Coalgebraic Specifications and Models of deterministic Hybrid Systems*

Bart Jacobs

CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands.

Email: bjacobs@cwi.nl Tel ++31 – 20 – 592.4008 Fax ++31 – 20 – 592.4199

Abstract. *Coalgebraic specification and semantics, as used earlier for object-oriented programming, is extended with temporal aspects. The (non-temporal) expression* s.meth *expressing that method* meth *is applied in state* s *is extended to an expression* s.meth@α, *where* α *is a time parameter. It means: in state* s *let the state evolve for* α *units of time, and then apply method* meth. *With this formalism we specify various (elementary) deterministic hybrid systems (and give a few simulations). We also define a notion of model for such a specification, and define what it means for a model to be terminal. Terminal models are "optimal" in the sense that they involve a minimal set of states, as will be illustrated in a number of examples. This shows that standard model theory can be applied to temporal (coalgebraic) specifications.*

1 Introduction

Hybrid systems combine discrete and continuous dynamics. They involve a combination of automata theory and differential equations. A typical hybrid system is a thermostat keeping the temperature in a room close to a goal temperature that can be set by a user. There are different control laws describing the temperature in the room as a function of time, depending on whether the heater is switched on or off. And if the temperature rises above the goal temperature then the heater will be switched off, and if the temperature falls below the goal, then the heater will be switched on. These discontinuities in the control law through internal actions are based on internal pre-programmed decisions. Further, the user can set a new goal temperature, causing a discontinuity as a result of an external action. Such a hybrid system can be seen as a kind of automaton, with different differential equations describing the continuous behaviour in different discrete states.

In this paper we propose a temporal specification format for (deterministic) hybrid systems that grew out of earlier work on object-oriented programming (see [16, 8, 10, 9]). This format is called "coalgebraic", because the underlying models are based on "coalgebras". These are the formal duals of algebras, in which one only has "destructors" (or "observers") as operations, instead of "constructors" in algebras. Coalgebras may be seen as abstract machines, consisting of a state space together with certain operations acting on this space. But typically, we have no means for constructing elements of the state space. Cofree coalgebras are used in [9] to describe inheritance. Here we extend this coalgebraic specification format as used for object-oriented programming with temporal aspects. We introduce a notation which allows us to indicate that a method (object-oriented terminology for operation) will be applied after a certain time delay. The new specification format of "temporal" coalgebraic specification contains assertions for reasoning both about states and about time. Thus we combine object-oriented specification with time, but we do not consider non-determinism or parallelism (at this stage). And we use assertional methods (in contrast to process algebraic methods) to describe and reason about these systems.

The additional time component in specification asks for an extension at the semantic level. We shall describe the influence of the elapse of time on a state space via a so-called "monoid action", acting on the state space. And the monoids we use are the monoids $\langle \mathbb{N}, 0, + \rangle$ and $\langle \mathbb{R}_{\geq 0}, 0, + \rangle$ of

*This paper appeared in: M. Wirsing and M. Nivat (eds.) *Algebraic Methods and Software Technology*, Springer LNCS 1101, 1996, p. 520–535.

discrete and real time. Monoid actions are fundamental in system theory see e.g. [11, Definition 1.1] (the "consistency" and "composition" conditions for the state transition function of a dynamical system). They occur in the form of an "evolution function" in [13, Definition 2.1]. These monoid actions arise naturally via (unique) solutions of differential equations. A model of a temporal coalgebraic specification will consist of a coalgebra together with a monoid action. The monoid action captures the continuous dependence of attributes on time, and also the internal actions, but the external (input and output) actions are described by the coalgebra. A subtle point is what definition should be taken for "homomorphism of models". The obvious notion of both a homomorphism of monoid actions (also called an equivariant mapping, see e.g. [3, 3.2.1 and 3.2.2]) and a homomorphism of coalgebras does not work (in the sense that it does not yield the terminal characterization of the intended models). Therefore we introduce a different notion, see Definition 5.1 below. It tells us what a "terminal model" is: it is characterized by the property—dual to the property that determines initial models—that from an arbitrary model there is a unique homomorphism to the terminal model. We will show in various examples that terminal models are "optimal" models in the sense that they have the minimal set of states. They form minimal realizations, in the terminology of [5]. And the terminal model is usually the intended model of a specification. Terminal models are special because they identify all observationally indistinguishable (bisimilar) states (see e.g. [17]). We find that forcing oneself to identify the terminal model is a great way to get one's specification right. In writing out the details of the terminal model it often became clear (in our experience) that the specification was incomplete, and that extra assertions had to be added.

