(\underline{\mathrm{light}}) & = & ((T = 12), \mathit{true}) & \mathit{ec}(\underline{\mathrm{dark}}) & = & (T = 24, T' = 0)
\end{array}$$

$$[\![\mathit{const}]\!] = 1 \qquad [\![\mathit{linear}(X)]\!] = X$$

Fig. 1. HYPE model of the orbiter

In the sequel, we will use P, Q, ... for controlled systems (rather than *ConSys*) and for HYPE models. When referring to a HYPE model, the meta-variable for the controlled system will be used, such as P, and the tuple will be implied. In the case of two HYPE models P and Q without reference to the tuple, we will assume two implied tuples with identical elements except for the first elements. The syntax of HYPE is moderately complex because hybrid systems are complex structures displaying both continuous and discrete behaviour. Despite the complexity, the example shows how it is straightforward to construct a HYPE model once flows and the controller are identified. Additionally the modularity of definition helps the modeller to check that everything is specified.

3. Operational Semantics

We now introduce the behaviour of HYPE models by defining an appropriate semantics. We start by considering the events that can happen, and construct a labelled transition system describing this behaviour. To define the operational semantics, a notion of state is required. States record for each influence, its current strength and influence type. Each configuration will have an associated state which will capture the continuous behaviour in that configuration.

Definition 3.1. An *operational state* of the system is a function $\sigma : \mathit{IN} \to (\mathbb{R} \times \mathit{IT})$. The set of all operational states is $\mathcal{S}$. A *configuration* consists of a controlled system together with an operational state $\langle \mathit{ConSys}, \sigma \rangle$ and the set of configurations is $\mathcal{F}$.

We use 'state' for 'operational state' in the rest of this document. For convenience, states may be written as a set of triples of the form $(\iota, r, I(\mathcal{W}))$ where there is at most most one triple containing ι. This is the same

Prefix with influence:
$$\frac{}{\langle \underline{a} : (\iota, r, I).E, \sigma\rangle \xrightarrow{\underline{a}} \langle E, \sigma[\iota \mapsto (r, I)]\rangle}$$

Prefix without influence:
$$\frac{}{\langle \underline{a}.E, \sigma\rangle \xrightarrow{\underline{a}} \langle E, \sigma\rangle}$$

Choice:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle E + F, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle} \qquad \frac{\langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}{\langle E + F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}$$

Cooperation without synchronisation:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E' \underset{M}{\bowtie} F, \sigma'\rangle} (\underline{a} \notin M)$$

$$\frac{\langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \sigma'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E \underset{M}{\bowtie} F', \sigma'\rangle} (\underline{a} \notin M)$$

Cooperation with synchronisation:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \tau\rangle \quad \langle F, \sigma\rangle \xrightarrow{\underline{a}} \langle F', \tau'\rangle}{\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{a}} \langle E' \underset{M}{\bowtie} F', \Gamma(\sigma, \tau, \tau')\rangle} (\underline{a} \in M, \Gamma \text{ defined})$$

Constant:
$$\frac{\langle E, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle}{\langle A, \sigma\rangle \xrightarrow{\underline{a}} \langle E', \sigma'\rangle} (A \overset{def}{=} E)$$

Fig. 2. Operational semantics for HYPE

$$\frac{\langle \mathit{Heat}, \sigma\rangle \xrightarrow{\underline{\text{init}}} \langle \mathit{Heat}, \sigma_1\rangle \qquad \langle \mathit{Shade}, \sigma\rangle \xrightarrow{\underline{\text{init}}} \langle \mathit{Shade}, \sigma_3\rangle}{\langle \mathit{Heat} \underset{\underline{\text{init}}}{\bowtie} \mathit{Shade}, \sigma\rangle \xrightarrow{\underline{\text{init}}} \langle \mathit{Heat} \underset{\underline{\text{init}}}{\bowtie} \mathit{Shade}, \sigma_3\rangle}$$

$$\begin{array}{rclcl}
\sigma & = & & & \{h \mapsto *, \quad d \mapsto *, \quad s \mapsto *, c \mapsto *, t \mapsto *\} \\
\sigma_1 & = & \sigma[h \mapsto (0, \mathit{const})] & = & \{h \mapsto (0, \mathit{const}), d \mapsto *, \quad s \mapsto *, c \mapsto *, t \mapsto *\} \\
\sigma_2 & = & \sigma[d \mapsto (0, \mathit{const})] & = & \{h \mapsto *, \quad d \mapsto (0, \mathit{const}), s \mapsto *, c \mapsto *, t \mapsto *\} \\
\sigma_3 & = & \Gamma(\sigma, \sigma_1, \sigma_2) & = & \{h \mapsto (0, \mathit{const}), d \mapsto (0, \mathit{const}), s \mapsto *, c \mapsto *, t \mapsto *\}
\end{array}$$

Fig. 3. Example derivation using Γ ($*$ indicates an undefined value)

form as an activity to reflect the fact that the state captures the activities that are currently in effect. The notion of state here is not a valuation of system variables but rather a collection of flows that occur in the system.

The operational semantics give a labelled transition system over configurations $(\mathcal{F}, \mathcal{E}, \rightarrow \subseteq \mathcal{F} \times \mathcal{E} \times \mathcal{F})$. We write $F \xrightarrow{\underline{a}} F'$ for $(F, \underline{a}, F') \in \rightarrow$. In the following, $E, F \in \mathcal{C}_{Sys}$. The rules are given in Figure 2 and are fairly standard. In Choice, Prefix without influence, Cooperation without synchronisation and Constant, states are not changed by the application of the rule. For Prefix with influence, the state needs to be updated, and for Cooperation with synchronisation, the two new states in the premise of the rule are merged using

$$\begin{array}{lllll}
\sigma_0 = \{h \mapsto (0, \mathit{const}), & d \mapsto (0, \mathit{const}), & s \mapsto (0, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_1 = \{h \mapsto (0, \mathit{const}), & d \mapsto (0, \mathit{const}), & s \mapsto (r_s, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_2 = \{h \mapsto (0, \mathit{const}), & d \mapsto (-r_d, \mathit{const}), & s \mapsto (0, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_3 = \{h \mapsto (0, \mathit{const}), & d \mapsto (-r_d, \mathit{const}), & s \mapsto (r_s, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_4 = \{h \mapsto (r_h, \mathit{const}), & d \mapsto (0, \mathit{const}), & s \mapsto (0, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_5 = \{h \mapsto (r_h, \mathit{const}), & d \mapsto (0, \mathit{const}), & s \mapsto (r_s, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_6 = \{h \mapsto (r_h, \mathit{const}), & d \mapsto (-r_d, \mathit{const}), & s \mapsto (0, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\} \\
\sigma_7 = \{h \mapsto (r_h, \mathit{const}), & d \mapsto (-r_d, \mathit{const}), & s \mapsto (r_s, \mathit{const}), & c \mapsto (-1, \mathit{linear}(K)), & t \mapsto (1, \mathit{const})\}
\end{array}$$

Fig. 4. The states of the orbiter temperature control system

the function Γ. The updating function $\sigma[\iota \mapsto (r, I)]$ is defined as

$$\sigma[\iota \mapsto (r, I)](x) = \begin{cases} (r, I) & \text{if } x = \iota \\ \sigma(x) & \text{otherwise.} \end{cases}$$

The notation $\sigma[u]$ will also be used for an update, with $\sigma[u_1 \dots u_n]$ denoting $\sigma[u_1] \dots [u_n]$. The partial function $\Gamma : \mathcal{S} \times \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ is defined as follows.

$$(\Gamma(\sigma, \tau, \tau'))(\iota) = \begin{cases} \tau(\iota) & \text{if } \sigma(\iota) = \tau'(\iota), \\ \tau'(\iota) & \text{if } \sigma(\iota) = \tau(\iota), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

When synchronisation occurs, two states must be merged and the function uses the previous state and the new states to determine which values have changed and then puts these changed values into the new state. Γ will be undefined if both the second and third argument differ from the first argument, namely if the values in the new state both differ from the old state since this represents conflicting updates. Figure 3 illustrates the use of Γ. We show in Section 5 how we can constrain the syntactic form of our models to ensure that conflicts do not occur in updates while also ensuring well-structured models.

The next two definitions will be useful in referring to the states of the model.

Definition 3.2. The *derivative set* of a controlled system P, $\mathrm{ds}(P)$ is defined as the smallest set satisfying

- if $\langle P, \sigma \rangle \xrightarrow{\underline{\text{init}}} \langle P', \sigma' \rangle$ then $\langle P', \sigma' \rangle \in \mathrm{ds}(P)$
- if $\langle P', \sigma' \rangle \in \mathrm{ds}(P)$ and $\langle P', \sigma' \rangle \xrightarrow{\underline{\text{a}}} \langle P'', \sigma'' \rangle$ then $\langle P'', \sigma'' \rangle \in \mathrm{ds}(P)$.

Definition 3.3. The set of *states* of the derivative set of a controlled system P is defined as $\mathrm{st}(P) = \{\sigma \mid \langle Q, \sigma \rangle \in \mathrm{ds}(P)\}$.

In the *TempCtrl* model, there are eight states of interest (we have omitted the state before the <u>init</u> event) given in Figure 4. Each state captures the influences that are currently active. Since the influence strengths and types of c and t do not change, and each of h, d and s have two possible strengths, there are eight states. For example, σ_3 reflects that the heater (h) is off (and has no effect), the shade (d) is up, the sun (s) is shining, cooling (c) is happening, and time (t) is passing.

4. Hybrid Semantics

We have defined the labelled transition system via the operational semantics and it captures the discrete behaviour of the system in terms of events. We now move on to consider the continuous behaviour of the system. We extract a set of ODEs for each state which appears in a configuration in the labelled transition system, thus describing the continuous behaviour at that configuration. We will label this set as P_σ where P is the constant used for the controlled system and σ is the state.

Given a controlled system P, and a derivative $\langle P', \sigma \rangle \in \mathrm{ds}(P)$, the ODEs associated with the state σ are defined as follows.

$$P'_\sigma = \left\{ \frac{\mathrm{d}V}{\mathrm{d}t} = \sum \left\{ r \times [\![I(\mathcal{W})]\!] \;\middle|\; iv(\iota) = V \text{ and } \sigma(\iota) = (r, I(\mathcal{W})) \right\} \;\middle|\; V \in \mathcal{V} \right\}$$

For each variable, we identify the influences that are associated with that variable, and then, using the information in the state, the ODE is contructed from the product of the strength and type of each influence. The influence type allows the variables on which the variable depend to appear in the ODE. Hence, for each state, we have a collection of ODEs, one for each variable V. We keep influence strength and influence type separate since in many examples, the influence strength varies whereas the influence type does not, allowing a fine-grained description of dynamics. For a discussion of the range of ODEs expressible in HYPE, refer to Section 9.

We obtain the following ODE from σ_3.

$$\mathit{TempCtrl}_{\sigma_3} = \left\{ \frac{\mathrm{d}K}{\mathrm{d}t} = r_s - r_d - K,\ \frac{\mathrm{d}T}{\mathrm{d}t} = 1 \right\}$$

Therefore, this process enables us to obtain ODEs describing how the continuous part of the system evolves, and we have different sets of ODEs to describe the different dynamics that can be in operation. We wish to combine this information with the event conditions already defined and an obvious way to do this is to translate this information into a hybrid automaton. Therefore, this well-supported formalism provides a powerful back-end for HYPE.

Hybrid automata are dynamic systems presenting both discrete and continuous evolution. They consist of variables evolving continuously in time, subject to abrupt changes induced by discrete instantaneous *control* events. When discrete events happen the automaton enters its next *mode*, where the rules governing the flow of continuous variables change. See [Hen96] for further details.

Definition 4.1. A *hybrid automaton* is a tuple $\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$, where

- $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a finite set of real-valued variables. The time derivative of X_j is $\dot{X}_j$, and the value of X_j after a change of *mode* is X'_j.
- The *control graph* $G = (V, E)$ is a finite labelled multigraph. Vertices in the set V are the *(control) modes*, while edges in the multiset E are called *(control) switches* and model the happening of a discrete event.
- Associated with each vertex $v \in V$ there is a set of ordinary differential equations $\dot{\mathbf{X}} = \mathit{flow}(v)$ referred to as the *flow conditions*, where $\mathit{flow}(v)$ is a vector field from $\mathbb{R}^n$ to $\mathbb{R}^n$. Moreover, $\mathit{init}(v)$ and $\mathit{inv}(v)$ are two formulae on $\mathbf{X}$ specifying the *admissible initial conditions* and some *invariant conditions* that must be true during the continuous evolution of variables in v.
- Edges $e \in E$ of the control graph are labelled by an event $\mathit{event}(e) \in \mathcal{E}$ and by $\mathit{jump}(e)$, a formula on $\mathbf{X}$ stating for which values of variables each transition is active, and by $\mathit{reset}(e)$, a formula on $\mathbf{X} \cup \mathbf{X}'$ specifying the change of the variables' values after the transition has taken place. Moreover, each edge $e \in E$ can be declared urgent, by setting to true the boolean flag $\mathit{urgent}(e)$, meaning that the transition is taken at once when $\mathit{jump}(e)$ becomes true. Otherwise, the transition can be taken nondeterministically whenever $\mathit{jump}(e)$ is true. Two edges with the same vertices must differ in at least one of *event*, *jump* or *reset*.

Definition 4.2. Given a hybrid automaton $\mathcal{H}$, *an (automaton) state* is a pair $(v, \mathbf{x})$ for $v \in V$ and $\mathbf{x} = (x_1, \ldots, x_n)$ a tuple of values in $\mathbb{R}^n$.

Consider a HYPE model $(P_0, \mathcal{V}, \mathit{IN}, \mathit{IT}, \mathcal{E}, \mathcal{A}, \mathit{ec}, \mathit{iv}, \mathit{EC}, \mathit{ID})$ with initial configuration $\langle P_0, \sigma_0 \rangle \in \mathcal{F}$. For P_0 the only possible transition is the event $\underline{\text{init}}$. Let $\langle P, \sigma \rangle$ be the configuration reached after its occurrence, $\langle P_0, \sigma_0 \rangle \xrightarrow{\underline{\text{init}}} \langle P, \sigma \rangle$. We introduce some notation for ease of referring to event conditions.

Definition 4.3. Given a HYPE model with $\mathit{ec} : \mathcal{E} \to \mathit{EC}$, let $\mathit{ec}(\underline{a}) = (\mathit{act}(\underline{a}), \mathit{res}(\underline{a}))$ where $\mathit{act}(\underline{a})$ describes the activation condition for event $\underline{a}$ and $\mathit{res}(\underline{a})$ which describes the reset for event $\underline{a}$.

Definition 4.4. The hybrid automaton $\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ can be obtained from the HYPE model $(P_0, \mathcal{V}, \mathit{IN}, \mathit{IT}, \mathcal{E}, \mathcal{A}, \mathit{ec}, \mathit{iv}, \mathit{EC}, \mathit{ID})$ as follows.

- The set of modes V is the set of configurations reachable in 0 or more steps from $\langle P, \sigma \rangle$, namely $\mathrm{ds}(P_0)$.

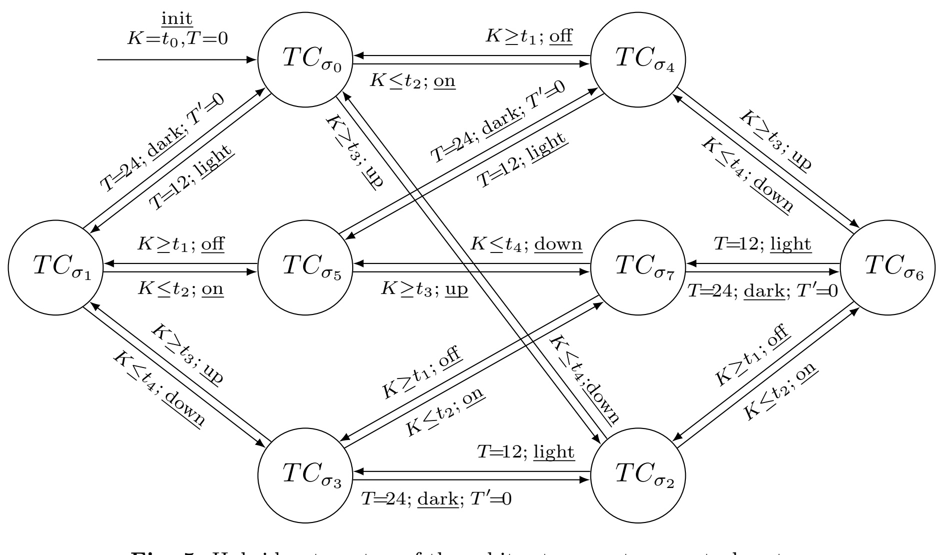


Fig. 5. Hybrid automaton of the orbiter temperature control system

- The edges E of the control graph connect two modes (v_1, v_2) iff $v_1 = \langle P_1, \sigma_1\rangle$, $v_2 = \langle P_2, \sigma_2\rangle$ and $\langle P_1, \sigma_1\rangle \xrightarrow{\underline{a}} \langle P_2, \sigma_2\rangle$ is a derivation for some $\underline{a}$.
- $\mathbf{X} = \mathcal{V}$ is the set of variables of the HYPE system.
- $\mathcal{E}$ is the set of events $\mathcal{E}$ of P_0.
- Let $v_j = \langle P_j, \sigma_j\rangle$, then $\mathit{flow}(v_j)[X_i] = \sum\{r[\![I(\mathcal{W})]\!] \mid iv(\iota) = X_i \text{ and } \sigma_j(\iota) = (r, I(\mathcal{W}))\}$.
- $\mathit{init}(v) = \begin{cases} \mathit{res}(\underline{\text{init}}), & \text{if } v = \langle P, \sigma\rangle \\ \mathit{false}, & \mathit{otherwise} \end{cases}$ with primes removed from variables[5].
- $\mathit{inv}(v) = \mathit{true}$.
- Let $e = (\langle P_1, \sigma_1\rangle, \langle P_2, \sigma_2\rangle)$ with $\langle P_1, \sigma_1\rangle \xrightarrow{\underline{a}} \langle P_2, \sigma_2\rangle$. Then $\mathit{event}(e) = \underline{a}$ and $\mathit{reset}(e) = \mathit{res}(\underline{a})$. Moreover, if $\mathit{act}(\underline{a}) \neq \perp$, then $\mathit{jump}(e) = \mathit{act}(\underline{a})$ and $\mathit{urgent}(e) = \mathit{true}$, otherwise $\mathit{jump}(e) = \mathit{true}$ and $\mathit{urgent}(e) = \mathit{false}$.

Given a HYPE model P, we denote its hybrid automaton as $\mathcal{H}(P)$. As long as there is a finite number of subcomponents (since infinite sum is not possible) and a finite number of controller components, then the number of modes is finite[6]. Figure 5 shows the HYPE model from the example as a hybrid automaton. Note that in this case, which modes are visited is determined by the values of k_i, $i = 0, \ldots, 4$.

We do not use invariants in HYPE models because they are specific to modes of the hybrid automata, which are obtained in HYPE as the possible configurations of the controlled system. This implies that invariants should be specified within the controlled system, not orthogonally as for event conditions. This conflicts with the modelling philosophy of HYPE. However, the behaviour of invariants can be obtained in our setting by incorporating them in the activation condition of events, that is by forcing events to happen as soon as the conditions that would have been specified by an invariant become false.

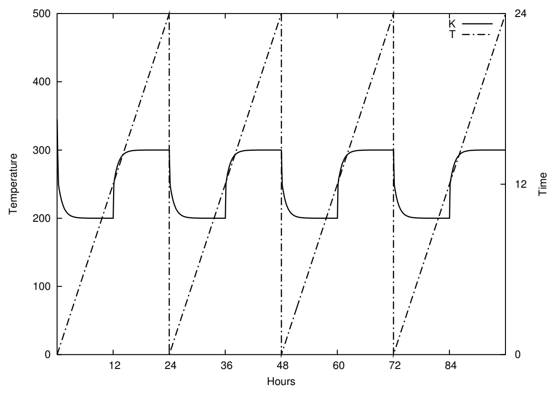
The behaviour of a hybrid automaton can be defined in terms of a transition system [Hen96].

Definition 4.5. The *timed transition system* of a hybrid automaton has a set of states $\mathcal{S} \subseteq V \times \mathbb{R}^n$ with initial states $\mathcal{S}_0$ and transitions labelled with elements from $\mathcal{E} \cup \mathbb{R}$ such that

1. $(v, \mathbf{x}) \in \mathcal{S}$ whenever $\mathit{inv}(v)(\mathbf{x})$ holds and $(v, \mathbf{x}) \in \mathcal{S}_0$ whenever $\mathit{init}(v)(\mathbf{x})$ holds.
2. $(v, \mathbf{x}) \xrightarrow{a} (v', \mathbf{x}')$ whenever there is an edge $e \in E$ such that $e = (v, v')$, $\mathit{event}(e) = a$, $\mathit{jump}(e)(\mathbf{x})$ and $\mathit{reset}(e)(\mathbf{x}, \mathbf{x}')$ both hold.

[5] $\mathit{res}(\underline{\text{init}})$ is a reset so uses primed variables to refer to the new values of variables whereas $\mathit{init}(v)$ is an initialisation condition and refers to variables without primes.

[6] To ensure finiteness of controller components, recurrent controller components with increasing indices should be avoided, for example $\mathit{Con}_i \stackrel{\mathit{def}}{=} \underline{a}.\mathit{Con}_{i+1}$. This type of unbounded controller is unlikely to be required when modelling hybrid systems, and hence a maximum controller can be specified, for example $\mathit{Con}_n \stackrel{\mathit{def}}{=} \underline{a}.0$.



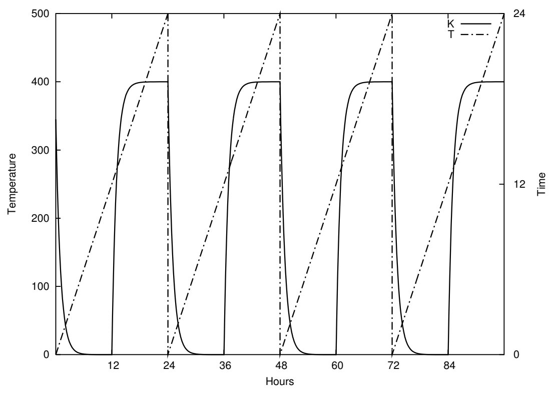


Fig. 6. Traces for the orbiter with shade and heating (left) and without shade and heating (right) where $r_h = 200$, $r_d = 100$, $r_s = 400$, $k_0 = 350$, $k_1 = 250$, $k_2 = 250$, $k_3 = 300$ and $k_4 = 300$.

3. $(v, \mathbf{x}) \xrightarrow{\tau} (v', \mathbf{x}')$ whenever $v = v'$ and there is a differentiable function $f : [0, \tau] \to \mathbb{R}^n$ with first derivative $f' : (0, \tau) \to \mathbb{R}^n$ such that $f(0) = \mathbf{x}$ and $f(\tau) = \mathbf{x}'$ and for all $t \in (0, \tau)$, $inv(v)(\mathbf{x})$ holds and $flow(v)(\mathbf{x}) = f'(\mathbf{x})$. Moreover, for all $t \in (0, \tau)$ and all $e = (v, u) \in E$ with $urgent(e) = true$, $jump(e)(f(t)) = false$.

The third condition of this definition ensures that a transition associated with the passing of time can only proceed if no jump condition becomes true for an urgent edge. Therefore continuous behaviour can only continue if a mode change is not required. Note that for our mapping of HYPE models to hybrid automata, this definition is somewhat simplified by the lack of invariants.

Definition 4.6. Given a timed transition of a hybrid automaton, a *trace* of the hybrid automaton is a sequence of the form $(v_0, \mathbf{x}_0) \xrightarrow{\rho_0} (v_1, \mathbf{x}_1) \xrightarrow{\rho_1} \ldots$ where $(v_i, \mathbf{x}_i) \in \mathcal{S}$, $(v_0, \mathbf{x}_0) \in \mathcal{S}_0$ and $\rho_i \in \mathcal{E} \cup \mathbb{R}$.

With the appropriate software, we can construct simulations of our HYPE models, and demonstrate their traces graphically. Note that HYPE models are not necessarily deterministic. The presence of non-urgent transitions can introduce non-determinism, as can choice between events when activation conditions become true in the same time instant. In the case of the orbiter, we do have determinism. A trace is demonstrated in the left graph of Figure 6 for the parameters $r_h = 200$, $r_d = 100$, $r_s = 400$, $k_1 = 250$, $k_2 = 250$, $k_3 = 300$, $k_4 = 300$ and $k_0 = 350$. If we remove the temperature controls in the orbiter, we obtain the right hand graph which demonstrates the temperature extremes which can be achieved. This graph is based on the HYPE model

$$((Sun \underset{\{\underline{\mathrm{init}}\}}{\bowtie} Cool(K)) \underset{\{\underline{\mathrm{init}},\underline{\mathrm{light}},\underline{\mathrm{dark}}\}}{\bowtie} Time) \underset{\{\underline{\mathrm{init}},\underline{\mathrm{light}},\underline{\mathrm{dark}}\}}{\bowtie} \underline{\mathrm{init}}.Con_s$$

This section has defined the semantics of HYPE models, both discrete and continuous. In the example, the subcomponents of the orbiter have a very specific form which is not required by the HYPE definition which is more liberal. In the next two section, we consider how constraining some aspects of models can result in models with desirable behaviour.

5. Well-defined HYPE models

Now that we have defined the behaviour of HYPE models, we wish to specify a subset of HYPE models that have "good" structure and behaviour. We introduce the notion of a well-defined HYPE model and reason about how this notion affects derivation of transitions specifically where the cooperation operator is used. First we require some auxiliary definitions.

Definition 5.1. Given a controlled system P, the *set of events*, $\mathsf{ev}(P)$ is defined structurally as

$$\mathsf{ev}(\underline{a} : \alpha.S) = \{\underline{a}\} \quad \mathsf{ev}(\underline{a}.S) = \{\underline{a}\} \quad \mathsf{ev}(S_1 + S_2) = \mathsf{ev}(S_1) \cup \mathsf{ev}(S_2) \quad \mathsf{ev}(P_1 \underset{L}{\bowtie} P_2) = \mathsf{ev}(P_1) \cup \mathsf{ev}(P_2)$$

Definition 5.2. Given a controlled system P, the *set of influences*, $\mathsf{in}(P)$ is defined structurally as

$$\mathsf{in}(\underline{a} : (\iota, r, I(\mathcal{W})).S) = \{\iota\} \quad \mathsf{in}(\underline{a}.S) = \emptyset \quad \mathsf{in}(S_1 + S_2) = \mathsf{in}(S_1) \cup \mathsf{in}(S_2) \quad \mathsf{in}(P_1 \underset{L}{\bowtie} P_2) = \mathsf{in}(P_1) \cup \mathsf{in}(P_2)$$

Definition 5.3. Given a controlled system P, the *set of prefixes*, $\mathsf{pr}(P)$ is defined structurally as

$$\mathsf{pr}(\underline{\mathrm{a}}:\alpha.S) = \{\underline{\mathrm{a}}:\alpha\} \quad \mathsf{pr}(\underline{\mathrm{a}}.S) = \emptyset \quad \mathsf{pr}(S_1 + S_2) = \mathsf{pr}(S_1) \cup \mathsf{pr}(S_2) \quad \mathsf{pr}(P_1 \underset{L}{\bowtie} P_2) = \mathsf{pr}(P_1) \cup \mathsf{pr}(P_2)$$

Definition 5.4.
A *well-defined* controlled system has the following properties.

1. Subcomponents have the form $S(\mathcal{W}) \overset{\mathit{def}}{=} \sum_{j=1}^{n} \underline{\mathrm{a}}_j{:}(\iota, r_j, I_j(\mathcal{W})).S(\mathcal{W}) + \underline{\mathrm{init}}{:}(\iota, r, I(\mathcal{W})).S(\mathcal{W})$ where $n \geq 0$, $\underline{\mathrm{a}}_j \neq \underline{\mathrm{a}}_k$ for $j \neq k$ and $\underline{\mathrm{a}}_j \neq \underline{\mathrm{init}}$ for all j.
2. For each pair of subcomponents S_i and S_j, $\mathsf{in}(S_i) \cap \mathsf{in}(S_j) = \emptyset$ for $i \neq j$.
3. For a component $C_i \underset{L}{\bowtie} C_j$, $\mathsf{ev}(C_i) \cap \mathsf{ev}(C_j) = L$.
4. For the uncontrolled system Σ, $\mathsf{ev}(\Sigma) = \mathcal{E}$ and $\mathsf{in}(\Sigma) = \mathit{IN}$.
5. For the controlled system $\Sigma \underset{L}{\bowtie} \underline{\mathrm{init}}.\mathit{Con}$, $\mathsf{ev}(\Sigma) \cap \mathsf{ev}(\underline{\mathrm{init}}.\mathit{Con}) = L$ and $\mathsf{ev}(\mathit{Con}) \subseteq \mathsf{ev}(\Sigma)$.

This definition ensures that HYPE models are built up out of individual flows in a consistent manner. Each subcomponent captures the events that affect a particular influence and this influence is linked to a variable by the function iv. The first two conditions ensure a one-to-one relationship between subcomponents and influence names. The third and fifth conditions ensure that when an event happens, all subcomponents and controllers react to it. We use the symbol $\underset{*}{\bowtie}$ in HYPE models to show when these two conditions hold in a model. The fourth condition ensures there are no unused events or influence names, and is useful to know when defining the synchronisation of two HYPE models. The orbiter is a well-defined HYPE model.

The fifth condition permits events to appear in the uncontrolled system that do not appear in the controller, since not all events need to depend causally on other events, and we do not wish to compel modellers to define a controller for each of these events. However, for simplicity of definitions, in the remainder of this document, we will assume that there is a controller for each event $\underline{\mathrm{u}}$ that does not appear in any defined controller, with the form $\mathit{Con}_u \overset{\mathit{def}}{=} \underline{\mathrm{u}}.\mathit{Con}_u$.

Not only does well-definedness ensure our models have a specific form that exemplifies how we build them, but as we will show in Section 8 this form is useful when we consider equivalence semantics and reason about the behaviour of our models. Note that we also impose, through the definition of a HYPE model, the conditions that resets and initial values are deterministic, namely a single value is defined. Additionally, activation conditions define closed sets, as guaranteed by the use of nonstrict inequalities and of positive boolean formulae to combine them.

We now present a result that follows from well-definedness. The function Γ used in the cooperation rules, is always defined. This ensures that it is always possible to derive a transition for two components in cooperation.

Proposition 5.1. Let E and F be components of a well-defined HYPE model such that $\langle E, \sigma\rangle \xrightarrow{\underline{\mathrm{a}}} \langle E', \tau\rangle$ and $\langle F, \sigma\rangle \xrightarrow{\underline{\mathrm{a}}} \langle F', \tau'\rangle$ then the transition $\langle E \underset{M}{\bowtie} F, \sigma\rangle \xrightarrow{\underline{\mathrm{a}}} \langle E' \underset{M}{\bowtie} F', \Gamma(\sigma, \tau, \tau')\rangle$ is always possible for $\underline{\mathrm{a}} \in M$.

Proof. The first two conditions of well-definedness ensure that each subcomponent is associated with exactly one influence name and *vice versa*. Therefore $\mathsf{in}(E) \cap \mathsf{in}(F) = \emptyset$. Let ι be an influence name appearing in E. Since it does not appear in F, $\sigma(\iota) = \tau'(\iota)$ since it cannot have been modified in the derivation of F, hence $(\Gamma(\sigma, \tau, \tau'))(\iota)$ is defined. A similar argument can be applied to influence names appearing in F and not E. Hence $\Gamma(\sigma, \tau, \tau')$ is defined. $\square$

Additionally, we provide a result about the form that the configurations of a HYPE model take.

Proposition 5.2. If P is a well-defined HYPE model with controlled system $P \overset{\mathit{def}}{=} \Sigma \underset{*}{\bowtie} \underline{\mathrm{init}}.\mathit{Con}$ then $ds(P) = \{\langle \Sigma \underset{*}{\bowtie} \mathit{Con}, \tau\rangle, \langle \Sigma \underset{*}{\bowtie} D_1, \sigma_1\rangle, \ldots, \langle \Sigma \underset{*}{\bowtie} D_n, \sigma_n\rangle\}$ where, for arbitrary state σ, the derivative set of the controller is $ds(\underline{\mathrm{init}}.\mathit{Con}) = \{\langle \mathit{Con}, \sigma\rangle, \langle D_1, \sigma\rangle, \ldots, \langle D_n, \sigma\rangle\}$. Moreover, each $\Sigma \underset{*}{\bowtie} D_i$ is well-defined.

Proof. First note that by the particular form of Σ imposed by well-definedness, for any event $\underline{\mathrm{a}}$ that appears in Σ, $\langle \Sigma, \rho\rangle \xrightarrow{\underline{\mathrm{a}}} \langle \Sigma, \rho'\rangle$ for some states ρ, ρ'. This means that the configurations of $\Sigma \underset{*}{\bowtie} \mathit{Con}$ are totally determined by the events that are allowed by the controller. Moreover, the value of σ in $ds(\mathit{Con})$ does not determine these events. Additionally, $\underset{*}{\bowtie}$ is a static operator. For well-definedness, note that Conditions 1-4 in Definition 5 hold for Σ since $\Sigma \underset{*}{\bowtie} \underline{\mathrm{init}}.\mathit{Con}$ is well-defined, and Condition 5 is only required to hold of the system before the event $\underline{\mathrm{init}}$ has occurred. $\square$

We can also reason about derivatives and the relationship between states. In well-defined HYPE models, subcomponents are defined as a number of simple loops, reflecting events that can occur and the associated changes in the continuous part of the system. For such systems, we can show that the current state in a configuration does not determine which future events happen (only the controller can influence this) and hence the state can be discounted in certain settings.

The next proposition shows how, if we know how one state is related to a prior state, then we can conclude that a similar relationship holds if we use a different state. Additionally, we can express a state modification in terms of the a minimum number of updates, one for each influence that changes.

Proposition 5.3. In a well-defined controlled system, if $\langle P', \sigma[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]\rangle$ is a derivative of $\langle P, \sigma\rangle$, then $\langle P', \tau[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]\rangle$ is a derivative of $\langle P, \tau\rangle$. Moreover, it is the case that $\sigma[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)] = \sigma[\iota'_1 \mapsto (r'_1, I'_1)] \ldots [\iota'_m \mapsto (r'_m, I'_m)]$ for $m \leq n$ where each influence name ι'_k appears at most once.

Proof. We first consider the single step case and prove this by induction on the the height of the derivation of the transition. Symmetrical cases have been omitted.

Prefix with influence $P \equiv \underline{a} : (\iota, r, I).P_1$ so $\langle \underline{a} : (\iota, r, I).P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1, \sigma[\iota \mapsto (r, I)]\rangle$. Similarly, we also have $\langle \underline{a} : (\iota, r, I).P_1, \tau\rangle \xrightarrow{\underline{a}} \langle P_1, \tau[\iota \mapsto (r, I)]\rangle$.

Prefix without influence $P \equiv \underline{a}.P_1$ so $\langle \underline{a}.P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1, \sigma\rangle$. Similarly, $\langle \underline{a}.P_1, \tau\rangle \xrightarrow{\underline{a}} \langle P_1, \tau\rangle$.

Choice $P \equiv P_1 + P_2$ so $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]\rangle$ by a shorter inference. By the inductive hypothesis, it is the case that $\langle P_1, \tau\rangle \xrightarrow{\underline{a}} \langle P', \tau[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]\rangle$, and therefore $\langle P_1 + P_2, \tau\rangle \xrightarrow{\underline{a}} \langle P', \tau[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]\rangle$.

Cooperation without synchronisation $P \equiv P_1 \underset{L}{\bowtie} P_2$ with $\underline{a} \notin L$ so we know that $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]\rangle$ by a shorter inference. Straightforward.

Cooperation with synchronisation $P \equiv P_1 \underset{L}{\bowtie} P_2$ with $\underline{a} \in L$. We have the transitions $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P'_1, \sigma'_1\rangle$ and $\langle P_2, \sigma\rangle \xrightarrow{\underline{a}} \langle P'_2, \sigma'_2\rangle$ by a shorter inference where $\sigma'_1 = \sigma[\eta_1 \mapsto (s_1, J_1)] \ldots [\eta_m \mapsto (s_m, J_m)]$ and $\sigma'_2 = \sigma[\xi_1 \mapsto (t_1, K_1)] \ldots [\xi_p \mapsto (t_p, K_p)]$ such that $\Gamma(\sigma, \sigma'_1, \sigma'_2) = \sigma[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]$ with $P' \equiv \langle P'_1 \underset{L}{\bowtie} P'_2, \Gamma(\sigma, \sigma'_1, \sigma'_2)\rangle$. So by the inductive hypothesis $\langle P_1, \tau\rangle \xrightarrow{\underline{a}} \langle P'_1, \tau'_1\rangle$ and $\langle P_2, \tau\rangle \xrightarrow{\underline{a}} \langle P'_2, \tau'_2\rangle$ where $\tau'_1 = \tau[\eta_1 \mapsto (s_1, J_1)] \ldots [\eta_m \mapsto (s_m, J_m)]$ and $\tau'_2 = \tau[\xi_1 \mapsto (t_1, K_1)] \ldots [\xi_p \mapsto (t_p, K_p)]$ and $\langle P, \tau\rangle \xrightarrow{\underline{a}} \langle P'_1 \underset{L}{\bowtie} P'_2, \Gamma(\tau, \tau'_1, \tau'_2)\rangle$. Note that $\Gamma(\tau, \tau'_1, \tau'_2) = \tau[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]$ since the same influences have been updated to the same values, and Γ is defined piecewise over influences.

Constant Straightforward.

We can use an identical proof to go from an $(n-1)$-step derivation to an n-step derivation. Hence by induction, over the length of the derivation, this result holds for all derviations.

Finally, because of the lack of dependence on previous states in the labelled transition system, it is possible to reduce the number of updates by only retaining the last update for a given influence name ι. $\square$

The next proposition shows that the state is irrelevant when considering the actions that can be performed by a system.

Proposition 5.4. The labelled transition systems of $\langle P, \sigma\rangle$ and $\langle P, \tau\rangle$ are isomorphic if P is well-defined.

Proof. This can be proved using Proposition 5.3. $\square$

We can also consider what happens to a system after the first $\underline{\text{init}}$ event. Note that the definition of a well-defined controlled system requires that the first transition be an $\underline{\text{init}}$ event since all events must appear in the cooperation set and *Con* is prefixed by $\underline{\text{init}}$. The following proof shows that the starting state is irrelevant since the $\underline{\text{init}}$ action will set every value in the state.

Proposition 5.5. Let P be a well-defined controlled system. If $\langle P, \sigma\rangle \xrightarrow{\underline{\text{init}}} \langle P', \sigma'\rangle$ and $\langle P, \tau\rangle \xrightarrow{\underline{\text{init}}} \langle P', \tau'\rangle$ then $\sigma' = \tau'$.

Proof. By definition of a well-defined controlled system, there is a prefix containing $\underline{\text{init}}$ and ι for every ι in the controlled system. Assuming $\mathcal{A} = \{\iota_1, \ldots \iota_n\}$ which appear in the controlled system and no others, then a state σ maps each ι to a rate and influence type pair. After an $\underline{\text{init}}$ each ι will be updated and the state before the transition is irrelevant, namely $\sigma' = \sigma[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)] = \emptyset[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)] = \tau[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)] = \tau'$ by Proposition 5.3. $\square$

6. Well-behaved HYPE models

We have considered the syntactic form that HYPE models can take, and the advantages of this at the level of the labelled transition system. We now consider a notion of behaviour that ensures our models show reasonable behaviour, and motivate it by considering the underlying hybrid automaton of the model.

Definition 6.1. A HYPE model P is well-behaved if it has a finite number of finite sequences of simultaneous instantaneous events and these sequences are independent of the initial state of the system.

This ensures that our models cannot perform an infinite number of discrete steps in a single time instant. We call this unwanted behaviour *instantaneous Zeno behaviour* by analogy with Zeno behaviour which describes the possibility of an infinite number of steps in a finite time period. Our focus on this unwanted instantaneous behaviour is informed by our later work [BGH10a] where we consider Piecewise Deterministic Markov Processes [Dav93] as a semantic model which excludes this behaviour.

We now characterise well-behaved HYPE models in terms of their event conditions and controllers by constructing an instantaneous activation graph or I-graph for a model, and checking it for acyclicity.

The I-graph captures information about the flow of data in the HYPE model. It simulates the behaviour of the instantaneous transitions so that each sequence of simultaneous instantaneous events corresponds to a path in the graph. This overapproximates behaviour since it is based on constraints imposed by the event conditions and the controller only, and does not take account of changes that occur due to initial conditions or continuous flow. This means additional information that could further constrain the possible values is ignored. This is an appropriate approach to take because we want to reason about the HYPE model without investigating its full behaviour.

For each event, we consider the intersection between its activation region (the set of points making the activation condition true for the event) and its reset region (the set of points reachable immediately after the occurrence of the event). Such set-based constraints thus take into account overlap only at the set level and not at the single value level. This means that the I-graph does not capture the changes in the value of a variable, only the regions that these values can fall into. When necessary, we also overapproximate activation and reset regions with computable sets to ensure that testing for acyclicity of an I-graph is decidable.

We will show that if no cycles are found in the I-graph, then instantaneous Zeno behaviour is not possible in the HYPE model (which, in this context, means in the hybrid automaton), but we cannot know the converse – a cycle in the graph does not guarantee that there is instantaneous Zeno behaviour in the model since these are overapproximations.

We start with some preliminary definitions.

Definition 6.2. Given an event $\underline{\mathrm{a}}$, $\mathcal{V}(\underline{\mathrm{a}})$ is the set of variables that appear explicitly in $ec(\underline{\mathrm{a}}) = (act(\underline{\mathrm{a}}), res(\underline{\mathrm{a}}))$.

Hence $\mathcal{V}(\underline{\mathrm{a}})$ contains all of the variables in $\mathcal{V}$ except those that have implicit identity resets for $\underline{\mathrm{a}}$. From this, we can define when two events have disjoint (explicit) variables.

Definition 6.3. Events $\underline{\mathrm{a}}$ and $\underline{\mathrm{b}}$ are *independent* if $\mathcal{V}(\underline{\mathrm{a}}, \underline{\mathrm{b}}) = \mathcal{V}(\underline{\mathrm{a}}) \cap \mathcal{V}(\underline{\mathrm{b}}) = \emptyset$.

We wish to investigate the overlap of values between resets of one event and activation conditions of another, and also take into account the scheduling of events by the controller. Let $r_{\underline{\mathrm{a}}}$ be the reset $res(\underline{\mathrm{a}})$, expressed as a function.

Definition 6.4. For any event $\underline{\mathrm{a}}$, let $G_{\underline{\mathrm{a}}} = \{\mathbf{x} \in \mathbb{R}^n \mid act(\underline{\mathrm{a}}) \text{ is true for } \mathbf{x}\}$ and $R_{\underline{\mathrm{a}}} = r_{\underline{\mathrm{a}}}(G_{\underline{\mathrm{a}}})$

$G_{\underline{\mathrm{a}}}$ describes all points at which $act(\underline{\mathrm{a}})$ is true and hence the event $\underline{\mathrm{a}}$ is active, and $R_{\underline{\mathrm{a}}}$ takes an image of this set, so we can consider $R_{\underline{\mathrm{a}}}$ as all possible values that the system can take after the occurrence of $\underline{\mathrm{a}}$. For $V \notin \mathcal{V}(\underline{\mathrm{a}})$, the value associated with V will be $\mathbb{R}$. We can then determine enablers and inhibitors of events. In the following definition, $\pi_{\{V_{i_1},\dots,V_{i_p}\}}(A) \subseteq \mathbb{R}^p$ for $A \subseteq \mathbb{R}^n$ is the projection of A to the coordinates associated with variables $V_{i_1}, \dots, V_{i_p}$.

Definition 6.5. Consider two non-independent events $\underline{\mathrm{a}}$ and $\underline{\mathrm{b}}$ of an HYPE model. Then

- $\underline{\mathrm{a}}$ is a *(potential) enabler* of $\underline{\mathrm{b}}$ if and only if $\pi_{\mathcal{V}(\underline{\mathrm{a}},\underline{\mathrm{b}})}(R_{\underline{\mathrm{a}}}) \cap \pi_{\mathcal{V}(\underline{\mathrm{a}},\underline{\mathrm{b}})}(G_{\underline{\mathrm{b}}}) \neq \emptyset$;
- $\underline{\mathrm{a}}$ is an *inhibitor* of $\underline{\mathrm{b}}$ if and only if $\pi_{\mathcal{V}(\underline{\mathrm{a}},\underline{\mathrm{b}})}(R_{\underline{\mathrm{a}}}) \cap \pi_{\mathcal{V}(\underline{\mathrm{a}},\underline{\mathrm{b}})}(G_{\underline{\mathrm{b}}}) = \emptyset$.

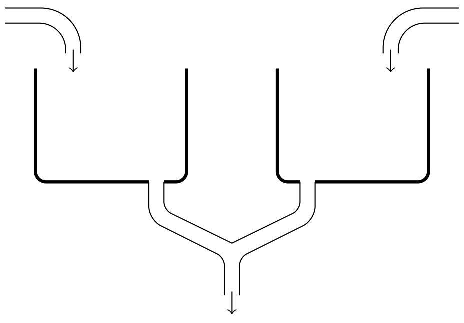


Fig. 7. Two tanks with separate inputs and joint output

This definition considers how the resets of one event can affect whether another event can happen immediately after it. We can use these ideas as the basis for constructing a graph that captures the possible sequences of instantaneous events.

As mentioned in the previous section, we assume a controller for every uncontrolled event $\underline{\mathrm{u}}$ of the form $Con_u \overset{def}{=} \underline{\mathrm{u}}.Con_u$. It is not sufficient only to require that uncontrolled events inhibit themselves (an obvious condition to avoid infinite instantaneous sequences of the same event) since their event conditions may interact with the event conditions of other events, and hence we must consider these controllers from the beginning of the construction of the I-graph.

First, we let $\mathrm{ds}(\underline{\mathrm{init}}.Con) = \{C_1, \dots, C_m\}$ be the controller states (by applying the definition of ds in the obvious manner to the controller, and then ignoring the state which is identical for each controller derivative). The state $\underline{\mathrm{init}}.Con$ is not in this set. Let the instantaneous events appearing in the controller be $\{\underline{\mathrm{a}}_1, \dots, \underline{\mathrm{a}}_k\}$ and $\mathsf{ev}(C_i)$ is the subset of events that can be executed in mode C_i of the controller. Note that we are not working compositionally here. For a vector κ of k values, we write $\kappa[i \leftarrow u]$ to represent the ith position being updated with the value u. This vector will record which events are enabled and which are inhibited.