Since we have a clear separation between specification and implementation, our work falls under the "two-language approach" distinguished in [15]. In the coalgebraic approach that we introduce below, invariance conditions form part of specifications (and need not be derived), since they describe essential aspects of models. Also, in contrast to the descriptions of hybrid systems as used in [13, 1], coalgebraic specifications are somewhat verbose, and contain many details. But with these details one can easily compute the values of attributes (see for example the computation after the REACT_A specification below). What we see as advantages of the coalgebraic framework are: it is intuitively clear, easy to manipulate, has a precise semantics, and offers the perspective of incorporating useful object-oriented notions like inheritance (for incremental specification and implementation, see [18, 9]) into the study of hybrid systems (see also [2]).

2 Monoids of time, and monoid actions

We recall that a monoid is a 3-tuple $\langle M, 0, +\rangle$ consisting a set M with a distinguished "zero" element $0 \in M$, and with a binary operation $+\colon M \times M \to M$ which is associative: $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$ and has $0 \in M$ as neutral element: $\alpha + 0 = \alpha$ and $0 + \alpha = \alpha$. Often we write M for the 3-tuple $\langle M, 0, +\rangle$ when $0, +$ are understood from the context. We shall mainly use the (commutative) monoids $\langle \mathbb{N}, 0, +\rangle$ of **discrete time** and $\langle \mathbb{R}_{\geq 0}, 0, +\rangle$ of **real time**, where $\mathbb{R}_{\geq 0} = \{\alpha \in \mathbb{R} \mid \alpha \geq 0\}$ is the set of positive reals. Actually we shall also use that these are *ordered* monoids (i.e. monoids in the category of posets).

Let $\langle M, 0, +\rangle$ be an arbitrary monoid. An **action** of this monoid on a set U consists of a function $\mu\colon U \times M \to U$ satisfying the following two requirements.

$$\mu(x, 0) = x \qquad \text{and} \qquad \mu(x, \alpha + \beta) = \mu(\mu(x, \alpha), \beta).$$

In the examples below, the set U will be the set of states of a certain abstract machine, and the function $\mu\colon U \times M \to U$ may be seen as giving for a state $x \in U$ and amount of time $\alpha \in M$ a new state $\mu(x, \alpha) \in U$ obtained by letting the machine run for α units of time starting in state x. The above two conditions express a certain linearity of this action: $\mu(x, 0) = x$ says that letting the machine run for 0 units of time does not change the state, and $\mu(x, \alpha + \beta) = \mu(\mu(x, \alpha), \beta)$ expresses that the effect of letting the machine run $\alpha + \beta$ units of time is the same as first letting it run α units of time, and then β units of time. But there are many more (non-temporal) instances of monoid-actions: for example, modules and vector spaces are monoid-actions, given by the application $(a, v) \mapsto a \cdot v$ of a scalar a to a vector v; it satisfies $1 \cdot v = v$ and $(ab) \cdot v = a \cdot (b \cdot v)$ and is thus a monoid action with respect to the (multiplicative) monoid structure on the scalars. Also, for a deterministic automaton with alphabet A and transition function $\delta\colon X \times A \to X$ there is (by induction) an extended transition function $\delta^{\star}\colon X \times A^{\star} \to X$ forming a monoid action with respect to the (free) monoid $A^{\star}$ of words. Finally, unique solutions to differential equations give rise to monoid actions, see e.g. [7, 8.7] (where they are called flows).

We mention a paradigmatic (temporal) example of a monoid action. It involves the "monus" function $\dot{-}$ (also called truncated subtraction) defined as follows.

$$x \mathbin{\dot{-}} y = \max(0, x - y) = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise.} \end{cases}$$

This monus will be used as a function $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and also as a function $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Let $U = \{s \in \mathbb{R}_{\geq 0} \mid s \leq 10\}$ be the set of states of a (real-time) timer, where the state $s \in U$ indicates that the timer will give a signal in s units of time. There is then an action $\mu\colon U \times \mathbb{R}_{\geq 0} \to U$ given by $\mu(s, \alpha) = s \mathbin{\dot{-}} \alpha$. Thus, if we have a state $5 \in U$ indicating that a signal will be given in 5 units of time, then the state $\mu(5, 3)$ obtained by letting the timer run for 3 units of time, is $2 \in U$. It is not hard to see that μ satisfies the two equations of a monoid action.