Definition 6.6. Given a HYPE model P with controller Con, the *instantaneous activation graph*, or *I-graph*, is given by $G_P(Con) = (V_P(Con), E_P(Con))$ where vertices are defined by

$$V_P(Con) = \bigcup_{C_i \in \mathrm{ds}(\underline{\mathrm{init}}.Con)} \Big(\{C_i\} \times \mathsf{ev}(C_i) \times \{0,1\}^k\Big) \subseteq \{C_1, \dots, C_k\} \times \{\underline{\mathrm{a}}_1, \dots, \underline{\mathrm{a}}_k\} \times \{0,1\}^k.$$

The edge $(C_i, \underline{\mathrm{a}}_i, \kappa_i) \to (C_j, \underline{\mathrm{a}}_j, \kappa_j)$ belongs to $E_P(Con)$ if and only if

1. $\langle C_i, \sigma\rangle \xrightarrow{\underline{\mathrm{a}}_i} \langle C_j, \sigma\rangle$ is a transition in the labelled transition system of the controller,
2. $\kappa_i[i] = 1$,
3. $\kappa_j[j] = 1$,
4. $\kappa_j = \kappa_i[k_1 \leftarrow 0] \dots [k_s \leftarrow 0][l_1 \leftarrow 1] \dots [l_r \leftarrow 1]$ where $\underline{\mathrm{a}}_{k_1}, \dots, \underline{\mathrm{a}}_{k_s}$ are the events for which $\underline{\mathrm{a}}_i$ is an inhibitor and $\underline{\mathrm{a}}_{l_1}, \dots, \underline{\mathrm{a}}_{l_r}$ are the events for which $\underline{\mathrm{a}}_i$ is an enabler.

The boolean vector κ records if an event is enabled or not. Hence, the second condition ensures that $\underline{\mathrm{a}}_i$ can fire and the third condition ensures that an edge is only added when it is possible for the next action to fire – this reduces the number of edges in the graph. The fourth condition encodes the updating of what is enabled. A node $(C, \underline{\mathrm{a}}, \kappa)$ describes being in the state C with $\underline{\mathrm{a}}$ as the next action together with a vector describing which events are possible before the event $\underline{\mathrm{a}}$ has occurred. The I-graph represents information about the structure of the controller and the event conditions of a HYPE model.

We now introduce an example to illustrate how this construction works. Figure 7 shows two tanks that each have an input, and which are drained to a common pipe with a switch that allows liquid to be drained from either tank A or tank B. Once A is empty, the tap is switched for B to drain and *vice versa*. Each tank stops filling once it is full with capacity c_A or c_B, and a very small amount of liquid is lost reducing the level by a small amount, say l which is much smaller than either capacity. The initial state of the model is when tank A is draining and B is filling and the initial values are between empty and full. The HYPE model is now defined. Here we use $\|$ for $\underset{\emptyset}{\bowtie}$.

$$\begin{array}{rcl}
\mathit{In}_A & \overset{\mathit{def}}{=} & \underline{\mathrm{full}}_A{:}(\mathit{in}_A, 0, c).\mathit{In}_A + \underline{\mathrm{empty}}_B{:}(\mathit{in}_A, r_{in}, c).\mathit{In}_A + \underline{\mathrm{init}}{:}(\mathit{in}_A, r_{in}, c).\mathit{In}_A \\
\mathit{Out}_A & \overset{\mathit{def}}{=} & \underline{\mathrm{empty}}_A{:}(\mathit{out}_A, 0, c).\mathit{Out}_A + \underline{\mathrm{empty}}_B{:}(\mathit{out}_A, -r_{out}, c).\mathit{Out}_A + \underline{\mathrm{init}}{:}(\mathit{out}_A, -r_{out}, c).\mathit{Out}_A \\
 & & \\
\mathit{In}_B & \overset{\mathit{def}}{=} & \underline{\mathrm{full}}_B{:}(\mathit{in}_B, 0, c).\mathit{In}_B + \underline{\mathrm{empty}}_A{:}(\mathit{in}_B, s_{in}, c).\mathit{In}_B + \underline{\mathrm{init}}{:}(\mathit{in}_B, s_{in}, c).\mathit{In}_B \\
\mathit{Out}_B & \overset{\mathit{def}}{=} & \underline{\mathrm{empty}}_B{:}(\mathit{out}_B, 0, c).\mathit{Out}_B + \underline{\mathrm{empty}}_A{:}(\mathit{out}_B, -s_{out}, c).\mathit{Out}_B + \underline{\mathrm{init}}{:}(\mathit{out}_B, 0, c).\mathit{Out}_B
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
C_A & \overset{\mathit{def}}{=} & \underline{\mathrm{full}}_A.C_A & C_B & \overset{\mathit{def}}{=} & \underline{\mathrm{full}}_B.C_B \\
C_{dA} & \overset{\mathit{def}}{=} & \underline{\mathrm{empty}}_A.C_{dB} & C_{dB} & \overset{\mathit{def}}{=} & \underline{\mathrm{empty}}_B.C_{dA} \\
 & & & & & \\
\mathit{System} & \overset{\mathit{def}}{=} & ((\mathit{In}_A \underset{N}{\bowtie} \mathit{Out}_A) \underset{N \cup M}{\bowtie} (\mathit{In}_B \underset{M}{\bowtie} \mathit{Out}_B)) & N & = & \{\underline{\mathrm{empty}}_B, \underline{\mathrm{init}}\} \\
 & & \underset{L}{\bowtie} \underline{\mathrm{init}}.(C_A \parallel C_B \parallel C_{dA}) & M & = & \{\underline{\mathrm{empty}}_A, \underline{\mathrm{init}}\} \\
 & & & L & = & \{\underline{\mathrm{full}}_A, \underline{\mathrm{full}}_B\} \cup N \cup M \\
 & & & & & \\
\mathit{iv}(\iota_A) & = & A & \mathit{iv}(\iota_B) & = & B \\
 & & & & & \\
\mathit{ec}(\underline{\mathrm{empty}}_A) & = & (A = 0, \mathit{true}) & \mathit{ec}(\underline{\mathrm{full}}_A) & = & (A = c_A, A' = c_A - l) \\
\mathit{ec}(\underline{\mathrm{empty}}_B) & = & (B = 0, \mathit{true}) & \mathit{ec}(\underline{\mathrm{full}}_B) & = & (B = c_B, B' = c_B - l) \\
\mathit{ec}(\underline{\mathrm{init}}) & = & (\mathit{true}, A' = A_0 \wedge B' = B_0) & [\![c]\!] & = & 1
\end{array}$$

Fig. 8. HYPE model of the two tanks

$$\begin{tikzcd}
C1 \arrow[loop left, "\underline{\mathrm{full}}_A{,}\,\underline{\mathrm{full}}_B"] \arrow[rr, bend left, "\underline{\mathrm{empty}}_A"] & & C2 \arrow[loop right, "\underline{\mathrm{full}}_A{,}\,\underline{\mathrm{full}}_B"] \arrow[ll, bend left, "\underline{\mathrm{empty}}_B"]
\end{tikzcd}$$

Fig. 9. Labelled transition system of the controller for the two tanks model

The HYPE model is given in Figure 8. The values of r_{in}, r_{out}, s_{in}, s_{out} determine the behaviour of the system. Since the system starts with tank A draining, if $r_{in} \geq r_{out}$ then liquid will only be taken from tank A. When $r_{in} > r_{out}$ then the tank will fill repeatedly. If $r_{in} < r_{out}$ and $s_{in} \geq s_{out}$ then there will be a single switch to tank B after which all liquid will be drained by tank B. This means that the most interesting behaviour occurs in the situation when $r_{in} < r_{out}$ and $s_{in} < s_{out}$.

Figure 9 gives the labelled transition system of the controller for the two tanks model. Recall that the labelled transition system does not consider event conditions. Here $C_1 = C_A \parallel C_B \parallel C_{dA}$ and $C_2 = C_A \parallel C_B \parallel C_{dB}$. There are many cycles in this system, but for the purposes of this example, the two of interest are the infinite sequence of events that consists of $\underline{\mathrm{empty}}_A, \underline{\mathrm{empty}}_B$ repeated, and the infinite sequence that consists of $\underline{\mathrm{full}}_A, \underline{\mathrm{full}}_B, \underline{\mathrm{empty}}_A, \underline{\mathrm{empty}}_B$. The I-graph for this controller is given in Figure 10. It only includes vertices that are reachable from vertices where κ consists of ones. As can be seen, it is not possible to obtain the second of the sequences, since $\underline{\mathrm{full}}_A$ inhibits $\underline{\mathrm{empty}}_A$ and $\underline{\mathrm{full}}_B$ does not enable it. In fact, there are no cycles involving $\underline{\mathrm{full}}_A$ or $\underline{\mathrm{full}}_B$. However, the cycle consisting of $\underline{\mathrm{empty}}_A, \underline{\mathrm{empty}}_B$ is possible, and it occurs when the levels of tanks A and B are both initialised to zero. Otherwise, by induction on the number of $\underline{\mathrm{empty}}_A$ and $\underline{\mathrm{empty}}_B$ that have fired, it can be shown that the cycle does not happen. This example shows both the limits of I-graphs but also how it is possible to detect undesirable situations.

Note that by having a separate controller, we are able to reason about the behaviour of the model, and either show that certain bad behaviours do not happen, or at least alert the modeller to the possibilities of these behaviours, which then may be ruled out by further investigation of specific parameters of the model.

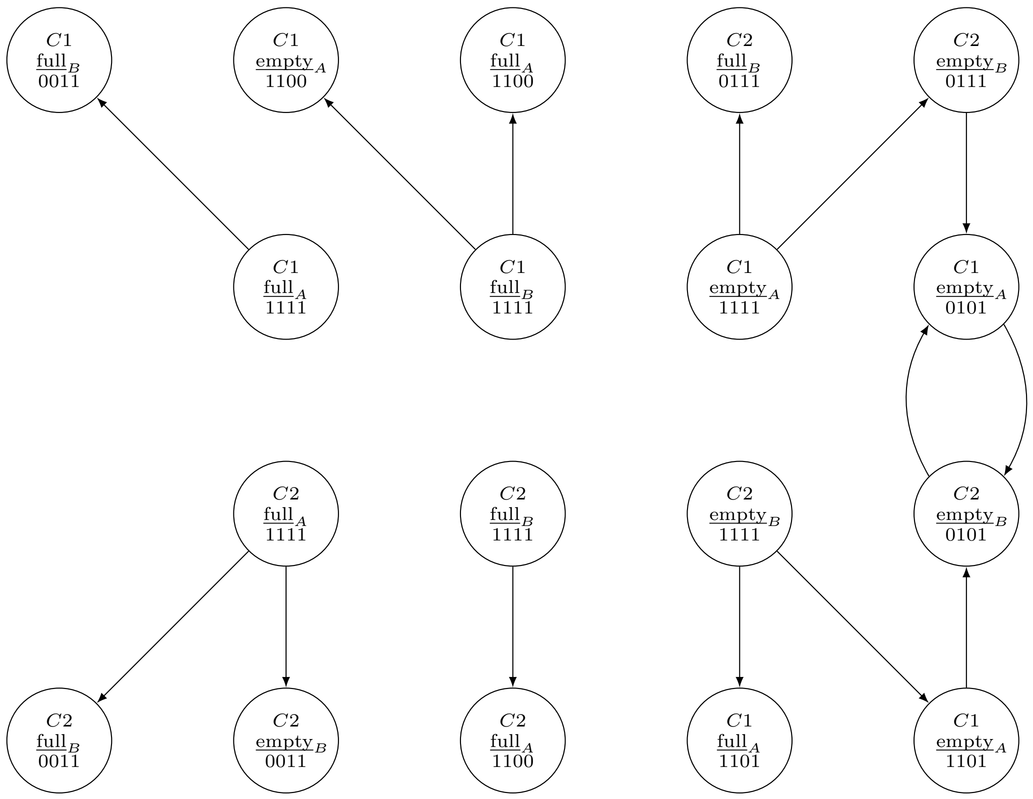

$C1$ $\underline{\text{full}}_B$ 0011
$C1$ $\underline{\text{empty}}_A$ 1100
$C1$ $\underline{\text{full}}_A$ 1100
$C2$ $\underline{\text{full}}_B$ 0111
$C2$ $\underline{\text{empty}}_B$ 0111
$C1$ $\underline{\text{full}}_A$ 1111
$C1$ $\underline{\text{full}}_B$ 1111
$C1$ $\underline{\text{empty}}_A$ 1111
$C1$ $\underline{\text{empty}}_A$ 0101
$C2$ $\underline{\text{full}}_A$ 1111
$C2$ $\underline{\text{full}}_B$ 1111
$C2$ $\underline{\text{empty}}_B$ 1111
$C2$ $\underline{\text{empty}}_B$ 0101
$C2$ $\underline{\text{full}}_B$ 0011
$C2$ $\underline{\text{empty}}_B$ 0011
$C2$ $\underline{\text{full}}_A$ 1100
$C1$ $\underline{\text{full}}_A$ 1101
$C1$ $\underline{\text{empty}}_A$ 1101

Fig. 10. I-graph for the two tanks model where the order of the κ vector is $\underline{\text{full}}_A$, $\underline{\text{empty}}_A$, $\underline{\text{full}}_B$, $\underline{\text{empty}}_B$.

6.1. Ensuring well-behavedness

We now define the property of the I-graph that ensures its HYPE model is well-behaved.

Theorem 6.1. A HYPE model with an acyclic I-graph is well-behaved.

Earlier, we defined a HYPE model as well-behaved by the absence of infinite sequences of instantaneous events. Since the detailed behaviour of a HYPE model is defined in terms of a hybrid automaton, this result must be proved in terms of hybrid automata and we introduce the appropriate definitions to achieve this. First, we define a notion of behaviour in a hybrid automaton.

Definition 6.7. Given a hybrid automaton $\mathcal{H}$, *a sequence of simultaneous instantaneous events* is a finite or infinite trace

$$(v_{i_1}, \mathbf{x}_{i_1}) \xrightarrow{\mathrm{a}_{i_1}} (v_{i_2}, \mathbf{x}_{i_2}) \xrightarrow{\mathrm{a}_{i_2}} (v_{i_3}, \mathbf{x}_{i_3}) \xrightarrow{\mathrm{a}_{i_3}} \dots$$

such that the events happen in the same time instance, $act(\underline{\mathrm{a}}_{i_k})$ is true for $\mathbf{x}_{i_k}$ and $res(\underline{\mathrm{a}}_{i_k})$ maps $\mathbf{x}_{i_k}$ to $\mathbf{x}_{i_{k+1}}$

This implies that $act(\underline{\mathrm{a}}_{i_k}) \neq \perp$ for all i_k which is correct because we are only considering events that happen in the same time point rather than after some unspecified delay. From this definition, we can prove the following lemma.

Lemma 6.1. Given a HYPE model P, let $(v_i, \mathbf{x}_i) \xrightarrow{\mathrm{a}_i} (v_j, \mathbf{x}_j) \xrightarrow{\mathrm{a}_j}$ be a sequence of simultaneous instantaneous transitions in $\mathcal{H}(P)$ then $\mathbf{x}_j \in R_{\underline{\mathrm{a}}_i} \cap G_{\underline{\mathrm{a}}_j}$.

Proof. By definition of the behaviour of hybrid automata. □

Definition 6.8. Given a well-defined HYPE model $P \overset{\mathit{def}}{=} \Sigma \underset{*}{\bowtie} \underline{\mathrm{init}}.\mathit{Con}$, and a configuration $v = \langle \Sigma \underset{*}{\bowtie} C, \sigma\rangle$ in $\mathcal{H}(P)$, $\mathit{con}(v) = C$. For an automaton state $(v, \mathbf{x})$, $\mathit{con}((v, \mathbf{x})) = \mathit{con}(v)$.

Since P is well-defined, the model must have the form $\langle \Sigma \underset{*}{\bowtie} C, \sigma\rangle$ and additionally, $C \in \mathrm{ds}(\underline{\mathrm{init}}.\mathit{Con})$, both from Proposition 5.2.

We now have the following results about I-graphs relating them to the behaviour of a hybrid automaton.

Lemma 6.2. Given a HYPE model P, let $(v_{i_1}, \mathbf{x}_{i_1}) \xrightarrow{\underline{\mathrm{a}}_{i_1}} (v_{i_2}, \mathbf{x}_{i_2}) \xrightarrow{\underline{\mathrm{a}}_{i_2}} (v_{i_3}, \mathbf{x}_{i_3}) \xrightarrow{\underline{\mathrm{a}}_{i_3}} \ldots$ be an infinite or finite sequence of instantaneous events firing in the same time instant in $\mathcal{H}(P)$. Then

$$(\mathit{con}(v_{i_1}), \underline{\mathrm{a}}_{i_1}, 1^h) \to (\mathit{con}(v_{i_2}), \underline{\mathrm{a}}_{i_2}, \kappa_{i_2}) \to (\mathit{con}(v_{i_3}), \underline{\mathrm{a}}_{i_3}, \kappa_{i_3}) \to \ldots$$

is a path in G, the I-graph of P where $\kappa_{i_{k+1}}$ is constructed from κ_{i_k} as described in Definition 6.6.

Proof. First we consider the induction step. Suppose $(v_{i_{k-1}}, \mathbf{x}_{i_{k-1}}) \xrightarrow{\underline{\mathrm{a}}_{i_{k-1}}} (v_{i_k}, \mathbf{x}_{i_k}) \xrightarrow{\underline{\mathrm{a}}_{i_k}} (v_{i_{k+1}}, \mathbf{x}_{i_{k+1}}) \xrightarrow{\underline{\mathrm{a}}_{i_{k+1}}}$ and we know that $(\mathit{con}(v_{i_{k-1}}), \underline{\mathrm{a}}_{i_{k-1}}, \kappa_{i_{k-1}}) \to (\mathit{con}(v_{i_k}), \underline{\mathrm{a}}_{i_k}, \kappa_{i_k})$. We need to show that the I-graph contains the edge $(\mathit{con}(v_{i_k}), \underline{\mathrm{a}}_{i_k}, \kappa_{i_k}) \to (\mathit{con}(v_{i_{k+1}}), \underline{\mathrm{a}}_{i_{k+1}}, \kappa_{i_{k+1}})$. The transition $(v_{i_k}, \mathbf{x}_{i_k}) \xrightarrow{\underline{\mathrm{a}}_{i_k}} (v_{i_{k+1}}, \mathbf{x}_{i_{k+1}})$ is only possible if there is a transition in the labelled transition system of P, and moreover, this is only possible if there is such a transition in the controller, so $\langle \mathit{con}(v_{i_k}), \sigma\rangle \xrightarrow{\underline{\mathrm{a}}_{i_k}} \langle \mathit{con}(v_{i_{k+1}}), \sigma\rangle$.

Since $(v_{i_k}, \mathbf{x}_{i_k}) \xrightarrow{\underline{\mathrm{a}}_{i_k}} (v_{i_{k+1}}, \mathbf{x}_{i_{k+1}}) \xrightarrow{\underline{\mathrm{a}}_{i_{k+1}}}$ and by Lemma 6.1, $\underline{\mathrm{a}}_{i_k}$ must have been enabled hence $\kappa_{i_k}[i_k] = 1$. Similarly, since $(v_{i_{k+1}}, \mathbf{x}_{i_{k+1}}) \xrightarrow{\underline{\mathrm{a}}_{k+1}}$, we know $\kappa_{i_{k+1}}[i_{k+1}] = 1$. Finally, κ_{i_k} is correct by induction and $\kappa_{i_{k+1}}$ is an update of κ_{i_k} taking into account which events a_{i_k} has enabled and inhibited. Since there is a node with $C_{i_{k+1}}$ and $\underline{\mathrm{a}}_{i_{k+1}}$ for each possible κ, the edge can be constructed. Hence the four conditions to be satisfied for an edge in the I-graph have been satisfied.

Returning to the base case, the same argument can be used for the first condition and third condition, the second condition is immediate and the fourth condition follows since κ_{i_2} captures which events are inhibited by $\underline{\mathrm{a}}_{i_1}$. □

This lemma allows us to now prove the theorem stated at the start of this section, and thereby characterise the I-graphs that give us well-behaved HYPE models.

Proof of Theorem 6.1: For a HYPE model P, we need to show that $\mathcal{H}(P)$ has a finite number of finite sequences of instantaneous events that can fire in the same time instance and these sequences are independent of the initial state of the system.

From Lemma 6.2, the I-graph of a HYPE model captures the sequences of instantaneous events which can occur in the execution of $\mathcal{H}(P)$ since by construction, all events appear in the controller. If there are no cycles in the I-graph, which is finite by construction, then there are no infinite sequences possible in the execution. Since an I-graph is finite, there can only be a finite number of finite sequences in $\mathcal{H}(P)$. Finally, since there are nodes in the I-graph where the vector κ consist of 1's which means that all events are active, this graph is independent of the initial state. □

6.2. Results for well-behaved HYPE models

We have now defined well-behavedness for HYPE models and shown how the I-graph of the model can be used to check for this behaviour. Additionally, this has been related to the hybrid semantics of the models. We now consider the implications of this definition and result and how we can ensure that it is a decidable property. As mentioned above, it is not always practicable to determine $G_{\underline{\mathrm{a}}}$ or $R_{\underline{\mathrm{a}}}$. We can extend the definition of an I-graph as follows.

Definition 6.9. Consider a HYPE model with $R_{\underline{\mathrm{a}}} = r_{\underline{\mathrm{a}}}(G_{\underline{\mathrm{a}}})$ for each event $\underline{\mathrm{a}}$, and let $R'_{\underline{\mathrm{a}}}, G'_{\underline{\mathrm{a}}} \subseteq \mathbb{R}^n$, be a collection of sets such that for all $\underline{\mathrm{a}}$, $R_{\underline{\mathrm{a}}} \subseteq R'_{\underline{\mathrm{a}}}$ and $G_{\underline{\mathrm{a}}} \subseteq G'_{\underline{\mathrm{a}}}$. These sets are called *extended sets*.

Definition 6.10. An *extended I-graph* of a HYPE model is defined in the same manner as its I-graph but where extended sets are used to determine enablers and inhibitors.

Our result from the previous section can be carried over to extended sets and hence if the exact sets are not computable or very hard to compute, we can work with an overapproximation.

Theorem 6.2. A HYPE model with an acyclic extended I-graph is well-behaved.

Proof. By increasing the size of the sets of values for activation conditions and resets, additional edges may be added to the I-graph but no edge can be removed. Hence if the extended I-graph is acyclic, so is the I-graph. $\square$

Corollary 6.1. Checking whether a HYPE model with computable extended sets has an acyclic I-graph is decidable.

Proof. Since an I-graph is finite, and the sets can be computed, it can be effectively constructed and checked for acyclicity. $\square$

If we know the controller has a specific cyclic form, we can also investigate conditions that will prevent a cycle from occurring in the I-graph. These results apply in the case of extended sets as well.

Proposition 6.1. Let P be a HYPE model with $Con \overset{def}{=} \underline{\mathrm{a}}_1.\ldots.\underline{\mathrm{a}}_n.Con$. If the resets for all $\underline{\mathrm{a}}_i$ are the identity and $\cap_{\underline{\mathrm{a}}_i \in Con} G_{\underline{\mathrm{a}}_i} = \emptyset$ then P is well-behaved. It is straightforward to see that $R_{\underline{\mathrm{a}}} = G_{\underline{\mathrm{a}}}$ and at some point there is no overlap between the sets of two different events.

Proof. The only possible cycle is that consisting of the events $\underline{\mathrm{a}}_1, \ldots, \underline{\mathrm{a}}_n$ repeated. Moreover, $\cap_{\underline{\mathrm{a}}_i \in Con} G_{\underline{\mathrm{a}}_i} = \emptyset$ implies that there exist j and k such that $G_{\underline{\mathrm{a}}_j} \cap G_{\underline{\mathrm{a}}_k} = \emptyset$. Additionally $R_{\underline{\mathrm{a}}_i} = G_{\underline{\mathrm{a}}_i}$ for all i. Hence $R_{\underline{\mathrm{a}}_j} \cap G_{\underline{\mathrm{a}}_k} = \emptyset$ and $R_{\underline{\mathrm{a}}_j} \cap G_{\underline{\mathrm{a}}_k} = \emptyset$ and hence these events inhibit each other and prevent the formation of a cycle in the I-graph. $\square$

Proposition 6.2. Let P be a HYPE model with $Con \overset{def}{=} \underline{\mathrm{a}}_1.\ldots.\underline{\mathrm{a}}_n.Con$. If there exists i such that $R_{\underline{\mathrm{a}}_i} \cap G_{\underline{\mathrm{a}}_{i+1}}$ is empty (where addition is modulo n) then P is well-behaved.

Proof. The lack of overlap of values at one point in the sequence of the controller prevents a cycle in the I-graph. $\square$

Now that we have presented the basic theory of well-definedness, we can consider how this can be applied compositionally. We prove the results for the collections of sets $G_{\underline{\mathrm{a}}}$ and $R_{\underline{\mathrm{a}}}$, but since none of the results depend on the set being minimal, they also apply to extended sets. The first result is to be expected since the events of the two systems cannot affect each other.

Proposition 6.3. Let Con and Con' be two controllers such that all events in Con are pairwise independent from those of Con'. If Con and Con' are well-behaved then $Con \parallel Con'$ is well-behaved.

Proof. The independence of events implies that $\mathsf{ev}(Con) \cap \mathsf{ev}(Con') = \emptyset$, hence the use of $\parallel$. Let the G be the I-graph of $Con \underset{*}{\Join} Con'$. Its vertices are in the set $ds(\underline{\mathrm{init}}.(Con \underset{*}{\Join} Con')) \times (\mathsf{ev}(Con) \cup \mathsf{ev}(Con')) \times \{0,1\}^h$. Assume there is an cycle in G. We will show it is possible to construct a cycle in H, the I-graph of Con.

Let $(C_1 \underset{*}{\Join} C'_1, \underline{\mathrm{a}}_1, \kappa_1) \to (C_2 \underset{*}{\Join} C'_2, \underline{\mathrm{a}}_2, \kappa_2) \to (C_3 \underset{*}{\Join} C'_3, \underline{\mathrm{a}}_3, \kappa_3)$ be a subpath of the cycle in G such that $\underline{\mathrm{a}}_3 \in \mathsf{ev}(Con)$ and $\underline{\mathrm{a}}_2 \in \mathsf{ev}(Con')$. Since the events of the two controllers are disjoint, $C_2 = C_3$. Since the events of Con and Con' are pairwise independent, $\underline{\mathrm{a}}_2$ can have no effect on the activation of $\underline{\mathrm{a}}_3$ and hence $\kappa_2[3]$ must be 1.

Therefore, we know there is an edge $(C_1 \underset{*}{\Join} C'_1, \underline{\mathrm{a}}_1, \kappa_1) \to (C_3 \underset{*}{\Join} C'_3, \underline{\mathrm{a}}_3, \kappa_3)$ and $(C_2 \underset{*}{\Join} C'_2, \underline{\mathrm{b}}_2, \kappa_2)$ can be removed from the cycle, while ensuring a cycle remains. Similarly all other events from Con' can be removed from the cycle and the remaining cycle will consist only of events from Con_1. If we modify the vertices of G by changing any $C_i \underset{*}{\Join} C'_i$ to C_i and appropriately removing elements from the vectors, we obtain vertices from H. Moreover, this cycle must appear in H since it is constructed purely with reference to events in Con. Thus, we have a contradiction. $\square$

Proposition 6.4. Let Con and Con' be two controllers such that for all $\underline{\mathrm{a}} \in \mathsf{ev}(Con) \setminus \mathsf{ev}(Con')$ and for $\underline{\mathrm{a}}' \in \mathsf{ev}(Con') \setminus \mathsf{ev}(Con)$, no $\underline{\mathrm{a}}$ activates an $\underline{\mathrm{a}}'$ and no $\underline{\mathrm{a}}'$ activates an $\underline{\mathrm{a}}$. If Con and Con' are well-behaved then $Con \underset{*}{\Join} Con'$ is well-behaved.

Proof. Let the G be the I-graph of $Con \underset{*}{\Join} Con'$ with vertices $ds(\underline{\mathrm{init}}.(Con \underset{*}{\Join} Con')) \times (\mathsf{ev}(Con) \cup \mathsf{ev}(Con'))$ $\times \{0,1\}^h$. Assume there is a cycle in G. We will show this implies that there is a cycle in H, the I-graph of Con.

Let $(C_1 \underset{*}{\Join} C'_1, \underline{\mathrm{a}}_1, \kappa_1) \to (C_2 \underset{*}{\Join} C'_2, \underline{\mathrm{a}}_2, \kappa_2) \to (C_3 \underset{*}{\Join} C'_3, \underline{\mathrm{a}}_3, \kappa_3)$ be a subpath of the cycle in G. First consider the case where $\underline{\mathrm{a}}_3 \in \mathsf{ev}(Con)$ and $\underline{\mathrm{a}}_2 \in \mathsf{ev}(Con')$ and $\underline{\mathrm{a}}_2, \underline{\mathrm{a}}_3 \notin \mathsf{ev}(Con) \cap \mathsf{ev}(Con')$. Since the events are in separate controllers, $C_2 = C_3$. Since the unshared events of Con and Con' cannot activate each other, $\underline{\mathrm{a}}_2$ can have no effect on the activation of $\underline{\mathrm{a}}_3$ and hence $\kappa_2[3]$ must be 1.

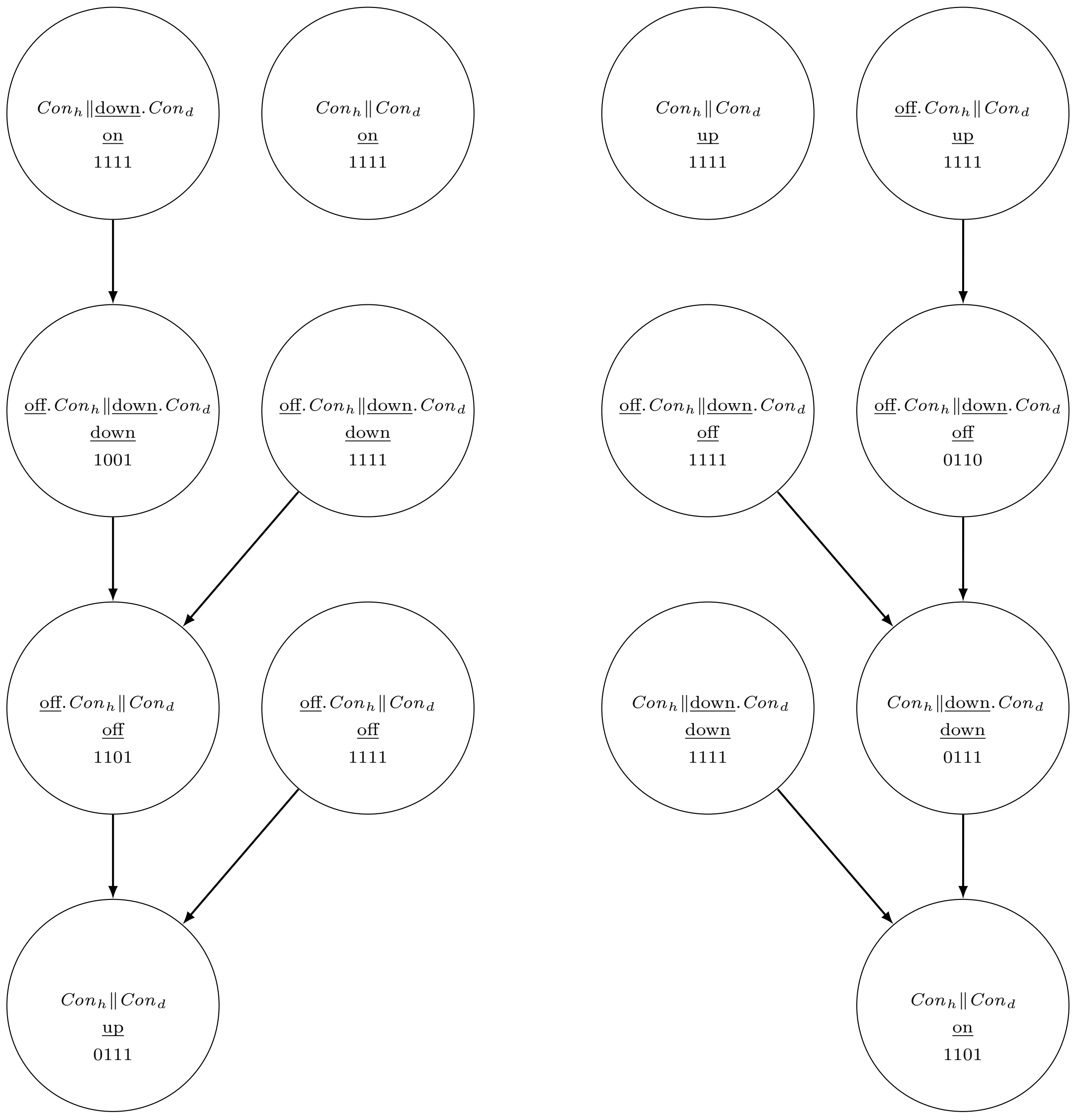

$Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{on}}$ 1111
$Con_h \| Con_d$ $\underline{\mathrm{on}}$ 1111
$Con_h \| Con_d$ $\underline{\mathrm{up}}$ 1111
$\underline{\mathrm{off}}.Con_h \| Con_d$ $\underline{\mathrm{up}}$ 1111
$\underline{\mathrm{off}}.Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{down}}$ 1001
$\underline{\mathrm{off}}.Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{down}}$ 1111
$\underline{\mathrm{off}}.Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{off}}$ 1111
$\underline{\mathrm{off}}.Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{off}}$ 0110
$\underline{\mathrm{off}}.Con_h \| Con_d$ $\underline{\mathrm{off}}$ 1101
$\underline{\mathrm{off}}.Con_h \| Con_d$ $\underline{\mathrm{off}}$ 1111
$Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{down}}$ 1111
$Con_h \| \underline{\mathrm{down}}.Con_d$ $\underline{\mathrm{down}}$ 0111
$Con_h \| Con_d$ $\underline{\mathrm{up}}$ 0111
$Con_h \| Con_d$ $\underline{\mathrm{on}}$ 1101

Fig. 11. I-graph for $Con_h \parallel Con_d$ where the order of the κ vector is $\underline{\mathrm{on}}$, $\underline{\mathrm{off}}$, $\underline{\mathrm{up}}$, $\underline{\mathrm{down}}$.

Therefore, we know there is an edge $(C_1 \bowtie_* C'_1, \underline{\mathrm{a}}_1, \kappa_1) \to (C_3 \bowtie_* C'_3, \underline{\mathrm{a}}_3, \kappa_3)$ and $(C_2 \bowtie_* C'_2, \underline{\mathrm{b}}_2, \kappa_2)$ can be removed from the cycle, while ensuring a cycle remains. Similarly all other events from Con' can be removed from the cycle and the remaining cycle will consist only of unshared events from Con_1, and events that occur in both controllers.

Next let $(C_1 \bowtie_* C'_1, \underline{\mathrm{a}}_1, \kappa_1) \to (C_2 \bowtie_* C'_2, \underline{\mathrm{a}}_2, \kappa_2) \to (C_3 \bowtie_* C'_3, \underline{\mathrm{a}}_3, \kappa_3)$ be a subpath of this reduced cycle in G with $\underline{\mathrm{a}}_1, \underline{\mathrm{a}}_3 \in \mathsf{ev}(Con) \setminus \mathsf{ev}(Con')$ and $\underline{\mathrm{a}}_2 \in \mathsf{ev}(Con) \cap \mathsf{ev}(Con')$. By the definition of I-graphs, there must be a transition $\langle C_2 \bowtie_* C'_2, \sigma\rangle \xrightarrow{\underline{\mathrm{a}}_i} \langle C_3 \bowtie_* C'_3, \sigma\rangle$ in the labelled transition system of $Con \bowtie_* Con'$, hence we know that the transition $\langle C_2, \sigma\rangle \xrightarrow{\underline{\mathrm{a}}_i} \langle C_3, \sigma\rangle$ exists, because of the use of $\bowtie_*$. If we can show that this transition leads to an edge in the graph, then by the same reasoning in the previous proof, we can conclude that we have a cycle in H.

We consider a path $(C_1, \underline{\mathrm{a}}_1, \kappa'_1) \to (C_2, \underline{\mathrm{a}}_2, \kappa'_2) \to (C_3, \underline{\mathrm{a}}_3, \kappa'_3)$. We require $\kappa'_3[3] = 1$ to ensure the existence of the second edge, regardless of the value of $\kappa_1'[3]$ and $\kappa_2'[3]$. If $\kappa_1'[3] = 1$, then $\underline{\mathrm{a}}_1$ and $\underline{\mathrm{a}}_2$ must not inhibit $\underline{\mathrm{a}}_3$, and if $\kappa_1'[3] = 0$, $\underline{\mathrm{a}}_2$ must activate $\underline{\mathrm{a}}_3$. But we know this already from the path in the reduced cycle in G. Hence we can conclude we have a cycle in H and we have shown a contradiction. $\square$

Finally, in this section, we consider the two examples. For the two tanks example, we can apply Proposition 6.3 to determine that the system with $Con_A \parallel Con_B$ is well-behaved. Since the I-graph of Con_{dB} contains a cycle, we cannot go further than this.

Considering the orbiter, if $k_2 < k_1$, then Con_h has well-behaved resets by Proposition 6.1 and similarly, if $k_4 < k_3$, then Con_d has well-behaved resets. However, neither Proposition 6.3 nor Proposition 6.4 apply because the events are not independent and activation is possible between events in each controller. Hence, the I-graph of $Con_h \parallel Con_d$ must be constructed using the fact that $k_1 < k_4$ which allows us to determine the enabling and inhibition of events by the other events. Figure 11 shows the I-graph, which is acyclic. Additionally, Con_s has well-behaved resets by Proposition 6.2. Hence, using Proposition 6.3, the system with the controller $(Con_h \underset{*}{\bowtie} Con_d) \underset{*}{\bowtie} Con_s$ is well-behaved.

The condition of having an acyclic I-graph is a reasonable combination of practical expressiveness and effective computability. It allows us to determine which controllers surely do not have instantaneous Zeno behaviour through a simple graph construction. Moreover, by choosing computable sets, we can ensure decidability of the process. In our experience, for reasonable models, the acyclicity condition should not yield too many false positives. Also, I-graphs are unlikely to be too large as only few vertices can be reached from those where κ is the vector consisting of ones and construction can be done on the fly.

Note that the I-graph construction does not take into account the initial values of variables. The fact that κ is initially set to ones in the construction means that all events are initially possible. This has the advantage that the I-graph applies to all behaviours of the HYPE model, regardless of initial values but the disadvantage that this leads to an overapproximation of behaviour. An alternative approach would be to construct a graph from the initial conditions, but this would require a more complex algorithm with difficult termination conditions. This would have been more precise in the sense of being a smaller overapproximation; however, the cost does not appear to be justified by the benefits. As we can see from the two examples, the approach taken here, is sufficient to reason about these controllers.

To sum up, the separate definition of the controller in the HYPE model, together with the event conditions allow us to reason about the behaviour of the model without considering its detailed behaviour in terms of a hybrid automaton. This supports the design of models that exclude unwanted behaviour.

7. Comparison with hybrid automata

We now compare the expressive power of HYPE with that of hybrid automata at the level of composition of models. We first need to describe how two different HYPE systems can be composed in a synchronisation. The explanation of the choices made are given after the definition.

Definition 7.1. Let $P_i \stackrel{\text{def}}{=} \Sigma_i \underset{*}{\bowtie} \underline{\mathrm{init}}.Con_i$ for $i = 1, 2$ be two well-defined HYPE systems given by (P_i, $\mathcal{V}_i$, IN_i, IT_i, $\mathcal{E}_i$, $\mathcal{A}_i$, ec_i, iv_i, EC_i, ID_i) respectively. The *synchronisation* of P_1 and P_2 ($P_1 \otimes P_2$) is the HYPE model (P, $\mathcal{V}_1 \cup \mathcal{V}_2$, $IN_1 \cup IN_2$, $IT_1 \cup IT_2$, $\mathcal{E}_1 \cup \mathcal{E}_2$, $\mathcal{A}_1 \cup \mathcal{A}_2$, ec, $iv_1 \cup iv_2$, $EC_1 \cup EC_2$, $ID_1 \cup ID_2$) which is defined whenever $IN_1 \cap IN_2 = \emptyset$ and $IT_1 \cap IT_2 = \emptyset$ and where the well-defined system is $P \stackrel{\text{def}}{=} (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} \underline{\mathrm{init}}.(Con_1 \underset{*}{\bowtie} Con_2)$ and ec is defined by

$$ec(\underline{\mathrm{a}}) = \begin{cases} ec_1(\underline{\mathrm{a}}) & \text{if } \underline{\mathrm{a}} \notin \mathcal{E}_2 \\ ec_2(\underline{\mathrm{a}}) & \text{if } \underline{\mathrm{a}} \notin \mathcal{E}_1 \\ (act_1(\underline{\mathrm{a}}) \wedge act_2(\underline{\mathrm{a}}), res_1(\underline{\mathrm{a}}) \wedge res_2(\underline{\mathrm{a}})) & \text{if } \underline{\mathrm{a}} \in \mathcal{E}_1 \cap \mathcal{E}_2 \end{cases}$$

The condition $IN_1 \cap IN_2 = \emptyset$ requires influence names to be disjoint. This is necessary to ensure a well-defined synchronisation. Note that this condition implies that $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$ and $iv_1 \cap iv_2 = \emptyset$. In the case where the same influence name is used in two different models, one can be renamed to a fresh name, as they are essentially arbitrary names used to identify flows (unlike variables which have specific meanings). After renaming, the synchronisation is well-defined.

The condition $IT_1 \cap IT_2 = \emptyset$ requires that influence types are disjoint. It would be possible to allow shared influence types with identical definitions up to substitution of variable names based on a bijection between the variables, but our choice is to go for the simpler definition.

Once these conditions hold, the union of most of the sets can be taken straightforwardly. This allows for influences from the different models to map to a shared variable which is an important aspect of the synchronisation definition. For the definition of *ec*, we need to define $\bot \wedge \phi = \phi \wedge \bot = \phi$ and $\bot \wedge \bot = \bot$. This means that for a shared event, it is only non-urgent if it is non-urgent in both components. It is reasonable to treat urgency as stronger than non-urgency.

Proposition 7.1. Given two well-defined HYPE models P_1 and P_2, $P_1 \otimes P_2$ is well-defined.

Proof. The first condition about the form of subcomponents holds because it holds in both models. The second condition about disjointness of influences between subcomponents holds because influences are disjoint between models. The conditions requiring cooperation on all shared events hold by the definition of the combined controlled systems. Since the two models are well-defined, $\mathsf{ev}(\Sigma_i) = \mathcal{E}_i$, $\mathsf{in}(\Sigma_i) = \mathit{IN}_i$, $\mathsf{ev}(\mathit{Con}_i) \subseteq \mathsf{ev}(\Sigma_i)$ for $i = 1, 2$, and hence we know that $\mathsf{ev}(\Sigma_1 \bowtie_* \Sigma_2) = \mathcal{E}_1 \cup \mathcal{E}_2$, $\mathsf{in}(\Sigma_1 \bowtie_* \Sigma_2) = \mathit{IN}_1 \cup \mathit{IN}_2$ and $\mathsf{ev}(\mathit{Con}_1 \bowtie_* \mathit{Con}_2) \subseteq \mathsf{ev}(\Sigma_1 \bowtie_* \Sigma_2)$. □

To compare synchronisation of HYPE models with a similar construction on hybrid automata, we work with the definition of product as given in [HH94] modified to take account of urgent actions.

Definition 7.2. Let H_1 and H_2 be two hybrid automata with $\mathcal{H}_i = (V_i, E_i, \mathbf{X}_i, \mathcal{E}_i, \mathit{flow}_i, \mathit{init}_i, \mathit{inv}_i, \mathit{event}_i, \mathit{jump}_i, \mathit{reset}_i, \mathit{urgent}_i)$ for $i = 1, 2$. The *synchronised product (or parallel composition)* of H_1 and H_2 ($H_1 \times H_2$) is $H = (V_1 \times V_2, E, \mathbf{X}_1 \cup \mathbf{X}_2, \mathcal{E}_1 \cup \mathcal{E}_2, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ where

- $(e_1, e_2) = ((v_1, v_2), (v_1', v_2')) \in E$ if
  1. $v_2 = v_2'$ and $\mathit{event}_1(e_1) \notin \mathcal{E}_2$ or
  2. $v_1 = v_1'$ and $\mathit{event}_2(e_2) \notin \mathcal{E}_1$ or
  3. $\mathit{event}_1(e_1) = \mathit{event}_2(e_1) \in \mathcal{E}_1 \cap \mathcal{E}_2$
- if $e = (e_1, e_2)$ with
  1. $\mathit{event}_1(e_1) \notin \mathcal{E}_2$ then $\mathit{event}(e) = \mathit{event}_1(e_1)$, $\mathit{jump}(e) = \mathit{jump}_1(e_1)$, $\mathit{reset}(e) = \mathit{reset}_1(e_1)$, $\mathit{urgent}(e) = \mathit{urgent}_1(e_1)$
  2. $\mathit{event}_2(e_2) \notin \mathcal{E}_1$ then $\mathit{event}(e) = \mathit{event}_2(e_2)$, $\mathit{jump}(e) = \mathit{jump}_2(e_2)$, $\mathit{reset}(e) = \mathit{reset}_2(e_2)$, $\mathit{urgent}(e) = \mathit{urgent}_2(e_2)$
  3. $\mathit{event}_1(e_1) = \mathit{event}_2(e_2) \in \mathcal{E}_1 \cap \mathcal{E}_2$ then $\mathit{event}(e) = \mathit{event}_1(e_1)$, $\mathit{jump}(e) = \mathit{jump}_1(e_1) \wedge \mathit{jump}_2(e_2)$, $\mathit{reset}(e) = \mathit{reset}_1(e_1) \wedge \mathit{reset}_2(e_2)$, $\mathit{urgent}(e) = \mathit{urgent}_1(e_1) \vee \mathit{urgent}_2(e_2)$,
- if $v = (v_1, v_2)$ then $\mathit{flow}(v) = \mathit{flow}_1(v_1) \wedge \mathit{flow}_2(v_2)$, $\mathit{init}(v) = \mathit{init}_1(v_1) \wedge \mathit{init}_2(v_2)$, $\mathit{inv}(v) = \mathit{inv}_1(v_1) \wedge \mathit{inv}_2(v_2)$.