3 Coalgebraic specification

What distinguishes coalgebraic specification from algebraic specification is the use of "destructors" instead of "constructors" as atomic function symbols. Typically, if X is an unknown type that we are specifying and A is a constant set, then a map of the form $A \longrightarrow X$ is a constructor, since it tells us how to form elements of X, and a map $X \longrightarrow A$ is a destructor since it gives us some observations about what is in X. In the coalgebraic specification format in this paper we shall restrict ourselves to two kinds of destructors, of the form $\mathsf{at}\colon X \longrightarrow A$, and $\mathsf{proc}\colon X \times B \longrightarrow X$. The first of these is an attribute giving us some information about X, and the second one is a procedure which allows us to produce a new state (from a given one and a parameter element in a constant set B). Attributes correspond to (instance) variables, whose values may be changed by procedures, see the example below. We mostly use the object-oriented dot-notation instead of the functional notation. Hence for a state $\mathsf{s} \in X$ we write $\mathsf{s.at}$ for $\mathsf{at}(\mathsf{s})$ and $\mathsf{s.proc}(b)$ for $\mathsf{proc}(\mathsf{s}, b)$. Thus $\mathsf{s.proc}(b)\mathsf{.at}$ is the result of applying in state s the procedure proc with parameter b, and then applying the attribute at to its outcome. Functionally this would be written as $\mathsf{at}(\mathsf{proc}(\mathsf{s}, b))$.

Here is a typical example of a coalgebraic specification, provided with some comments after the #-sign.

class spec: FF	#	'FF' is the name of the specification; it stands for 'flip-flop'
methods:	#	object-oriented terminology for function symbols
$\mathsf{val}\colon X \longrightarrow \{0, 1\}$	#	this is an attribute, with output values 0 or 1.
$\mathsf{on}\colon X \longrightarrow X$	#	this a procedure without parameter, giving a new state.
$\mathsf{off}\colon X \longrightarrow X$	#	same thing
assertions:	#	statements imposing some behavioural restrictions, where $\mathsf{s} \in X$
$\mathsf{s.on.val} = 1$	#	thus, after 'on' in a state s the value is 1
$\mathsf{s.off.val} = 0$	#	now the end result is 0
creation:	#	requirement for the initial state new
$\mathsf{new.val} = 0$	#	hence newly created instances of FF have value 0.
end class spec		

The typically coalgebraic aspect of such a specification is that it tells nothing about what is inside X; it only describes the operations on X, and the constraints that they satisfy. We restrict equations (here and below) to be exclusively between attribute values, and not between states. This is in line with the coalgebraic philosophy in which states are not directly accessible. More examples may be found in [10, 9], giving coalgebraic specifications and models for classes in object-oriented languages.

A model of such a flip-flop specification consists of three parts. First, it consists of an interpretation $U = [\![X]\!]$ of the unknown type X as a set (of states). Secondly, the methods are interpreted as functions $[\![\,\mathsf{val}\,]\!]\colon U \to \{0,1\}$, $[\![\,\mathsf{on}\,]\!]\colon U \to U$ and $[\![\,\mathsf{off}\,]\!]\colon U \to U$ acting on the state space U, which should be such that the above assertions are satisfied. Usually we omit these interpretation braces $[\![-]\!]$. Thirdly, there should be an initial state $u_0 \in U$ satisfying the creation condition, i.e. satisfying $\mathsf{val}(u_0) = 0$. The three (interpretations of the) methods can be combined into a single function $U \to \{0,1\} \times U \times U$, giving us a **coalgebra** on U of the functor $X \mapsto \{0,1\} \times X \times X$.

Such a model $\langle U \to \{0,1\} \times U \times U, u_0 \in U\rangle$ is called **terminal** if for every model $\langle V \to \{0,1\} \times V \times V, v_0 \in V\rangle$ there is a unique function $f\colon V \to U$ preserving the operations and the initial state:

$$\mathsf{val}_U \circ f = \mathsf{val}_V, \qquad \mathsf{on}_U \circ f = f \circ \mathsf{on}_V, \qquad \mathsf{off}_U \circ f = f \circ \mathsf{off}_V, \qquad f(u_0) = v_0.$$