For this definition to give a meaningful product, whenever $X \in \mathbf{X}_1 \cap \mathbf{X}_2$, we require that $\mathit{flow}_1(v_1)[X] = \mathit{flow}_2(v_2)[X]$ for a vertex (v_1, v_2). If this is not the case, *flow* will evaluate to false for X. For consistency with the definition of synchronisation, we consider urgency stronger than non-urgency and hence $\mathit{urgent}(e) = \mathit{urgent}_1(e_2) \vee \mathit{urgent}_2(e_2)$.

Now that we have a definition of product for HYPE models and synchronised product for hybrid automata, we can compare them and identify whether they have the same or different expressive power.

Theorem 7.1. Let P_1 and P_2 be well-defined HYPE systems $(P_i, \mathcal{V}_i, \mathit{IN}_i, \mathit{IT}_i, \mathcal{E}_i, \mathcal{A}_i, \mathit{ec}_i, \mathit{iv}_i, \mathit{EC}_i, \mathit{ID}_i)$ for $i = 1, 2$. If $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ then $\mathcal{H}(P_1 \otimes P_2) = \mathcal{H}(P_1) \times \mathcal{H}(P_2)$ when vertices unreachable from the initial vertex are excluded.

Proof. See Appendix A. □

Corollary 7.1. Let P_1 and P_2 be well-defined HYPE systems $(P_i, \mathcal{V}_i, \mathit{IN}_i, \mathit{IT}_i, \mathcal{E}_i, \mathcal{A}_i, \mathit{ec}_i, \mathit{iv}_i, \mathit{EC}_i, \mathit{ID}_i)$ for $i = 1, 2$, with hybrid automata $\mathcal{H}(P_i) = (V_i, E_i, \mathbf{X}_i, \mathcal{E}_i\ \mathit{flow}_i, \mathit{init}_i, \mathit{inv}_i, \mathit{event}_i, \mathit{jump}_i, \mathit{reset}_i, \mathit{urgent}_i)$, synchronisation $P_1 \otimes P_2$, and synchronised product hybrid automaton $\mathcal{H}(P_1 \otimes P_2) = (V, E, \mathbf{X}, \mathcal{E}\ \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$. If $\mathbf{X}_1 \cap \mathbf{X}_2 \neq \emptyset$, there exists $X \in \mathbf{X}_1 \cap \mathbf{X}_2$ with flow not constant zero at some mode, and $\mathcal{H}(P_1) \times \mathcal{H}(P_2)$ is defined then $\mathcal{H}(P_1 \otimes P_2) \neq \mathcal{H}(P_1) \times \mathcal{H}(P_2)$.

Proof. Since $\mathcal{H}(P_1) \times \mathcal{H}(P_2)$ is defined, then for all $X \in \mathbf{X}_1 \cap \mathbf{X}_2$, and for all vertices, $v = (v_1, v_2)$ in $\mathcal{H}(P_1) \times \mathcal{H}(P_2)$, $\mathit{flow}_1(v_1)[X] = \mathit{flow}_2(v_2)[X]$. Each $\mathit{flow}_i(v_i)[X] = \sum\{r[\![I(\mathcal{W})]\!] \mid \mathit{iv}(\iota) = X \text{ and } \sigma_i(\iota) =$

$(r, I(\mathcal{W}))\}$ where $v_i = \langle \Sigma_i \underset{*}{\bowtie} D_i, \sigma_i \rangle$. Without considering the details of the right hand side of the equation, we can write $\mathit{flow}_1(v_1)[X] = \mathit{flow}_2(v_2)[X] = f(\mathbf{X})$.

Using the technique of the proof of Theorem 7.1, we can conclude that $\langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1 \underset{*}{\bowtie} D_2), \sigma_1 \cup \sigma_2 \rangle$ is a configuration in the labelled transition system of $P_1 \otimes P_2$ and hence it is a vertex in $\mathcal{H}(P_1 \otimes P_2)$, say v'. Hence $\mathit{flow}(v')[X] = \sum\{r[\![I(\mathcal{W})]\!] \mid iv(\iota) = X \text{ and } (\sigma_1 \cup \sigma_2)(\iota) = (r, I(\mathcal{W}))\}$. Since $IN_1 \cap IN_2 = \emptyset$, the domains of σ_1 and σ_2 are distinct. From this, we can conclude that $\mathit{flow}(v')[X] = 2f(\mathbf{X})$. □

We have shown that by restricting the HYPE models to those with disjoint sets of variables, they have the same expressive power with respect to compositionality as hybrid automata. The necessity of this restriction is shown by the corollary. When two hybrid automata have shared variables, either their product is undefined or there must be a zero flow for the shared variables, restricting the compositionality of the formalism.

By contrast, in HYPE, influences from each component of the product can map to a shared variable and hence the ODE for that variable can be determined by both components. Therefore compositionality is supported fully by the synchronisation product for HYPE, and allows for useful interaction between models. From this we can conclude that HYPE is more fine-grained and more flexible in its modelling style than hybrid automata. In many cases, changes to a model can be achieved by composition rather than by rewriting the model. In further work, we will investigate how this can provide gains in terms of analysis.

8. Equivalence Semantics

An important technique in the realm of process algebras is that of semantic equivalences, where we are able to define notions that capture the idea of same behaviour. An advantage of working with a process algebra as a language to describe systems is that we aim to understand how we can identify models that have the same behaviour at the semantic level but differ syntactically. Additionally, if we can prove congruence, we are able to substitute one equivalent component or subcomponent for another, in a model. One use of substitution is to replace a component with a smaller but equivalent component to reduce the size of the labelled transition system and the resulting hybrid automaton.

We define equivalent behaviour over the labelled transition system given in Section 3 thereby focussing on the configuration of the system rather than evaluations of continuous variables, allowing us to abstract away from these details. Since our operational semantics apply to more general terms than those specified in our HYPE models, we define our equivalence over those more general terms. Let $\mathcal{C}$ be the set of terms obtained by the grammar $S ::= \underline{\mathrm{a}} : \alpha.S \mid \underline{\mathrm{a}}.S \mid S' + S'$ and $P ::= C \mid P \underset{L}{\bowtie} P$ where $C \overset{\mathit{def}}{=} S$ defines a constant. Note that $\mathcal{C}_{\mathit{Sys}} \subset \mathcal{C}$ and hence we can use compositionality results such as congruence to reason about HYPE models.

We base our equivalence on bisimulation [Mil89], because it is an equivalence that makes strong distinctions relating to issues such as deadlock and branching whereas language/trace equivalence cannot make these distinctions [vG90].

Definition 8.1. A relation $B \subseteq \mathcal{C} \times \mathcal{C}$ is a *system bisimulation* if for all $(P, Q) \in B$, for all $\underline{\mathrm{a}} \in \mathcal{E}$, for all $\sigma \in \mathcal{S}$ whenever

1. $\langle P, \sigma \rangle \xrightarrow{\underline{\mathrm{a}}} \langle P', \sigma' \rangle$, there exists $\langle Q', \sigma' \rangle$ with $\langle Q, \sigma \rangle \xrightarrow{\underline{\mathrm{a}}} \langle Q', \sigma' \rangle$, $(P', Q') \in B$.
2. $\langle Q, \sigma \rangle \xrightarrow{\underline{\mathrm{a}}} \langle Q', \sigma' \rangle$, there exists $\langle P', \sigma' \rangle$ with $\langle P, \sigma \rangle \xrightarrow{\underline{\mathrm{a}}} \langle P', \sigma' \rangle$, $(P', Q') \in B$.

P and Q are *system bisimilar*, $P \sim_s Q$ if they are in a system bisimulation.

It is straightforward to show that system bisimulation is a congruence for our operators.

Theorem 8.1. $\sim_s$ is a congruence for Prefix, Choice and Cooperation.

Proof. Let $P_1 \sim_s P_2$. Symmetrical cases have been omitted.

Prefix with influence We have the transition $\langle \underline{\mathrm{a}} : (\iota, r, I).P_1, \sigma \rangle \xrightarrow{\underline{\mathrm{a}}} \langle P_1, \sigma[\iota \mapsto (r, I)] \rangle$ and likewise the transition $\langle \underline{\mathrm{a}} : (\iota, r, I).P_2, \sigma \rangle \xrightarrow{\underline{\mathrm{a}}} \langle P_2, \sigma[\iota \mapsto (r, I)] \rangle$. Since $P_1 \sim_s P_2$, then we can conclude that $\underline{\mathrm{a}} : (\iota, r, I).P_1 \sim_s \underline{\mathrm{a}} : (\iota, r, I).P_2$.

Prefix without influence Similar to the previous case.

$$D_1 \overset{def}{=} \underline{\mathrm{full}}_A.D_1 + \underline{\mathrm{full}}_B.D_1 + \underline{\mathrm{empty}}_A.D_2 \qquad D_2 \overset{def}{=} \underline{\mathrm{full}}_A.D_2 + \underline{\mathrm{full}}_B.D_2 + \underline{\mathrm{empty}}_B.D_1$$

$$S \overset{def}{=} ((In_A \underset{*}{\bowtie} Out_A) \underset{*}{\bowtie} (In_B \underset{*}{\bowtie} Out_B)) \underset{*}{\bowtie} \underline{\mathrm{init}}.D_1$$

Fig. 12. Alternative controller for two tanks

Choice There are two cases. First if $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1', \sigma'\rangle$, then we have $\langle P_1 + Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1', \sigma'\rangle$ and since $P_1 \sim_s P_2$, $\langle P_2, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2', \sigma'\rangle$ with $P_1' \sim_s P_2'$, and hence $\langle P_2 + Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2', \sigma'\rangle$ as required. Second, $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$ and $\langle P_1 + Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$. Also $\langle P_2 + Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$ and we know $Q' \sim_s Q'$.

Cooperation We need to show that $B = \{(P_1 \underset{L}{\bowtie} Q, P_2 \underset{L}{\bowtie} Q) | P_1 \sim_s P_2\}$ is a system bisimulation. There are three cases. First, if $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1', \sigma'\rangle$ with $\underline{a} \notin L$ then $\langle P_1 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1' \underset{L}{\bowtie} Q, \sigma'\rangle$. Since $P_1 \sim_s P_2$, $\langle P_2, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2', \sigma'\rangle$ with $P_1' \sim_s P_2'$, and hence $\langle P_2 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2' \underset{L}{\bowtie} Q, \sigma'\rangle$ with $(P_1' \underset{L}{\bowtie} Q, P_2' \underset{L}{\bowtie} Q) \in B$ as required. Second, if $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$ with $\underline{a} \notin L$ then the proof is straightforward and clearly $(P_1 \underset{L}{\bowtie} Q', P_2 \underset{L}{\bowtie} Q') \in B$. Third and lastly, $\underline{a} \in L$ and $\langle P_1, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1', \sigma'\rangle$ and $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma''\rangle$, then $\langle P_1 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_1' \underset{L}{\bowtie} Q', \Gamma(\sigma, \sigma', \sigma'')\rangle$. Since $P_1 \sim_s P_2$, $\langle P_2, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2', \sigma'\rangle$ with $P_1' \sim_s P_2'$, and hence $\langle P_2 \underset{L}{\bowtie} Q, \sigma\rangle \xrightarrow{\underline{a}} \langle P_2' \underset{L}{\bowtie} Q', \Gamma(\sigma, \sigma', \sigma'')\rangle$. $(P_1' \underset{L}{\bowtie} Q', P_1' \underset{L}{\bowtie} Q') \in B$ as required.

□

It can be remarked that $\sim_s$ is a *robust* or *stateless bisimulation* [CR03, MRG05]. This style of equivalence, which relates process terms for arbitrary states or data, has been used extensively in hybrid process algebras. Moreover, our transition system specification in Figure 2 conforms to the *process-tyft* format which is a congruence format for stateless bisimulation and this offers an alternative route to proving the theorem above.

Given bisimilar controllers, we can show that an uncontrolled system combined with either will result in the same behaviour in terms of system bisimulation.

Corollary 8.1. Given an uncontrolled systems Σ and two controllers such that $Con_1 \sim_s Con_2$ then $\Sigma \underset{L}{\bowtie} \underline{\mathrm{init}}.Con_1 \sim_s \Sigma \underset{L}{\bowtie} \underline{\mathrm{init}}.Con_2$ if these controlled systems are well-defined.

Proof. By congruence. □

Consider the alternative controller in Figure 12 for the two tanks of Section 6. Controllers $C_A \parallel C_B \parallel C_{dA}$ and D_1 are clearly isomorphic as can be seen from Figure 9, and hence are system bisimilar, because no state changes can happen during the execution of the controllers on their own. We can apply the corollary and conclude that *System* and S are system bisimilar.

The preceding congruence theorem is useful when reasoning about a single model and proving later results about bisimilar systems; however, due to the constrained form of HYPE models, we are also interested in congruence with respect to the synchronisation operator between HYPE models, and we now prove this. We need to lift the definition of system bisimulation to HYPE models, and this is achieved by requiring all elements of the tuple except the first are identical.

Definition 8.2. Given two well-defined HYPE models $(P_i, \mathcal{V}, IN, IT, \mathcal{E}, \mathcal{A}, ec, iv, EC, ID)$ for $i = 1, 2$, they are *system bisimilar as models* $(P_1 \sim_{\mathbf{sm}} P_2)$ if $P_1 \sim_s P_2$.

Theorem 8.2. Let P_1, P_2 and Q be well-defined HYPE models. If $P_1 \sim_{\mathbf{sm}} P_2$ then $P_1 \otimes Q \sim_{\mathbf{sm}} P_2 \otimes Q$ and $Q \otimes P_1 \sim_{\mathbf{sm}} Q \otimes P_2$.

Proof. First, note that the synchronisations are defined because P_1 and P_2 have the same elements in their tuples (excluding the first element). This means that the synchronisations will also have this property and hence can be system bisimilar as models.

We need to construct a relation over $P_1 \otimes Q$ and $P_2 \otimes Q$ and their derivative sets, and show that this is a system bisimulation. Let $P_i \overset{def}{=} \Sigma_i \underset{*}{\bowtie} \underline{\mathrm{init}}.Con_i$ for $i = 1, 2$ and $Q \overset{def}{=} \Sigma \underset{*}{\bowtie} \underline{\mathrm{init}}.Con$. Then by definition, $P_i \otimes Q \overset{def}{=} (\Sigma_i \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} \underline{\mathrm{init}}.(Con_i \underset{*}{\bowtie} Con)$ for $i = 1, 2$. By Proposition 7.1, these are well-defined and by Proposition 5.2, we know that

$$ds(P_i \otimes Q) = \{\langle (\Sigma_i \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (Con_i \underset{*}{\bowtie} Con), \tau_i\rangle, \langle (\Sigma_i \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} D_{i,1}, \sigma_{i,1}\rangle, \ldots, \langle (\Sigma_i \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} D_{i,n}, \sigma_{i,n}\rangle\}$$

where, for arbitrary state σ, $ds(\underline{\text{init}}.(Con_i \underset{*}{\bowtie} Con)) = \{\langle Con_i \underset{*}{\bowtie} Con, \sigma\rangle, \langle D_{i,1}, \sigma\rangle, \ldots, \langle D_{i,n}, \sigma\rangle\}$. Moreover, since $\underset{*}{\bowtie}$ is a static operator, each $D_{i,k}$ has the form $C_{i,k} \underset{*}{\bowtie} C_j$ for $C_{i,k} \in ds(\underline{\text{init}}.Con_i)$ and $C_j \in ds(\underline{\text{init}}.Con)$. We define the relation

$$\mathcal{R}' = \big\{\big((\Sigma_1 \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (C_{1,j_1} \underset{*}{\bowtie} C_j), (\Sigma_2 \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (C_{2,j_2} \underset{*}{\bowtie} C_j)\big) \mid \Sigma_1 \underset{*}{\bowtie} C_{1,j_1} \sim_s \Sigma_2 \underset{*}{\bowtie} C_{2,j_2}\big\}$$

and let $\mathcal{R} = \mathcal{R}' \cup \{(P_1 \otimes Q, P_2 \otimes Q)\}$. We will show that $\mathcal{R}$ is a system bisimulation.

Consider a transition $\langle(\Sigma_1 \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (C_{1,j_1} \underset{*}{\bowtie} C_j), \rho\rangle \xrightarrow{\underline{a}} \langle(\Sigma_1 \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (C_{1,k_1} \underset{*}{\bowtie} C_k), \rho'\rangle$. By our assumption that all events appear in the controller, we can conclude that $\underline{a}$ is a synchronised event in both the uncontrolled system and the controller. Thus we have 3 subcases: either the events come from Σ_1 and C_{1,j_1}, or from Σ and C_j, or from $\Sigma_1 \underset{*}{\bowtie} \Sigma$ and $C_{1,j_1} \underset{*}{\bowtie} C_j$. We will prove the last case, as the other two are simpler.

By shorter inferences in the derivation tree of the transition, we can conclude that $\langle\Sigma_1, \rho\rangle \xrightarrow{\underline{a}} \langle\Sigma_1, \mu\rangle$, $\langle\Sigma, \rho\rangle \xrightarrow{\underline{a}} \langle\Sigma, \nu\rangle$, $\langle C_{1,j_1}, \rho\rangle \xrightarrow{\underline{a}} \langle C_{1,k_1}, \rho\rangle$ and $\langle C_j, \rho\rangle \xrightarrow{\underline{a}} \langle C_k, \rho\rangle$, with $\rho' = \Gamma(\rho, \Gamma(\rho, \mu, \nu), \rho) = \Gamma(\rho, \mu, \nu)$ where ρ' is the state appearing in the target configuration of the original transition. Therefore $\langle\Sigma_1 \underset{*}{\bowtie} C_{1,j_1}, \rho\rangle \xrightarrow{\underline{a}} \langle\Sigma_1 \underset{*}{\bowtie} C_{1,k_1}, \mu\rangle$ since $\Gamma(\rho, \mu, \rho) = \mu$.

Since $\Sigma_1 \underset{*}{\bowtie} C_{1,j_1} \sim_s \Sigma_2 \underset{*}{\bowtie} C_{2,j_2}$, there is a transition $\langle\Sigma_1 \underset{*}{\bowtie} C_{2,j_2}, \rho\rangle \xrightarrow{\underline{a}} \langle\Sigma_1 \underset{*}{\bowtie} C_{2,k_2}, \mu\rangle$ such that $\Sigma_1 \underset{*}{\bowtie} C_{1,k_1} \sim_s \Sigma_2 \underset{*}{\bowtie} C_{2,k_2}$. By a shorter inference, $\langle\Sigma_1, \rho\rangle \xrightarrow{\underline{a}} \langle\Sigma_1, \mu\rangle$ and $\langle C_{2,j_2}, \rho\rangle \xrightarrow{\underline{a}} \langle C_{2,k_2}, \rho\rangle$. This allows us to infer the transition $\langle(\Sigma_2 \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (C_{2,2_1} \underset{*}{\bowtie} C_j), \rho\rangle \xrightarrow{\underline{a}} \langle(\Sigma_2 \underset{*}{\bowtie} \Sigma) \underset{*}{\bowtie} (C_{2,k_2} \underset{*}{\bowtie} C_k), \rho'\rangle$ as required. The symmetric condition is proved similarly. The same approach can be used to show that the pair $(P_1 \otimes Q, P_2 \otimes Q)$ are bisimilar by considering their $\underline{\text{init}}$ transitions (they have no others). $\square$

The following result shows that it is the prefixes which determine the behaviour of the controlled systems because of the restrictions imposed on well-defined controlled systems.

Theorem 8.3. Let $\Sigma_1 \underset{L}{\bowtie} \underline{\text{init}}.Con$ and $\Sigma_2 \underset{L}{\bowtie} \underline{\text{init}}.Con$ be two well-defined controlled systems. If $\mathsf{pr}(\Sigma_1) = \mathsf{pr}(\Sigma_2)$ then $\Sigma_1 \underset{L}{\bowtie} \underline{\text{init}}.Con \sim_s \Sigma_2 \underset{L}{\bowtie} \underline{\text{init}}.Con$

Proof. We will show that $\{(\Sigma_1, \Sigma_2)\}$ is a bisimulation. Since we are dealing with well-defined controlled systems, we know that each prefix occurs in a subcomponent of the form $S \overset{\text{def}}{=} \underline{a} : (\iota, r, I(\vec{X})).S + \ldots$ and this means that $\underline{a}$ can always happen. Additionally, for any other prefix starting with $\underline{a}$ appearing in a different subcomponent, these events will be synchronised on. Blocking due to a subcomponent being unable to perform the same action cannot occur because all events are able to occur. Hence the events that Σ_1 can perform are all the events that appear in a prefix in $\mathsf{pr}(\Sigma_1)$. Because of the form of S, we have that $\Sigma_1 \xrightarrow{\underline{a}} \Sigma_1$ for each possible event $\underline{a}$. This argument applies for each $\underline{a}$ in a prefix in $\mathsf{pr}(\Sigma_2)$. Since $\mathsf{pr}(\Sigma_1) = \mathsf{pr}(\Sigma_2)$, $\{(\Sigma_1, \Sigma_2)\}$ is a bisimulation and $\Sigma_1 \sim_s \Sigma_2$. Since $\sim_s$ is a congruence, we have the result. $\square$

This result allows us to tell whether two HYPE models are bisimilar by inspecting the prefixes in the model description to see if they are the same. Hence bisimulation can be checked syntactically. The next two results show that bisimilar HYPE models have the same ODEs.

Lemma 8.1. Let P and Q be well-defined controlled systems. If $P \sim_s Q$ then $\mathrm{st}(P) = \mathrm{st}(Q)$.

Proof. Consider $\langle P', \sigma'\rangle$ a derivative of $\langle P, \sigma\rangle$ then $\sigma' \in \mathrm{st}(P)$. Since $P \sim_s Q$, we can find $\langle Q', \sigma'\rangle$ with $\sigma' \in \mathrm{st}(Q)$. Hence $\mathrm{st}(P) \subseteq \mathrm{st}(Q)$ and vice versa. $\square$

Proposition 8.1. Let P and Q be well-defined controlled systems. If $P \sim_s Q$ then for every state $\sigma \in \mathrm{st}(P)$, $P_\sigma = Q_\sigma$.

Proof. The two well-defined controlled systems are $(P, \mathcal{V}, \mathcal{X}, \mathcal{E}, \mathcal{A}, ec, iv, pr, EC, ID)$ and $(Q, \mathcal{V}, \mathcal{X}, \mathcal{E}, \mathcal{A}, ec, iv, pr, EC, ID)$. Hence

$$P_\sigma = \left\{ \frac{dV}{dt} = \sum\big\{r[\![I(\vec{W})]\!] \;\big|\; iv(\iota) = V \text{ and } \sigma(\iota) = (r, I(\vec{W}))\big\} \;\Big|\; V \in \mathcal{V} \right\}$$

and

$$Q_\sigma = \left\{ \frac{dV}{dt} = \sum\big\{r[\![I(\vec{W})]\!] \;\big|\; iv(\iota) = V \text{ and } \sigma(\iota) = (r, I(\vec{W}))\big\} \;\Big|\; V \in \mathcal{V} \right\}$$

which are clearly the same. $\square$

The converse of Proposition 8.1 does not hold. It is possible for two states to be the same and hence give identical ODEs, but this does not mean that their associated derivatives are system bisimilar. We now go on to consider bisimulations that have been defined for other hybrid process algebras and those defined directly on hybrid systems.

8.1. Bisimulations for hybrid process algebras

Other process algebras for hybrid systems use a hybrid transition system with two types of transition: one type represents discrete events and the other continuous evolution of the system [Kha06]. By comparison, our transition system only has transitions for events. This gives a smaller transition system on which it is possible to consider simpler notions of equivalence.

Bergstra and Middelburg [BM05] present two bisimulations for their process algebra for hybrid systems ACP^{srt}_{hs}, defined over a hybrid transition system. One bisimulation fits with their axiomatic definition, and the other gives congruence with respect to the parallel operator, and was originally defined in [CR03]. We can recast these equivalences for our transition system and for our language, so that they are of interest here.

Definition 8.3 (Bergstra and Middelburg 2005). A *bisimulation* is a binary relation $B \subseteq (\mathcal{C} \times S) \times (\mathcal{C} \times S)$ such that for all $(\langle P, \sigma\rangle, \langle Q, \sigma\rangle) \in B$, for all $\underline{a} \in \mathcal{E}$, whenever

1. $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$, there exists $\langle Q', \sigma'\rangle$ such that $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$ and $(\langle P', \sigma'\rangle, \langle Q', \sigma'\rangle) \in B$, and
2. $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$, there exists $\langle P', \sigma'\rangle$ such that $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$ and $(\langle P', \sigma'\rangle, \langle Q', \sigma'\rangle) \in B$.

$\langle P, \sigma\rangle$ and $\langle Q, \sigma\rangle$ are *bisimilar*, written $\langle P, \sigma\rangle \leftrightarroweq \langle Q, \sigma\rangle$ if they are contained in a bisimulation. P and Q are *bisimilar*, written $P \leftrightarroweq Q$ if $\langle P, \sigma\rangle \leftrightarroweq \langle Q, \sigma\rangle$ for all states σ.

Definition 8.4 (Bergstra and Middelburg 2005). A *interference-compatible (ic-)bisimulation* is a binary relation $B \subseteq \mathcal{C} \times \mathcal{C}$ such that for all $(P, Q) \in B$, for all $\underline{a} \in \mathcal{E}$, for all $\sigma \in \mathcal{S}$ whenever

1. $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$, there exists $\langle Q', \sigma'\rangle$ such that $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$ and $(P', Q') \in B$, and
2. $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$, there exists $\langle P', \sigma'\rangle$ such that $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$ and $(P', Q') \in B$.

P and Q are *ic-bisimilar*, written $P \overset{\leftrightarrow}{\equiv} Q$ if they are contained in a ic-bisimulation.

It can be seen that $\overset{\leftrightarrow}{\equiv}$ is the same as $\sim_s$ since our system bisimulation definition also requires identical states. Both bisimulation and ic-bisimulation are congruences for the sequential operators of ACP^{srt}_{hs}; however, bisimulation is not a congruence for the parallel and merge operators. On the other hand, certain axioms are not sound for ic-bisimulation, hence the need for a complex two-level derivation system. In contrast, for HYPE, these two bisimulations identify the same models as we now show.

Theorem 8.4. For well-defined controlled systems P and Q, $P \leftrightarroweq Q$ implies $P \sim_s Q$.

Proof. Let $B = \{(P, Q) \mid P \leftrightarroweq Q\}$. Consider $(P, Q) \in B$ with $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$. Since $P \leftrightarroweq Q$, there exists $\langle Q', \sigma'\rangle$ such that $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$ and $\langle P', \sigma'\rangle \leftrightarroweq \langle Q', \sigma'\rangle$. If $P' \leftrightarroweq Q'$ then the proof is complete, so let $B' = \{(\langle P, \tau\rangle, \langle Q, \tau\rangle) \mid \langle P, \sigma\rangle \leftrightarroweq \langle Q, \sigma\rangle \text{for some } \sigma\}$. Consider arbitrary τ' with $\langle P', \tau'\rangle \xrightarrow{\underline{a}} \langle P'', \tau''\rangle$.

We can write $\tau'' = \tau'[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]$ for appropriate n and $\iota_j, r_j, I_j, 1 \leq j \leq n$. Let $\sigma'' = \sigma'[\iota_1 \mapsto (r_1, I_1)] \ldots [\iota_n \mapsto (r_n, I_n)]$. By Proposition 5.3, $\langle P', \sigma'\rangle \xrightarrow{\underline{a}} \langle P'', \sigma''\rangle$ and since $\langle P', \sigma'\rangle \leftrightarroweq \langle Q', \sigma'\rangle$, there exists $\langle Q'', \sigma''\rangle$ such that $\langle Q', \sigma'\rangle \xrightarrow{\underline{a}} \langle Q'', \sigma''\rangle$ and $\langle P'', \sigma''\rangle \leftrightarroweq \langle Q'', \sigma''\rangle$. Moreover $\langle Q', \tau'\rangle \xrightarrow{\underline{a}} \langle Q'', \tau''\rangle$ by Proposition 5.3 with $(\langle P'', \tau''\rangle, \langle Q'', \tau''\rangle) \in B'$. Hence $P' \leftrightarroweq Q'$ and $(P', Q') \in B$. The symmetric case is proved similarly. □

Theorem 8.5. For well-defined controlled systems P and Q, $P \overset{\leftrightarrow}{\equiv} Q$ implies $P \leftrightarroweq Q$.

Proof. Let $B = \{(\langle P, \sigma\rangle, \langle Q, \sigma\rangle) | P \overset{\leftrightarrow}{\equiv} Q\}$. Consider $(\langle P, \sigma\rangle, \langle Q, \sigma\rangle) \in B$ with $\langle P, \sigma\rangle \xrightarrow{\underline{a}} \langle P', \sigma'\rangle$. Since $P \overset{\leftrightarrow}{\equiv} Q$, there exists $\langle Q', \sigma'\rangle$ such that $\langle Q, \sigma\rangle \xrightarrow{\underline{a}} \langle Q', \sigma'\rangle$ and $P' \overset{\leftrightarrow}{\equiv} Q'$, hence $(\langle P', \sigma'\rangle, \langle Q', \sigma'\rangle) \in B$ and B is a bisimulation. Since σ is arbitrary, $P \leftrightarroweq Q$. □

The counter-example presented by Bergstra and Middelburg [BM05] to illustrate that congruence of bisimulation does not hold for a parallel operator uses the fact that in bisimulation, it is only required that target configurations are bisimilar for a specific state. This can be described as an initially stateless bisimilarity [Kha08, MRG05] since it is only for the initial pair that they must be able to match all transitions for any state. Hence it is possible to have two bisimilar processes, but when each is put in parallel with another process, the state in the target configuration can change, allowing previously blocked actions to become available. In contrast, under ic-bisimulation, target configurations are only ic-bisimilar if they have the same transitions under all possible states. This is described as stateless bisimilarity, because all pairs must be able to match transitions for any state. One can also consider a state-based bisimulation where pairs must be able match transitions for specific states only but this type of bisimulation is not considered here.

More specifically, one of the two processes involves the setting of the variable v to 1, followed by an a event prefixing deadlock, and the other has the same variable setting followed by event a prefixing a condition $v = 0$ that must be true for the event b to occur. These are equivalent under bisimulation since for any derived state, after the a event, the state contains the fact that v is associated with the value 1 and the b-event can never occur. However, when putting these two in parallel with a process that can change the value of v to zero, it is possible for the b event to occur in the second process but not the first. In the case of ic-bisimulation the two target states are not bisimilar since all possible states must be considered, including those where $v = 0$.

This cannot occur in HYPE because activation conditions (which determine if something can happen because a variable has the appropriate value) and resets (which change the values of variables) are disjoint from the syntactic description of the model, and the labelled transition system generated by the operational semantics does not take these into account. Thus, although bisimulation may only consider some states when applied to HYPE models, it is not possible for the same situation to arise.

8.2. Bisimulations on hybrid automata

The standard notion of bisimulation of hybrid automata is defined for a transition system encoding the dynamical evolution. Both continuous transitions and discrete transitions are used. These two relations are combined (usually interleaving discrete and continuous transitions) into one relation which defines the behaviour of the system. The hybrid automata bisimulation is defined on this relation in the usual way [Hen96, HTP05, DT07]. Most of the research into these bisimulations focusses on finding restrictions on hybrid automata syntax implying the existence of a finite bisimulation quotient thus guaranteeing decidability of reachability and of model checking, such as [LPS00].

Another form of bisimulation, the *U-bisimulation*, acts on the control graph, and it is used for projecting away some variables and for collapsing some modes of the automaton [AMP$^+$03]. In the following, we consider a simplified form of this U-bisimulation, where we retain all variables and collapse modes that have the same structure.

Definition 8.5. Let $\mathcal{H} = (V, E, \mathbf{X}, \mathcal{E}, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ be an hybrid automaton. Two modes v_1, v_2 are U-bisimilar, $v_1 \sim_{HA} v_2$, if and only if the following conditions hold:

1. $\mathit{flow}(v_1) = \mathit{flow}(v_2)$, $\mathit{inv}(v_1) = \mathit{inv}(v_2)$, and $\mathit{init}(v_1) = \mathit{init}(v_2)$;
2. for each edge $e_1 = (v_1, v_1')$ there exists an edge $e_2 = (v_2, v_2')$ such that $\mathit{event}(e_1) = \mathit{event}(e_2)$, $\mathit{reset}(e_1) = \mathit{reset}(e_2)$, $\mathit{jump}(e_1) = \mathit{jump}(e_2)$, $\mathit{urgent}(e_1) = \mathit{urgent}(e_2)$, and $v_1' \sim_{HA} v_2'$;
3. for each edge $e_2 = (v_2, v_2')$ there exists an edge $e_1 = (v_1, v_1')$ such that $\mathit{event}(e_1) = \mathit{event}(e_2)$, $\mathit{reset}(e_1) = \mathit{reset}(e_2)$, $\mathit{jump}(e_1) = \mathit{jump}(e_2)$, $\mathit{urgent}(e_1) = \mathit{urgent}(e_2)$, and $v_1' \sim_{HA} v_2'$;

This notion of bisimulation is related to system bisimulation. We have the following lemma.

Proposition 8.2. Let $P_1, P_2 \in \mathcal{C}_{Sys}$ such that $P_1 \sim_s P_2$. Then $\langle P_1, \sigma\rangle \sim_{HA} \langle P_2, \sigma\rangle$.

Proof. The first point of Definition 8.5 follows because we are considering the same state σ, assuming the initial conditions to be the same (they also depend just on σ). The next two points follow from the definition of $\sim_s$ and from the fact that the predicates *jump*, *event*, *urgent*, *reset* depend just on the event labelling the edge. □

Interestingly, the converse of the lemma does not hold. In fact, two states $\langle P, \sigma\rangle$ and $\langle Q, \tau\rangle$ of an hybrid automaton can be U-bisimilar even if $\sigma \neq \tau$: we only require the differential equations obtained from σ and τ to be the same. In fact, different states can lead to the same set of equations: consider two activities α and β mapped by iv to the same variable V. Then, if σ contains $(\alpha, 1, I)$ and $(\beta, -1, I)$ and τ contains $(\alpha, 2, I)$ and $(\beta, -2, I)$, then both σ and τ imply the equation $\dot{V} = 0$.

This means that the notion of bisimulation for hybrid automata is coarser than that for HYPE. Indeed, in a HYPE model we make explicit the source of each single flow of the system, while in a hybrid automaton flows are merged together in differential equations, and they cannot be separated in single influences. Stated otherwise, ODEs irremediably lose information about the logic of the system.

9. Modelling Style

In this section we discuss in more detail the modelling style of HYPE, comparing it with the modelling style of other hybrid process algebras and hybrid automata.

The main difference between HYPE and other hybrid process algebras is in the description of the continuous dynamics. Most of these process algebras describe the continuous dynamics by explicitly providing ODEs for each variable [BM05, CR05, RS03, vBMR$^+$06] and this is also the case for hybrid automata [Hen96]. In HYPE, we have adopted a different approach, modelling a set of influences, or flows, that affect the evolution of continuous variables, and then deriving the ODE by looking at the set of active influences in each mode. Discrete modes themselves are not modelled explicitly, but are derived from the operational semantics of the language. Since this represents a significant change in style from the existing process algebras, it was not feasible to develop HYPE as an extension of an existing hybrid process algebra. Instead we decided that HYPE must be developed afresh, centred on the idea of additive influences.

The advantage of this mechanism to describe continuous dynamics is particularly evident for systems in which there are many influences that can affect the dynamics of a specific variable, each of them having two or more possible forms. Consider again the orbiter example of Section 2 (see also Figure 1), and focus on the continuous dynamics of the temperature K. This variable is affected by four different flow sources: the heat of the sun, the heater, the effect of the shade and the thermodynamic cooling. The first three sources can be in two different states (i.e. present or absent), and their effect on the temperature depends on their state. Indeed, this gives rise to eight different differential equations for the temperature K. In languages that require the explicit provision of the ODEs for each variable, and that do not have a compositional mechanisms at the level of the continuous dynamics, like hybrid automata [Hen96] and ACP_{hs}^{srt} [BM05], the modeller is forced to write these eight ODEs explicitly. We can see this in Figure 13, where an ACP_{hs}^{srt} model for the orbiter is presented. This monolithic representation of the ODEs would be similar in any of the other existing hybrid process algebras. Notice that the number of possible ODEs grows exponentially with the number of flow sources that have more than one state (think of a tank with many inflow and outflow pipes). In HYPE, instead, we model the continuous dynamics of the orbiter temperature by associating an influence with each source, and changing the form of the influence in response to discrete events (i.e. activation and inactivation of the flow sources). This allows a much more compact description of the continuous dynamics: the complexity of the description grows linearly with the number of flow sources.

The use of additive flows also has the advantage that HYPE has more expressive power in relation to compositionality, as demonstrated by Theorem 7.1 and Corollary 7.1. In these results, it is shown that hybrid automata product is restricted to the case where there are no shared variables between the two hybrid automata (or flows are zero for all variables), whereas the product of HYPE models allows for shared variables. This means that it is possible to add new influences to an existing model that affect one or more of that model's variables. To achieve a similar effect in a hybrid automaton, the flow at some modes must be changed directly since it cannot be achieved using a product construction. In HYPE, addition of new levels can either be done at the model level using product, or as we demonstrate in the next section, by adding new subcomponents with new influences.

HYPE is also characterised by modularity in its specification style. Subcomponents are defined separately from controllers and the two have events on which they synchronise. Additionally, event conditions are specified separately from events, influence names from variables, and influence strength from influence type. In general, this means that a modeller can consider different concerns separately and we now highlight two examples of this modularity, and show the practical benefits it can bring. The reader can observe how the different states of the influences for the orbiter (see Figure 1) differ only in their influence strength, and not in their influence type. We found this to be a common pattern, and this motivated our separation of the influence strength from the influence type, i.e. the description of the functional form of the flow from its intensity. Moreover, as discussed below, we chose to combine influences into ODEs in an additive way.

Another modular feature of HYPE, is the strong separation that exists between the description of the continuous dynamics and the description of the discrete control. In particular, in HYPE one models the

$$
\begin{aligned}
T_{\mathbf{dFD}} &\overset{def}{=} (\dot{K} = -K \wedge K \geq t_2 \wedge K \leq t_3) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{light}}} \cdot T_{\mathbf{lFD}}\big)+ \\
&\quad \big((K \leq t_2) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{on}}} \cdot T_{\mathbf{dND}}\big)\big) + \big((K \geq t_3) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{up}}} \cdot T_{\mathbf{dNU}}\big)\big)\Big) \\
T_{\mathbf{lFD}} &\overset{def}{=} (\dot{K} = r_s - K \wedge K \geq t_2 \wedge K \leq t_3) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{dark}}} \cdot T_{\mathbf{dFD}}\big)+ \\
&\quad \big((K \leq t_2) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{on}}} \cdot T_{\mathbf{dND}}\big)\big) + \big((K \geq t_3) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{up}}} \cdot T_{\mathbf{dNU}}\big)\big)\Big) \\
T_{\mathbf{dND}} &\overset{def}{=} (\dot{K} = r_h - K \wedge K \leq t_1 \wedge K \leq t_3) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{light}}} \cdot T_{\mathbf{lND}}\big)+ \\
&\quad \big((K \geq t_1) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{off}}} \cdot T_{\mathbf{dFD}}\big)\big) + \big((K \geq t_3) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{up}}} \cdot T_{\mathbf{dNU}}\big)\big)\Big) \\
T_{\mathbf{lND}} &\overset{def}{=} (\dot{K} = r_s + r_h - K \wedge K \leq t_1 \wedge K \leq t_3) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{dark}}} \cdot T_{\mathbf{dND}}\big)+ \\
&\quad \big((K \geq t_1) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{off}}} \cdot T_{\mathbf{lFD}}\big)\big) + \big((K \geq t_3) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{up}}} \cdot T_{\mathbf{lNU}}\big)\big)\Big) \\
T_{\mathbf{dFU}} &\overset{def}{=} (\dot{K} = -r_d - K \wedge K \geq t_2 \wedge K \geq t_4) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{light}}} \cdot T_{\mathbf{lFU}}\big)+ \\
&\quad \big((K \leq t_2) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{on}}} \cdot T_{\mathbf{dNU}}\big)\big) + \big((K \leq t_4) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{down}}} \cdot T_{\mathbf{dND}}\big)\big)\Big) \\
T_{\mathbf{lFU}} &\overset{def}{=} (\dot{K} = r_s - r_d - K \wedge K \geq t_2 \wedge K \geq t_4) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{dark}}} \cdot T_{\mathbf{dFU}}\big)+ \\
&\quad \big((K \leq t_2) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{on}}} \cdot T_{\mathbf{dNU}}\big)\big) + \big((K \leq t_4) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{down}}} \cdot T_{\mathbf{dND}}\big)\big)\Big) \\
T_{\mathbf{dNU}} &\overset{def}{=} (\dot{K} = r_h - r_d - K \wedge K \leq t_1 \wedge K \geq t_4) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{light}}} \cdot T_{\mathbf{lNU}}\big)+ \\
&\quad \big((K \geq t_1) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{off}}} \cdot T_{\mathbf{dFU}}\big)\big) + \big((K \leq t_4) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{down}}} \cdot T_{\mathbf{dND}}\big)\big)\Big) \\
T_{\mathbf{lNU}} &\overset{def}{=} (\dot{K} = r_s + r_h - r_d - K \wedge K \leq t_1 \wedge K \geq t_4) \curvearrowright \sigma^{*}_{\mathsf{rel}}\Big(\big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{dark}}} \cdot T_{\mathbf{dNU}}\big)+ \\
&\quad \big((K \geq t_1) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{off}}} \cdot T_{\mathbf{lFU}}\big)\big) + \big((K \leq t_4) :\rightarrow \big((K^\bullet = {}^\bullet K) \Rsh \widetilde{\widetilde{\widetilde{down}}} \cdot T_{\mathbf{lND}}\big)\big)\Big)
\end{aligned}
$$

Fig. 13. The heater component expressed in ACP^{srt}_{hs} where the first element of the subscript indicates **l**ight or **d**ark, the second of**F** or o**N**, and the third **D**own or **U**p.

uncontrolled system and the controller, as distinct processes. The uncontrolled system is basically a flat description of all possible states of all flow sources, while the controller is used to impose an ordering on discrete events. The dependence of discrete events on continuous variables, via activation conditions and resets, is described separately from the controller. This allows the modeller to experiment with different combinations of discrete controllers, by either changing causality dependencies between events or by changing their dependencies on the continuous part of the system. Moreover, the choice of describing event conditions separately from the controlled system allows us to separate modelling concerns. In particular, we separate the description of the architecture of the model, i.e. the logical structure defining the possible interactions, from the description of the quantitative features of dynamics. This allows the modeller to experiment with different dynamical regimes without the necessity of modifying the structure of the model. We stress the fact that by combining the controller with activation conditions and resets, one can describe arbitrarily complex discrete dynamics.

Our choice to combine influences in an additive way when forming the ODEs, stems from the identification of influences with flow sources, which tend to be additive in nature (think of the inflows and outflows of a tank, or of the combined effect of different chemical reactions in a chemical system). We wish to stress, however, that this additive style is not a limitation. In fact, we can obtain any set of ODEs, with nonlinear terms as complex as desired. Indeed, this can be achieved by appropriate definition of the influence types. In the extreme case, one can model an ODE $\dot{V} = f(\mathcal{V})$ for a variable V by a single influence with strength 1 and influence type $I(\mathcal{V}) = f(\mathcal{V})$. However, whilst this is within the definition of the language, it is not inkeeping with the modelling spirit of HYPE, which emphasises modularity. It remains to remark that, in all the examples we have considered so far, additivity of flows has not been a limitation but we remain open to the possibility of studying different mechanisms to combine flows in the future.

In Section 4, we showed how to associate a hybrid automaton with each HYPE model. It is interesting to further consider the relationship between the class of models that can be described in HYPE and the class of models that can be described by hybrid automata (at least the class of hybrid automata that we consider in in Definition 4.1). It is also possible to construct a HYPE model that is equivalent to any given hybrid automaton[7]. The only element of hybrid automata that is not straightforwardly described in HYPE is the non-urgent discrete transition (with non-trivial activation conditions). However, these transitions can be modelled in HYPE by a sequence of events: an urgent event to fire as soon as the guard of the transition becomes active, followed by a non-deterministic event that is disallowed when the guard is false. We do not provide formal details of the construction of a HYPE model from a hybrid automata here. An example of it is given by the train-gate controller model of Section 10.

A modelling approach that is closely related to HYPE in the way of dealing with continuous dynamics is the (hybrid) bond graph approach [CBM08]. However, bond graphs are tailored to model physical systems, while HYPE has a more abstract approach, that supports the modelling of a large number of classes of systems showing hybrid dynamics (e.g. biochemical systems, ecological systems, financial markets, social networks, computer networks, and so on), even if it can be less efficient in specific domains than languages designed for these domains.

We close this section commenting on the notion of state in HYPE and in other hybrid languages. Usually, in hybrid languages a state describes the current mode and the current evaluation of continuous variables. In HYPE, instead, a state (operational state) is just a collection of active influences, hence it models the shape of the continuous dynamics, rather than a specific configuration of the system. This difference reflects on the notions of bisimulation. While in hybrid languages like ACP^{srt}_{hs} or hybrid automata, bisimulations predicate properties of the systems dynamics, in HYPE bisimulation compares two models in terms of their structure. In this sense, it is a stronger notion of bisimulation, as models with non-bisimilar structure can still exhibit a bisimilar dynamical behaviour. HYPE bisimulation, in particular, can be used to reason about the complexity of the description of a model and about the causal relationships between events, while bisimulation on the dynamics can be used to carry out analysis at that level, for instance reachability analysis. Further illustrating HYPE's more abstract style of modelling, as demonstrated in Section 8, is the fact that for HYPE, stateless bisimilarity is the same as initially stateless bisimilarity. This difference reflects also on the intrinsic computational complexity of these two notions: while bisimulation for finite HYPE models is (efficiently) computable, as it can be reduced to a bisimulation for labelled graphs [DPP04], bisimulations at the dynamical level are usually undecidable [HKPV95].

10. Example: train gate controller

We now consider the train gate system as described in [AHH96, BM05, Kha08]. We consider a section of railway track with a crossing gate. The track starts 1500m or more before the gate and ends 100m afterwards. There is a sensor at 1000m before the gate that sends a signal to the controller that a train is approaching, and a sensor at 100m after the gate that sends an *exit* signal to the controller. The train travels at between 48m/s and 52m/s and after the first sensor is passed, between 40m/s and 52m/s. The gate, on receipt of a *lower* signal, changes its angle to the ground from 90° to 0° at a rate of -20°/s until it is closed. Likewise, on the receipt of a *raise* signal, changes its angle to the ground from 0° to 90° at a rate of 20°/s until it is open. It must always respond to *lower* and *raise* signals. The controller on the receipt of an *approach* signal, sends a *lower* signal to the gate within 5 seconds, and on the receipt of an *exit* signal, sends a raise signal to the gate within 5 seconds, assuming it does not receive an *approach* signal within that time. It must always be available to receive an *approach* or *exit* signal. Once a train clears the section of track (passes the sensor after the gate), a subsequent train may arrive at the start of the section of track.