Terminal models form "minimal realizations" (in the terminology of [5]): they consist of the minimal set of states needed to perform the required behaviour. For example, the terminal model of the above flip-flop specification is the set $U = \{0,1\}$ of attribute values, with operations:

$$\begin{array}{ccc} \{0,1\} & \xrightarrow{\mathsf{val}} & \{0,1\} \\ x & \longmapsto & x \end{array} \qquad \begin{array}{ccc} \{0,1\} & \xrightarrow{\mathsf{on}} & \{0,1\} \\ x & \longmapsto & 1 \end{array} \qquad \begin{array}{ccc} \{0,1\} & \xrightarrow{\mathsf{off}} & \{0,1\} \\ x & \longmapsto & 0 \end{array}$$

and with $0 \in \{0,1\}$ as initial state. Indeed, for every model $\langle V \to \{0,1\} \times V \times V, v_0 \in V\rangle$ there is a unique homomorphism $f\colon V \to \{0,1\}$ satisfying the above requirements, namely $f = \mathsf{val}_V$. There are plenty of other models of this specification; for example, any set V with at least two elements can be turned into a model of this specification. But terminal models of coalgebraic specifications distinguish themselves as "optimal" models, in the same sense that initial (term) models of algebraic specifications are "optimal". See [4] for more information on the semantics of algebraic specifications.

Although we have described the notion of model only for a particular coalgebraic specification, it should be clear what a model is for an arbitrary coalgebraic specification: a carrier set together with functions acting on it which interpret the attributes and procedures, and satisfy the assertions, together with an initial state satisfying the creation conditions.

4 Temporal coalgebraic specification

In this section we extend coalgebraic specifications as above, with temporal aspects, and present a number of examples of the resulting "temporal coalgebraic specifications", together with a few simulations, using the OmSim simulator of OMOLA [2]. Semantics will be postponed until the next section.

A "temporal" coalgebraic specification is, like before, given by a collection of methods consisting of attributes and procedures, but the crucial difference lies in the formulations of the assertions. They will contain temporal information. For an arbitrary method meth and a state s we shall use the new notation

$\mathsf{s.meth}@\alpha$ for the result of applying method meth in state s after a delay of α units of time.

Or, more operationally, $\mathsf{s.meth}@\alpha$ means: in state s, wait α units of time and then apply method meth. We shall consider examples where α ranges over $\mathbb{N}$ (discrete time) and also over $\mathbb{R}_{\geq 0}$ (real time). We allow α to be 0, so that $\mathsf{meth}_1@\alpha.\mathsf{meth}_2@0$ means that meth_2 is applied immediately after meth_1 (which is applied after a delay of α time units). We assume that messages arrive in sequential order: if we write $\mathsf{s.meth}@\alpha$, then it is assumed that meth is the first method to be applied in state s (after α units of time), and that no other method was applied in the meantime. If meth is a method that takes a parameter $b \in B$ we shall write $\mathsf{s.meth}(b)@\alpha$ for the result of applying $\mathsf{meth}(b)$ after α units of time.

Let us consider an elementary example, building on the flip-flops from the previous section. Suppose we wish to specify flip-flops which can be switched on, and will automatically switch off after 10 units of discrete time. We specify these as follows.

$$
\begin{array}{lll}
\textbf{DT-class spec: } \mathrm{DTFF} & \# & \text{`DT' for `discrete time'; name `DTFF' for} \\
\quad \textbf{methods:} & \# & \quad \text{`discrete time flip-flop'} \\
\quad\quad \mathsf{val}\colon X \longrightarrow \{0,1\} & & \\
\quad\quad \mathsf{on}\colon X \longrightarrow X & \# & \text{(the method } \mathsf{off} \text{ is not used)}
\end{array}
$$

$$
\begin{array}{lll}
\quad \textbf{assertions:} & \# & \text{as before } \mathsf{s} \in X\text{; and } \alpha, \beta \in \mathbb{N} \text{ (discrete time)} \\
\quad\quad \mathsf{s.val}@\alpha = 0 \vdash \mathsf{s.val}@(\alpha + \beta) = 0 & \# & \text{``monotony'', forming an invariant} \\
\quad\quad \beta \geq 10 \vdash \mathsf{s.val}@\beta = 0 & \# & \text{also an invariant} \\
\quad\quad \beta < 10 \vdash \mathsf{s.on}@\alpha.\mathsf{val}@\beta = 1 & & \\
\quad \textbf{creation:} & & \\
\quad\quad \mathsf{new.val}@0 = 0 & & \\
\textbf{end class spec} & &
\end{array}
$$

We explain the meaning of the assertions. We use the turnstile $\vdash$ to describe conditional assertions. The first "monotony" assertion tells that if at some time α the value in state s is 0, then this value is still 0 at some later time $\alpha + \beta$. Hence the flip-flop does not simply switch on (get value 1) by itself. In this temporal coalgebraic format we have to indicate explicitly what the values of attributes are as a function of time. The second assertion tells us that no matter in what state our flip-flop is, if we wait at least 10 units of time, then its value will be 0. And finally, if we switch it on at some time α, and then inspect it at some time β less than 10 units later, then it will have value 1. This formally captures our informally described timer. Finally, the creation clause tells us that newly created instances have value 0 immediately after their creation. Then we can deduce $\mathsf{new.val}@\alpha = 0$ for any α, from the first assertion.