Initially, the train is at 1400m before the gate, and the gate is open. The safety property of the system in which we are interested, is whether the gate is down when the train is within a certain distance of the gate, say 100m. We will also consider a modification of the model, where the train is able to travel faster, with a reduction of speed closer to the gate, and consider whether the safety property still holds.

The HYPE model is given in Figure 14. Since HYPE does not directly support ranges of resets or ranges of influences, we will make decisions to simplify our modelling task. We will assume that subsequent trains appear at 1500m before the gate, that trains travel at their maximum speed of 52m/s and that the controller

[7] The notion of equivalence we have in mind here is the following: given a hybrid automaton, we associate with it a HYPE model. Then, we construct a new hybrid automaton associated with this HYPE model according to the construction of Section 4, and prove that it is bisimilar to the original one.

$$
\begin{array}{rcl}
\mathit{Gate} & \overset{\mathit{def}}{=} & \underline{\text{lower}} : (g, -r_{gt}, c).\mathit{Gate} + \underline{\text{closed}} : (g, 0, c).\mathit{Gate} + \\
 & & \underline{\text{raise}} : (g, -r_{gt}, c).\mathit{Gate} + \underline{\text{open}} : (g, 0, c).\mathit{Gate} + \\
 & & \underline{\text{init}} : (g, 0, c).\mathit{Gate}
\end{array}
$$

$$
\mathit{Train} \overset{\mathit{def}}{=} \underline{\text{init}} : (d, r_{tr}, c).\mathit{Train}
$$

$$
\begin{array}{rcl}
\mathit{Timer}_L & \overset{\mathit{def}}{=} & \underline{\text{appr}} : (t_L, 1, \mathit{const}).\mathit{Timer}_L + \underline{\text{lower}} : (t_L, 0, \mathit{const}).\mathit{Timer}_L + \underline{\text{init}} : (t_L, 0, \mathit{const}).\mathit{Timer}_L \\
\mathit{Timer}_R & \overset{\mathit{def}}{=} & \underline{\text{exit}} : (t_R, 1, \mathit{const}).\mathit{Timer}_R + \underline{\text{raise}} : (t_R, 0, \mathit{const}).\mathit{Timer}_R + \underline{\text{init}} : (t_R, 0, \mathit{const}).\mathit{Timer}_R
\end{array}
$$

$$
\begin{array}{rclrcl}
\mathit{Con}_a & \overset{\mathit{def}}{=} & \underline{\text{appr}}.\mathit{Con}_l + \underline{\text{exit}}.\mathit{Con}_a & \mathit{Con}_l & \overset{\mathit{def}}{=} & \underline{\text{appr}}.\mathit{Con}_l + \underline{\text{exit}}.\mathit{Con}_l + \underline{\text{lower}}.\mathit{Con}_e \\
\mathit{Con}_e & \overset{\mathit{def}}{=} & \underline{\text{appr}}.\mathit{Con}_e + \underline{\text{exit}}.\mathit{Con}_{ra} & \mathit{Con}_{ra} & \overset{\mathit{def}}{=} & \underline{\text{appr}}.\mathit{Con}_l + \underline{\text{exit}}.\mathit{Con}_{ra} + \underline{\text{raise}}.\mathit{Con}_a
\end{array}
$$

$$
\begin{array}{rclrcl}
GC_o & \overset{\mathit{def}}{=} & \underline{\text{raise}}.GC_o + \underline{\text{lower}}.GC_l & GC_l & \overset{\mathit{def}}{=} & \underline{\text{raise}}.GC_r + \underline{\text{lower}}.GC_l + \underline{\text{closed}}.GC_c \\
GC_c & \overset{\mathit{def}}{=} & \underline{\text{raise}}.GC_r + \underline{\text{lower}}.GC_c & GC_r & \overset{\mathit{def}}{=} & \underline{\text{raise}}.GC_r + \underline{\text{lower}}.GC_l + \underline{\text{open}}.GC_o
\end{array}
$$

$$
\mathit{Seq}_a \overset{\mathit{def}}{=} \underline{\text{appr}}.\mathit{Seq}_p \qquad \mathit{Seq}_p \overset{\mathit{def}}{=} \underline{\text{pass}}.\mathit{Seq}_e \qquad \mathit{Seq}_e \overset{\mathit{def}}{=} \underline{\text{exit}}.\mathit{Seq}_a
$$

$$
\begin{array}{rcl}
L & = & \{\underline{\text{lower}}, \underline{\text{raise}}, \underline{\text{appr}}, \underline{\text{pass}}, \underline{\text{exit}}, \underline{\text{init}}\} \\
\mathit{System} & \overset{\mathit{def}}{=} & (\mathit{Gate} \underset{\{\underline{\text{init}}\}}{\bowtie} \mathit{Train} \underset{\{\underline{\text{init}}\}}{\bowtie} \mathit{Timer}_L \underset{\{\underline{\text{init}}\}}{\bowtie} \mathit{Timer}_R) \underset{L}{\bowtie} \underline{\text{init}}.((\mathit{Con}_a \underset{\{\underline{\text{appr}},\underline{\text{exit}}\}}{\bowtie} \mathit{Seq}_a) \underset{\{\underline{\text{raise}},\underline{\text{lower}}\}}{\bowtie} GC_o)
\end{array}
$$

$$
iv(g) = G \qquad iv(d) = D \qquad iv(t_L) = T_L \qquad iv(t_R) = T_R \qquad [\![c]\!] = 1
$$

$$
ec(\underline{\text{init}}) = (\mathit{true}, G' = 90 \wedge D' = -1400 \wedge T_L' = 0 \wedge T_R' = 0)
$$

$$
\begin{array}{rclrcl}
ec(\underline{\text{lower}}) & = & (T_L = 5, T_L' = 0) & ec(\underline{\text{closed}}) & = & (G = 0, \mathit{true}) \\
ec(\underline{\text{raise}}) & = & (T_R = 5, T_R' = 0) & ec(\underline{\text{open}}) & = & (G = 90, \mathit{true})
\end{array}
$$

$$
\begin{array}{rclrcl}
ec(\underline{\text{appr}}) & = & (D = -1000, T_L' = 0) & r_{gt} & = & 20 \\
ec(\underline{\text{pass}}) & = & (D = 0, \mathit{true}) & r_{tr} & = & 52 \\
ec(\underline{\text{exit}}) & = & (D = 100, T_R' = 0 \wedge D' = -1500) & & &
\end{array}
$$

Fig. 14. HYPE model of train gate controller

will take the full 5 seconds to signal the gate, since these values represent the most negative conditions for the system. After our model description, we will demonstrate how to modify it to allow more flexibility.

In the model, there are 4 continuous components to represent the train, the gate, and two timers. There are also 3 distinct controlling or sequencing elements. Seq_a captures the fact that the train does not ever go backwards. Con_a is the controller of the system, and GC_o represents the discrete aspects of the gate itself. The gate is required to accept all $\underline{\text{raise}}$ and $\underline{\text{lower}}$ signals but it is reasonable to expect that the gate has a notion of internal state to determine what to do if a signal is received, and is aware when it is closed or open so that it can stop its own motors. All of these aspects given here are also present in the hybrid automata model given in [AHH96] but in our model continuous and discrete aspects are separated.

The model is well-defined and we wish to check that it is also well-behaved. Since the events of GC_o which do not appear in $\mathit{Con}_a \underset{*}{\bowtie} \mathit{Seq}_a$ do not activate any of those in in $\mathit{Con}_a \underset{*}{\bowtie} \mathit{Seq}_a$ and *vice versa*, we can apply Proposition 6.4. It is necessary to consider $\mathit{Con}_a \underset{*}{\bowtie} \mathit{Seq}_a$ as a single system since Con_a does allow infinite instantaneous behaviour. The event $\underline{\text{appr}}$ does not inhibit itself, and there are loops involving this event in the definition of Con_a. In the I-graph of Con_a, we obtain the loop $(\mathit{Con}_l, \underline{\text{appr}}, 10001) \to (\mathit{Con}_l, \underline{\text{appr}}, 10001)$ where the vector order is $\underline{\text{appr}}$, $\underline{\text{pass}}$, $\underline{\text{exit}}$, $\underline{\text{lower}}$, $\underline{\text{raise}}$.

The I-graph of GC_o consists of four directed acyclic graphs (when only considering those nodes that are reachable from nodes with vectors consisting of all ones), and the I-graph of $\mathit{Con}_a \underset{*}{\bowtie} \mathit{Seq}_a$ consists of 12 directed acyclic graphs, 6 of which are single nodes and 6 are two node graphs. Since these are both acyclic, we know that the overall controller is well-behaved by Proposition 6.4. There are 22 configurations

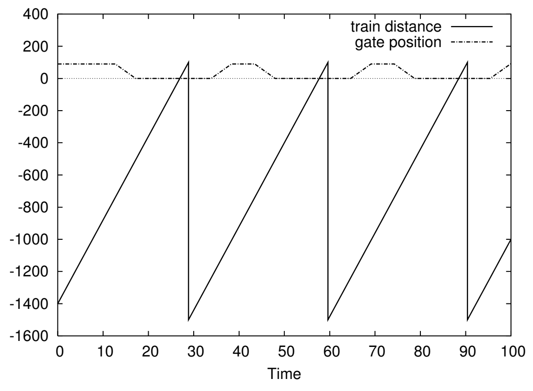


Fig. 15. Trace for the train gate system where $r_{tr} = 52$.

(excluding the initial configuration) in the labelled transition system of the model, hence giving a hybrid automaton with that number of modes.

When comparing our model to that of the hybrid automaton in [AHH96], our controller has an additional state reflecting a slightly more complex controller. If we construct the three-state controller, say Con' and put each controller in parallel with Seq_a, then we can show that $Con' \bowtie_* Seq_a$ and $Con_a \bowtie_* Seq_a$ are system bisimilar. From this we know by congruence, that $(Con' \bowtie_* Seq_a) \bowtie_* GC_o)$ and $(Con_a \bowtie_* Seq_a) \bowtie_* GC_o)$ are system bisimilar. Finally by Corollary 8.1, the two controlled systems are system bisimilar.

In terms of the safety property, we can run a single trace of the model (as given in Figure 15) since it is determinstic, and determine that the gate closes when the train is at 505.9m before the crossing agreeing with previous calculations [Kha08]. Another approach to considering the safety property is to construct a HYPE model with the same continuous components and a different controller that exactly describes how we want the system to behave with an additional event for failure. We address this below after we consider how to change the model.

To show the compositionality of our approach, we consider the scenario where the train is now able to travel at a much faster speed but the crossing gate is unchanged. To ensure that the gate is closed by the time the train is 100m from the crossing, it is necessary to slow the train as it approaches the crossing. We assume the train is required to slow at 1000m before the crossing.

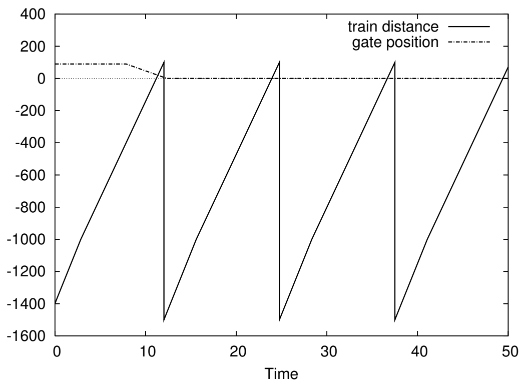
Since we already have an event at $D = -1000$, we do not need to introduce another event. We can simply add a new train component,

$$TrainRS \stackrel{\text{def}}{=} \underline{\text{appr}} : (d_s, -r_{sl}, c).TrainRS + \underline{\text{exit}} : (d_s, 0, c).TrainRS + \underline{\text{init}} : (d_s, 0, c).TrainRS$$

and let $iv(d_s) = D$. By adding this component, the train speed will be slowed at 1000m to $r_{tr} - r_{sl}$. In the model with the added component, we still have the same number of states in the labelled transition system because no new events have been added. The underlying hybrid automaton will be very similar to the previous one, except in certain states, the ODE for distance will differ slightly. In Figure 16, the graph for $r_{gt} = 140$ and $r_{sl} = 20$ is given on the left. We can see from the graph that the gate is not closed when the train passes the crossing and hence it cannot be closed 100m before the crossing. We wish to add to our model to capture the fact that this behaviour is not desirable.

We do this by adding an event called <u>fail</u> with event conditions $ec(\underline{\text{fail}}) = (-100 \leq D \wedge G \neq 0, true)$. Since this event can happen at anytime, we add the controller, $FC \stackrel{\text{def}}{=} \underline{\text{fail}}.0$. In the labelled transition system, this adds an additional configuration for each existing configuration (excluding the initial configuration). For each configuration $\langle \Sigma \bowtie_* D, \sigma \rangle$, there is a new configuration $\langle \Sigma \bowtie_* 0, \sigma \rangle$ with a new transition <u>fail</u> from the existing configuration to the new one. Thus the new hybrid automaton has 44 modes. This doubling of configurations and modes can be avoided by adding a <u>fail</u> event to each subcomponent and setting each influence to zero on the occurrence of the event. This means that there will be only one new state and there will be <u>fail</u> transitions from all existing states to the new state. The right graph in Figure 16 shows a trace of the gate system under the new speeds with the <u>fail</u> event being triggered.

By experimentation, we find that r_{sl}=50 leads to an execution without failure as shown in the left graph in Figure 17, however, this means that the gate is continually closed making the crossing unusable. By



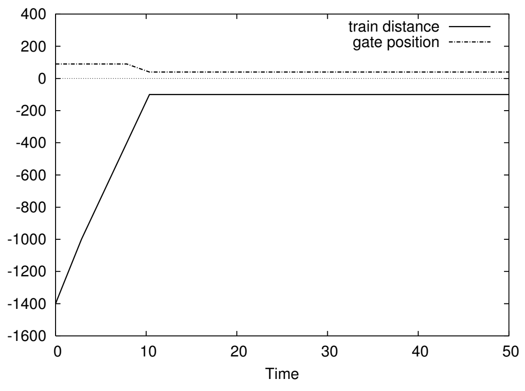
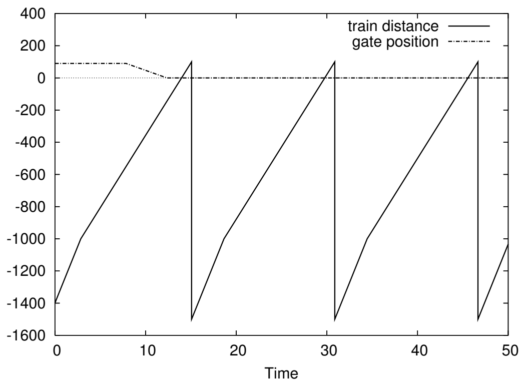


Fig. 16. Traces for the train gate system where $r_{tr} = 140$ and $r_{sl} = 20$. The model in the left trace does not deal explicitly with failure whereas the model in the right trace has an explicit failure event



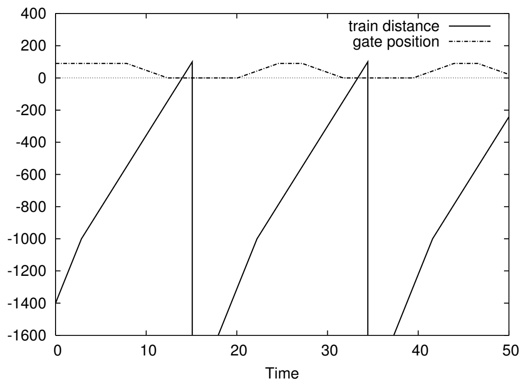


Fig. 17. Trace for the train gate system where $r_{tr} = 140$ and $r_{sl} = 50$ (left). Trace for the train gate system where $r_{tr} = 140$ and $r_{sl} = 50$, and originating distance is increased to 2000m

changing the gap between trains by increasing the originating distance to -2000m, we obtain a scenario where the behaviour of the gate is correct and the gate is opening between trains, as shown in the right graph in Figure 17.

Another approach to assessing the correctness of the model is to retain the continuous components of the system and use a controller that only allows the correct behaviour together with a failure component as before. We give the model in Figure 18. Using the underlying hybrid automaton which is smaller with only 10 modes, we obtain the same graphs as the last three using the same parameters.

We now consider aspects of the original scenario which we were not able to capture in HYPE. First, with

$$
\begin{array}{lcllcl}
C_1 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{appr}}.C_2 & C_6 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{appr}}.C_7 + \underline{\mathrm{raise}}.C_8 \\
C_2 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{lower}}.C_3 & C_7 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{lower}}.C_4 \\
C_3 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{close}}.C_4 & C_8 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{open}}.C_1 + \underline{\mathrm{appr}}.C_9 \\
C_4 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{pass}}.C_5 & C_9 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{open}}.C_2 + \underline{\mathrm{lower}}.C_3 \\
C_5 & \stackrel{\mathit{def}}{=} & \underline{\mathrm{exit}}.C_6 & & &
\end{array}
$$

$$
\begin{array}{rcl}
L & = & \{\underline{\mathrm{lower}}, \underline{\mathrm{raise}}, \underline{\mathrm{appr}}, \underline{\mathrm{pass}}, \underline{\mathrm{exit}}, \underline{\mathrm{init}}\} \\
\mathit{System} & \stackrel{\mathit{def}}{=} & (\mathit{Gate} \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Train} \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Timer}_L \underset{\{\underline{\mathrm{init}}\}}{\bowtie} \mathit{Timer}_R) \underset{L}{\bowtie} \underline{\mathrm{init}}.(C_1 \parallel \mathit{FC})
\end{array}
$$

Fig. 18. HYPE model of train gate controller with fail mode

respect to the originating distance being at least 1500m, we can use the $\perp$ activation condition in the event condition for $\underline{\text{exit}}$. This simulates a train starting from further away. We can take a similar approach for the 5 second limit for the signal to arrive from controller to gate. We can have one event $\underline{\text{lower}}_1$ which has an activation condition of $\perp$, and a second $\underline{\text{lower}}_2$ which has an activation condition of $T_L = 5$, which allows a $\underline{\text{lower}}$ event to happen at any point in time up to 5 seconds. Finally, we consider the speed range/non-determinism. There are a few ways to allow a choice from a discrete number of speeds from the range but none of them are ideal. This type of range can be also approached through experimentation on the model by varying parameters. Where non-determinism is used to reason about worst-case scenarios in the context of insufficient knowledge, the goal is to show that the system does not exhibit an unwanted behaviour, no matter what the parameters are (within their prescribed boundaries). In many cases, one can then identify extremal values for quantities of interest to use in the modelling process. For the train gate controller, we can do this by taking the fastest speed as the worst-case analysis, without loss of information and leading to a simpler model.

Thus this section has demonstrated how we can model a standard hybrid systems example in HYPE, as well as illustrating its compositionality both for adding additional components to the model, and adding failure events that depend on variables affected by different subcomponents.

11. Related Work

As mentioned in the introduction, HYPE takes a finer grained, less monolithic approach than the other process algebras for hybrid systems [BM05, vBMR$^+$06, RS03, CR05] because it enables the modelling of individual flows. In [Kha06] the comparison of these other process algebras is based on the train gate controller example and in each case, the train, gate and controller components have to be fully, sequentially described and then composed in parallel. This would also be necessary for the modelling of our orbiter example. For each of these process algebras, somewhere in the syntactic description of the system, a term such as $\dot{K} = r_s - r_d - K$, as well as terms for each of the other seven ODEs for the variable K, would need to appear to describe the continuous behaviour that can occur. By comparison, a modeller using HYPE would only need to model the individual flows, and not construct the ODEs explicitly. This could allow non-experts to model hybrid systems more easily.

A classical formalism for expressing hybrid systems is hybrid automata [Hen96]. As presented in Section 4 and Section 7, they are specified by defining explicitly both the control graph and the dynamical conditions within each mode, in terms of differential equations or, more generally, differential inclusions. Two hybrid automata are composed in parallel by synchronising transitions on shared events in the control graph [Hen96]. Flow conditions are combined by taking the logical conjunction of the predicates defining them. Where flows are defined by differential equations, the equation for each variable X must be defined only in one component, otherwise a logical inconsistency may arise. In Corollary 7.1, we showed the difference between the product of two hybrid automata and synchronisation of two HYPE models.

The modelling style of HYPE is quite different. Activities are identified with atomic flows acting on system variables. ODEs are then derived for an individual state by adding the different atomic flows acting on each variable. Activities can change in response to the happening of discrete events, which are controlled not by components but by an external controller and triggered by event conditions. This results in a separation of the description of the response of the system to events from the discrete control structure imposing causality on the happening of events. In contrast to hybrid automata, HYPE allows the separate description of flow conditions, event conditions, and the control graph, making easier the task of modifying the controller or the interactions with the environment. Furthermore, the fact that influence types are defined separately from the structure of the model also separates modelling concerns. Together, these features give a modularity of definition. In addition, compositionality of HYPE manifests on the set of activities (the state of the system) rather than on ODEs, hence we can allow different components of the system to influence the same continuous variable: the combined effect is obtained by superimposing flows, namely by addition on the right hand side of ODEs. Since a HYPE model can be expressed as a hybrid automaton, when using HYPE to model one gets the advantages of HYPE together with the formalism of hybrid automata. This is a distinct advantage over languages such as CHARON [AGLS06], SHIFT [DGV96], and HyCharts [GS02]. These are all compositional formalisms describing hybrid systems which do not map so readily to hybrid automata. They differ from HYPE in that they do not have the simple syntax and structured operational semantics of a process algebra.

Another formalism which has a mechanism to combine flows is hybrid action systems [RRS03]. This language is based on Dijkstra's guarded command language and has predicate transformer semantics. Differential actions consist of guards before ODEs, and parallel composition of these actions is defined as a linear combination of functions with the addition of functions over any shared domains. This differs from our approach where we associate influence names with a specific variable and then sum over all influences for a given variable to obtain the ODE.

Physical systems can also be modelled by constitutive equations and bond graphs [Pay61]. This is a modelling technique in which a system's components are described by means of equations relating main physical quantities of interest. Components are then glued together by imposing suitable conservation laws. This results in a set of differential equations with algebraic constraints, which after an algebraic manipulation, can be simplified by removing variables and constraints. There are also hybrid extensions of the bond graph method which have been represented in a hybrid process algebra [CBM08], to deal with discontinuities in physical systems (such as a bouncing ball).

In HYPE, instead, the modelling activity concentrates around the notion of flow or influence, which is not explicitly connected with physical quantities or with conservation laws, and components are described by specifying the way they react to external events through flows modifications. This may be less natural for certain physical systems, as one has to identify the different influences acting on each variable of interest (without relying on the implicit derivation mechanism provided by conservation laws). However, HYPE's modelling style is straightforwardly applicable to a wider class of systems, at different levels of abstraction.

12. Conclusions and Further Work

In this article, we have presented HYPE, a modelling formalism for hybrid systems, developed its theory and illustrated its applicability. HYPE has novel features that include a fine-grained approach to modelling flows and an explicit controller.

HYPE, in the usual process algebra style, has a syntax consisting of a small number of operators and an operational semantics that describe the behaviour of a HYPE controlled system. This results in a labelled transition system over configurations which are pairs of controlled system derivatives and states. States contain information about the continuous flows that are in operation at a configuration in the labelled transition system, and a transition represents a change to a configuration where different continuous flows apply. This provides a straightforward mapping to a hybrid automaton, where information about the flows in operation are used to construct the ODEs at a mode in the hybrid automaton.

We considered the general behaviour of HYPE models and identified syntactic restrictions called well-definedness and operational restrictions called well-behavedness, to ensure HYPE models have sensible behaviour. We showed for both of these restrictions that they are semantically meaningful in the behaviour they ensure. Moreover, particularly in the case of well-behavedness, the I-graph of the controller with event conditions allows us to reason about the behaviour of a HYPE model without needing to explore it in full detail.

Since hybrid automata are a classical formalism for hybrid systems and also because they provide the hybrid semantics for HYPE models, we considered the expressive differences between products in the two formalisms, showing that HYPE models, due to the use of influences, allow for more expressive and flexible products since the same variable can appear in both models that form the product.

We investigated equivalence semantics and provided a definition of bisimulation over HYPE models together with results about congruence and the ODEs obtained from bisimilar HYPE models. Finally, we presented a classical case study and demonstrated how HYPE can successfully model the problem, as well as an extension of the problem showing the value of compositionality.

In terms of further work, additional modelling using HYPE is important. To date, we have modelled a dual-tank system (as described by [TCT01]), a bottling line (as described by [BM05]), the abstract view of the repressilator [GHB08] (as described by [BP08]) and the circadian clock of a green alga [Gal10] (as described by [AGLT10]).

Since the process algebraic approach uses a language to describe systems, we wish to further explore how reasoning at the language level can provide results about the underlying model. A specific area of focus is that of equivalences that allow us to capture notions of similar behaviour in different systems. Here we could consider bisimulations for models with very different tuples with appropriate maps between variables, events, activities, functions and sets. In the first case these maps would be bijections, but it may be possible to explore surjective maps. We also wish to investigate links between bisimulation and I-graphs and to obtain results about the well-behavedness of a synchronisation of two HYPE models using the theory developed in Section 6 through constructing an I-graph for the controller $Con_1 \bowtie_{*} Con_2$ and the new event conditions ec defined from ec_1 and ec_2.

We have also extended HYPE models with stochastic durations for events [BGH10a] where the underlying semantic model is Transition-Driven Hybrid Stochastic Automata [BP09], a subset of Piecewise Deterministic Markov Processes [Dav93], and used this extension to model delay-tolerant networks [GHB10, BGH10b]. This is also a direction for future research.

Acknowledgements This research is supported by Royal Society International Joint Project JP090562. Vashti Galpin has been supported by the EPSRC SIGNAL Project, Grant EP/E031439/1. Jane Hillston has been supported by the EPSRC under ARF EP/c543696/01. We thank the reviewers for their comments.

References

[AGLS06] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional modeling and refinement for hierarchical hybrid systems. *Journal of Logic and Algebraic Programming*, 68:105–128, 2006.

[AGLT10] O.E. Akman, M.L. Guerriero, L. Loewe, and C. Troein. Complementary approaches to understanding the plant circadian clock. In *Proceedings of FBTC'10*, EPTCS, 19:1-19, 2010.

[AHH96] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181 –201, 1996.

[AMP$^+$03] M. Antoniotti, B. Mishra, C. Piazza, A. Policriti, and M. Simeoni. Modeling cellular behavior with hybrid automata: Bisimulation and collapsing. In C. Priami, editor, *Proceedings of CMSB 2003*, LNCS 2602, pages 57–74, 2003.

[BGH10a] L. Bortolussi, V. Galpin, and J. Hillston. HYPE with stochastic events, 2010. Submitted for publication.

[BGH10b] L. Bortolussi, V. Galpin, and J. Hillston. Modeling hybrid systems with stochastic events in HYPE. In *Proceedings of the 9th Workshop on Process Algebra and Stochastically Timed Activities (PASTA)*, pages 24–28, 2010.

[BM05] J.A. Bergstra and C.A. Middelburg. Process algebra for hybrid systems. *Theoretical Computer Science*, 335:215–280, 2005.

[BP08] L. Bortolusssi and A. Policriti. Hybrid approximation of stochastic process algebras for systems biology. In *IFAC World Congress,* Seoul, South Korea, July 2008.

[BP09] L. Bortolussi and A. Policriti. Hybrid semantics of stochastic programs with dynamic reconfiguration. In *Proceedings of COMPMOD 2009*, EPTCS, 6:63-76, 2009.

[CBM08] P.J.L. Cuijpers, J.F. Broenink, and P.J. Mosterman. Constitutive hybrid processes: a process-algebraic semantics for hybrid bond graphs. *SIMULATION*, 8:339–358, 2008.

[CR03] P.J.L. Cuijpers and M.A. Reniers. Hybrid process algebra. Computer Science Reports CSR 03-07, Department of Computer Science, Eindhoven Technical University, 2003.

[CR05] P.J.L. Cuijpers and M.A. Reniers. Hybrid process algebra. *Journal of Logic and Algebraic Programming*, 62:191–245, 2005.

[Dav93] M.H.A. Davis. *Markov Models and Optimization.* Chapman & Hall, 1993.

[DGV96] A. Deshpande, A. Göllü, and P. Varaiya. SHIFT: A formalism and a programming language for dynamic networks of hybrid automata. In P.J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Proceedings of Hybrid Systems IV*, LNCS 1273, pages 113–133, 1996.

[DPP04] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation. *Theoretical Computer Science*, pages 221–256, 2004.

[DT07] J.M. Davoren and P. Tabuada. On simulations and bisimulations of general flow systems. In A. Bemporad, A. Bicchi, and G.C. Buttazzo, editors, *Proceedings of HSCC 2007*, LNCS 4416, pages 145–158, 2007.

[EL00] M.B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, 2000.

[Gal10] V. Galpin. Modelling a circadian clock with HYPE. In *Proceedings of the 9th Workshop on Process Algebra and Stochastically Timed Activities (PASTA)*, pages 92–98, 2010.

[GBH09] V. Galpin, L. Bortolussi, and J. Hillston. Hype: a process algebra for compositional flows and emergent behaviour. In M. Bravetti and G. Zavattaro, editors, *Proceedings of CONCUR 2009*, LNCS 5710, pages 305–320. Springer, 2009.

[GHB08] V. Galpin, J. Hillston, and L. Bortolussi. HYPE applied to the modelling of hybrid biological systems. *Electronic Notes in Theoretical Computer Science*, 218:33–51, 2008.

[GHB10] V. Galpin, J. Hillston, and L. Bortolussi. A stochastic hybrid process algebra (poster). In *Models and Logics for Quantitative Analysis (MLQA 2010)*, Edinburgh, July, 2010.

[GS02] R. Grosu and T. Stauner. Modular and visual specification of hybrid systems: An introduction to HyCharts. *Formal Methods in System Design*, 21:5–38, 2002.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, 1996.

[HH94] T.A. Henzinger and P.-H. Ho. HYTECH: The Cornell HYbrid TECHnology Tool. In P.J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems*, LNCS 999, pages 265–293. Springer, 1994.

[Hil05] J. Hillston. Fluid flow approximation of PEPA models. In *Second International Conference on the Quantitative Evaluation of Systems (QEST 2005)*, pages 33–43. IEEE Computer Society, 2005.

[HKPV95] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing*, STOC '95, pages 373–382, 1995.

[HTP05] E. Haghverdi, P. Tabuada, and G.J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342:229–261, 2005.

[Kha06] U. Khadim. A comparative study of process algebras for hybrid systems. Computer Science Report CSR 06-23, Technische Universiteit Eindhoven, 2006. `http://alexandria.tue.nl/extra1/wskrap/publichtml/200623.pdf`.

[Kha08] U. Khadim. *Process algebras for hybrid systems: Comparison and development.* PhD thesis, IPA, Technische Universiteit Eindhoven, 2008.

[LPS00] G. Lafferriere, G.J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13:1–21, 2000.

[Mil89] R. Milner. *Communication and concurrency.* Prentice Hall, 1989.

[MRG05] M.R. Mousavi, M.A. Reniers, and J.F. Groote. Notions of bisimulation and congruence formats for sos with data. *Information and Computation*, 200:107–147, 2005.

[Pay61] H.M. Paynter. *Analysis and Design of Engineering Systems.* MIT Press, 1961.

[RRS03] M. Rönkkö, A.P. Ravn, and K. Sere. Hybrid action systems. *Theoretical Computer Science*, 290:937–973, 2003.

[RS03] W.C. Rounds and H. Song. The ϕ-calculus: A language for distributed control of reconfigurable embedded systems. In O. Maler and A. Pnueli, editors, *Proceedings of HSCC 2003*, LNCS 2623, pages 435–449, 2003.

[TCT01] B. Tuffin, D.S. Chen, and K.S. Trivedi. Comparison of hybrid systems and fluid stochastic Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 11:77–95, 2001.

[TGH10] M. Tribastone, S. Gilmore, and J. Hillston. Scalable differential analysis of process algebra models. *IEEE Transactions on Software Engineering*, 2010. to appear.

[vBMR$^+$06] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Syntax and consistent equation semantics of hybrid χ. *Journal of Logic and Algebraic Programming*, 68:129–210, 2006.

[vG90] R.J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *Proceedings of CONCUR 90*, LNCS 458, pages 278–297. Springer, 1990.

A. Proof of Theorem 7.1

Theorem. Let P_1 and P_2 be well-defined HYPE systems $(P_i, \mathcal{V}_i, IN_i, IT_i, \mathcal{E}_i, \mathcal{A}_i, ec_i, iv_i, EC_i, ID_i)$ for $i = 1, 2$. If $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ then $\mathcal{H}(P_1 \otimes P_2) = \mathcal{H}(P_1) \times \mathcal{H}(P_2)$ when vertices unreachable from the initial vertex are excluded.

Proof. To show that $\mathcal{H}(P_1 \otimes P_2) = \mathcal{H}(P_1) \times \mathcal{H}(P_2)$ we need to show that each component in the two hybrid automata are the same. For the functions that map from vertices and those that map from edges, it is sufficient to show that they map to the same values.

First, consider $P_1 \otimes P_2$ which is the tuple consisting of $(P \stackrel{def}{=} (\Sigma_1 \bowtie_* \Sigma_2) \bowtie_* \underline{\text{init}}.(Con_1 \bowtie_* Con_2)$, $\mathcal{V}_1 \cup \mathcal{V}_2$, $IN_1 \cup IN_2$, $IT_1 \cup IT_2$, $\mathcal{E}_1 \cup \mathcal{E}_2$, $\mathcal{A}_1 \cup \mathcal{A}_2$, ec, $iv_1 \cup iv_2$, $EC_1 \cup EC_2$, $ID_1 \cup ID_2)$ with the conditions and ec given by the definition. We can then construct the labelled transition for this product. Then the hybrid automaton we obtain from the product of the two HYPE models is defined as $H = (V, E, \mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{E}_1 \cup \mathcal{E}_2, \mathit{flow}, \mathit{init}, \mathit{inv}, \mathit{event}, \mathit{jump}, \mathit{reset}, \mathit{urgent})$ where

- $V = ds((\Sigma_1 \bowtie_* \Sigma_2) \bowtie_* \underline{\text{init}}.(Con_1 \bowtie_* Con_2))$
- E contains (v, v') if $v = \langle P, \sigma \rangle$ and $v' = \langle P', \sigma' \rangle$ for some derivation $\langle P, \sigma \rangle \xrightarrow{\underline{a}} \langle P', \sigma' \rangle$
- for $v_j = \langle P_j, \sigma_j \rangle$ then $flow(v_j)[X] = \sum\{r[\![I(\mathcal{W})]\!] \mid iv(\iota) = X \text{ and } \sigma_j(\iota) = (r, I(\mathcal{W}))\}$
- $init(v) = \begin{cases} res(\underline{\text{init}}), & \text{if } v = \langle P, \sigma \rangle \\ \mathit{false}, & \mathit{otherwise} \end{cases}$ with primes removed from variables
- $inv(v) = true$.
- Let $e = (\langle P, \sigma \rangle, \langle P', \sigma' \rangle)$ with $\langle P, \sigma \rangle \xrightarrow{\underline{a}} \langle P', \sigma' \rangle$. Then $event(e) = \underline{a}$ and $reset(e) = res(\underline{a})$. Moreover, if $act(\underline{a}) \neq \bot$, then $jump(e) = act(\underline{a})$ and $urgent(e) = true$, otherwise $jump(e) = true$ and $urgent(e) = false$.

Next consider the hybrid automata of P_1 and P_2. For $i = 1, 2$ we have $H_i = (V_i, E_i, \mathcal{V}_i, \mathcal{E}_i, \mathit{flow}_i, \mathit{init}_i, \mathit{event}_i, \mathit{jump}_i, \mathit{reset}_i, \mathit{urgent}_i)$ where

- $V_i = ds(\Sigma_i \bowtie_* \underline{\text{init}}.Con_i)$
- E_i contains (v, v') if $v_1 = \langle P, \sigma \rangle$ and $v' = \langle P', \sigma' \rangle$ for some derivation $\langle P, \sigma \rangle \xrightarrow{\underline{a}} \langle P', \sigma' \rangle$

flow_i, init_i, inv_i, event_i, reset_i, jump_i and urgent_i are defined in the expected manner. The synchronised product $\mathcal{H}(P_1) \times \mathcal{H}(P_2)$ is $H' = (V_1 \times V_2,\ E',\ \mathcal{V}_1 \cup \mathcal{V}_2,\ \mathcal{E}_1 \cup \mathcal{E}_2,\ \mathit{flow}',\ \mathit{init}',\ \mathit{event}',\ \mathit{jump}',\ \mathit{reset}',\ \mathit{urgent}')$ following the synchronised product definition.

Immediately, we can see that the events and the variables are the same in H and H'. Next we consider the vertices and edges of the graphs that make up the hybrid automata and define a function between these, and show that the function is a bijection, hence we have a graph isomorphism which is sufficient to conclude that the graphs are the same.

Let $W \subseteq V_1 \times V_2$ be the subset of vertices we consider. These are the vertices which are reachable from the initial configuration $(\langle \Sigma_1 \underset{*}{\bowtie} \mathit{Con}_1, \tau_1\rangle, \langle \Sigma_2 \underset{*}{\bowtie} \mathit{Con}_2, \tau_2\rangle)$ (τ_1 and τ_2 are the states immediately after $\underline{\mathrm{init}}$).

We define $f : W \to V$ by

$$f\big((\langle \Sigma_1 \underset{*}{\bowtie} D_1, \sigma_1\rangle, \langle \Sigma_2 \underset{*}{\bowtie} D_2, \sigma_2\rangle)\big) = \langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1 \underset{*}{\bowtie} D_2), \sigma_1 \cup \sigma_2\rangle$$

where $D_i \in \mathit{ds}(\underline{\mathrm{init}}.\mathit{Con}_i)$. Furthermore, for an edge (w, w'), $f(w, w') = (f(w), f(w'))$.

First, we show that if (w, w') is an edge in E' then $f(w, w')$ is an edge in E.

$$\begin{aligned}(w, w') &= \big((v_1, v_1'), (v_2, v_2')\big)\\ &= \big((\langle \Sigma_1 \underset{*}{\bowtie} D_1, \sigma_1\rangle, \langle \Sigma_2 \underset{*}{\bowtie} D_2, \sigma_2\rangle), (\langle \Sigma_1 \underset{*}{\bowtie} D_1', \sigma_1'\rangle, \langle \Sigma_2 \underset{*}{\bowtie} D_2', \sigma_2'\rangle)\big)\end{aligned}$$

$$\begin{aligned}f(w, w') &= \big(f(w), f(w')\big)\\ &= \big(\langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1 \underset{*}{\bowtie} D_2), \sigma_1 \cup \sigma_2\rangle, \langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1' \underset{*}{\bowtie} D_2'), \sigma_1' \cup \sigma_2'\rangle\big)\end{aligned}$$

There are three ways in which an edge can be obtained in a hybrid automata product, and we consider the most complex case only. The other two cases are similar. We consider the case where $\mathit{event}((w, w')) \in \mathcal{E}_1 \cap \mathcal{E}_2$. We can infer that we have two edges $(v_i, v_i') \in E_1$, $i = 1, 2$ with the same event label as (w, w'). Hence, we have transitions of the form $\langle \Sigma_i \underset{*}{\bowtie} D_i, \sigma_i\rangle \xrightarrow{\underline{\mathrm{a}}} \langle \Sigma_i \underset{*}{\bowtie} D_i', \sigma_i'\rangle$. By manipulation of derivation trees and well-definedness, $\langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1 \underset{*}{\bowtie} D_2), \sigma_1 \cup \sigma_2\rangle \xrightarrow{\underline{\mathrm{a}}} \langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1' \underset{*}{\bowtie} D_2'), \sigma_1' \cup \sigma_2'\rangle$. Since IN_1 and IN_2 are disjoint, we can start with $\sigma_1 \cup \sigma_2$ and as a result of the derivation, we will obtain $\sigma_1' \cup \sigma_2'$. We can reverse this argument to show that if $(f(w), f(w'))$ is an edge in E then (w, w') is an edge in E'.

The preceding paragraph implicitly assumes that f is well-defined but it must be proved. For $w \in W$, we show that $f(w) \in V$. Since w is reachable from the initial vertex w_0, there is a sequence of vertices $w_0, w_1, \ldots, w_n, w$ that describe the edges from w_0 to w. Hence, there is a sequence of vertices $f(w_0), f(w_1), \ldots, f(w_n), f(w)$. We need to ensure that the initial vertices have the same states and this is guaranteed for well-defined systems by Proposition 5.5.

Next, we need to show that f is a bijection. For $v \in V$, we can use Proposition 5.2 to show that there exists $w \in W$ such that $f(w) = v$. We have indicated above that given an edge $(f(w), f(w')) \in E$ there exist $(w, w') \in E'$. Hence f is surjective. If we have two vertices in W that only differ in one of their states, then the resulting vertices in V will differ, and if they differ in terms of their controller components, then the vertices in V will also differ. If we have two edges in E', then their images will also differ. Hence f is injective and we have a graph isomorphism.

We need to check the functions on vertices. Clearly, $\mathit{inv}'(w) = \mathit{inv}(f(w))$ since both inv' and inv map to true. $\mathit{init}'(w) = \mathit{false}$ for all $w \in W$ except $w' = (\langle \Sigma_1 \underset{*}{\bowtie} \mathit{Con}_1, \tau_1\rangle, \langle \Sigma_2 \underset{*}{\bowtie} \mathit{Con}_2, \tau_2\rangle)$ and $f(w') = \langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (\mathit{Con}_1 \underset{*}{\bowtie} \mathit{Con}_2), \tau\rangle$. $\mathit{init}((f(w)) = \mathit{true}$ and $\mathit{init}(v) = \mathit{false}$ for all other vertices $v \in V$.

We also need to consider flow. For $(v_1, v_2) \in W$, $\mathit{flow}'((v, v)) = \mathit{flow}_1(v_1) \wedge \mathit{flow}_2(v_2)$. Since $V_1 \cap V_2 = \emptyset$, each $\mathit{flow}_i(v_i)$ contributes distinct ODEs on distinct variables to flow' and does not evaluate to false for any variable. Consider $\mathit{flow}'(w)[X] = \sum\{r[\![I(\mathcal{W})]\!] \mid \mathit{iv}_1(\iota) = X \text{ and } \sigma_1(\iota) = (r, I(\mathcal{W}))\}$ where $X \in \mathcal{V}_1$ and $w = (\langle \Sigma_1 \underset{*}{\bowtie} D_1, \sigma_1\rangle, \langle \Sigma_2 \underset{*}{\bowtie} D_2, \sigma_2\rangle)$. We know that $f(w) = \langle (\Sigma_1 \underset{*}{\bowtie} \Sigma_2) \underset{*}{\bowtie} (D_1 \underset{*}{\bowtie} D_2), \sigma_1 \cup \sigma_2\rangle$ hence $\mathit{flow}(f(w))[X] = \sum\{r[\![I(\mathcal{W})]\!] \mid (\mathit{iv}_1 \cup \mathit{iv}_2)(\iota) = X \text{ and } (\sigma_1 \cup \sigma_2)(\iota) = (r, I(\mathcal{W}))\}$. Any ι that satisfies $(\mathit{iv}_1 \cup \mathit{iv}_2)(\iota) = X$ must be in IN_1 since $X \in \mathcal{V}_1$, therefore the ODE obtained is the same in both cases because we only need consider σ_1. An similar argument can be made for $X \in \mathcal{V}_2$.

Next, we need to check the functions on edges. Let $e = (e_1, e_2)$ be an edge in E'. From above, $\mathit{event}'(e) = \mathit{event}(f(e)) = \underline{\mathrm{a}}$. For jump, we need to consider if $\mathit{event}'(e) = \underline{\mathrm{a}}$ is a shared event or not.

- If $\underline{\mathrm{a}} \in \mathcal{E}_1 \backslash \mathcal{E}_2$, then there are two cases.
  - If $\mathit{act}_1(\underline{\mathrm{a}}) \neq \bot$ then $\mathit{jump}'((e_1, e_2)) = \mathit{jump}_1(e_1) = \mathit{act}_1(\underline{\mathrm{a}})$ and $\mathit{urgent}'(e) = \mathit{urgent}_1(e_1) = \mathit{true}$. For H, $\mathit{jump}(f(e)) = \mathit{act}(\underline{\mathrm{a}}) = \mathit{act}_1(\underline{\mathrm{a}})$ and $\mathit{urgent}(f(e)) = \mathit{true}$ since $\mathit{act}(\underline{\mathrm{a}}) \neq \bot$.

  - If $act_1(\underline{a}) = \perp$ then $jump'(e) = jump_1(e_1) = true$ and $urgent'(e) = urgent_1(e_1) = false$. For H, $jump(f(e)) = true$ and $urgent(f(e)) = false$.

  If $\underline{a} \in \mathcal{E}_2 \backslash \mathcal{E}_1$, the argument is similar.
- If $\underline{a} \in \mathcal{E}_1 \cap \mathcal{E}_2$ then there are again two cases.
  - If $act_1(\underline{a}) \wedge act_2(\underline{a}) \neq \perp$, we have $jump'((e_1, e_2)) = jump_1(e_1) \wedge jump_2(e_2) = act_1(\underline{a}) \wedge act_2(\underline{a})$ and $urgent'((e_1, e_2)) = true$ Likewise $jump(f(e)) = act(\underline{a}) = act_1(\underline{a}) \wedge act_2(\underline{a})$ and $urgent(f(e)) = true$ since $act_1(\underline{a}) \wedge act_2(\underline{a}) \neq \perp$.
  - Otherwise, $jump'((e_1, e_2)) = jump_1(e_1) \wedge jump_2(e_2) = true$ and $urgent'((e_1, e_2)) = urgent_1(e_1) \vee urgent_2(e_2) = false$, since $act_1(\underline{a}) \wedge act_2(\underline{a}) = \perp$ implies $urgent_1(e_1) = false$ and $urgent_2(e_2) = false$. Likewise, $jump(f(e)) = true$ and $urgent'(f(e)) = false$.

In the case of *reset* and *reset'*, we need to again consider if we have a shared event or not.

- If $event(e) = \underline{a} \in \mathcal{E}_1 \backslash \mathcal{E}_2$ then $reset'(e) = reset_1(e_1) = res_1(\underline{a})$ and $reset(f(e)) = res(\underline{a}) = res_1(\underline{a})$.
- If $event\,\underline{a} \in \mathcal{E}_1 \cap \mathcal{E}_2$ then $reset'(e) = reset_1(e_1) \wedge reset_2(e_2) = res_1(\underline{a}) \wedge res_2(\underline{a})$. On the other hand, $reset(f(e)) = res(\underline{a}) = res_1(\underline{a}) \wedge res_2(\underline{a})$.

□