In order to familiarize the reader with this formalism, we consider some variations. Notice that a timed flip-flop satisfying the above specification can be switched on (again) if it has value 1. In this way we can keep it with value 1 for a longer time than 10. Suppose we wish to alter this and stipulate that the flip-flop can only be switched on if it has value 0. We can achieve this by taking the following two assertions, instead of the above third assertion.

$$
\begin{array}{l}
\mathsf{s.val}@\alpha = 0, \beta < 10 \vdash \mathsf{s.on}@\alpha.\mathsf{val}@\beta = 1 \\
\mathsf{s.val}@\alpha = 1 \vdash \mathsf{s.on}@\alpha.\mathsf{val}@\beta = \mathsf{s.val}@(\alpha + \beta).
\end{array}
$$

The first new assertion is like above, except that it now has an extra assumption that the value is 0 at the moment α that the 'on-event' happens. This reflects our modification. And the second assertion tells us that at a moment α when the value is 1, an 'on-event' has no effect on the value: looking at the value β time later is the same as looking at the value $\alpha + \beta$ time after the original state. For example, if we have a state s with $\mathsf{s.val}@2 = 0$, then if we switch it on 2 units after s, switch it on again 5 units later, and inspect 7 seconds later, then the value will be 0, although the inspection took place less than 10 units after an on-event. Formally:

$$
\begin{array}{rcll}
\mathsf{s.on}@2.\mathsf{on}@5.\mathsf{val}@7 & = & \mathsf{s.on}@2.\mathsf{val}@12 & \text{since } \mathsf{s.on}@2.\mathsf{val}@5 = 1, \\
& & & \quad \text{since } \mathsf{s.val}@2 = 0 \text{ and } 5 < 10 \\
& = & 0 & \text{since } 12 \geq 10.
\end{array}
$$

We can further modify this example by requiring that after the timer has had value 1, it must remain with value 0 for at least 20 units (say) of time. This comes close to the (single) traffic light specification for pedestrians in [6] with value 0 standing for "red light" and 1 for "green light". We need an auxiliary (possibly private) attribute $\mathsf{waiting}\colon X \longrightarrow \{\mathrm{yes}, \mathrm{no}\}$ telling us if we have waited long enough in a state with value 0 (to switch the flip-flop on again). Details of such a specification are left to the reader. Another variation on the above discrete-time flip-flop DTFF is a corresponding real-time RTFF, which will be discussed in Example 5.3 below. Similarly one can coalgebraically specify more standard examples from the literature—like a railway crossing explicitly taking account of the times needed to open and close the gate, or a watch-dog surveying a number of processes and expecting signals that everything is all-right at regular intervals (see e.g. [12, 19]).

We turn to some examples from chemistry, showing the interaction between the discrete structure of method-events and the continuous structure associated with the elapse of time, typical of hybrid systems. Assume we have control over a confined reaction space into which we can inject a chemical substance A. In this space, A will start reacting and transforming itself to another substance, with a reaction speed proportional to the available amount of A. If we write this amount as a function $A = A(\alpha)$ depending on a time parameter $\alpha \in \mathbb{R}_{\geq 0}$, then we have a differential equation

$$\frac{dA}{d\alpha} = -kA \qquad \text{where } k \in \mathbb{R}_{\geq 0} \text{ is a reaction constant.}$$

The solution of this equation is the function $A(\alpha) = A(0) \cdot e^{-k\alpha}$. It is used in the following specification.

RT-class spec: REACT_A
methods:
$\mathsf{amount}_A \colon X \longrightarrow \mathbb{R}_{\geq 0}$
$\mathsf{add}_A \colon X \times \mathbb{R}_{\geq 0} \longrightarrow X$
$\mathsf{clear} \colon X \longrightarrow X$
assertions:
$\mathsf{s.add}_A(x)@\alpha.\mathsf{amount}_A@0 = (\mathsf{s.amount}_A@\alpha) + x$
$\mathsf{s.clear}@\alpha.\mathsf{amount}_A@0 = 0$
$\mathsf{s.amount}_A@(\alpha + \beta) = (\mathsf{s.amount}_A@\alpha) \cdot e^{-k\beta}$
creation:
$\mathsf{new.amount}_A@0 = 0$
end class spec

Hence the amount_A attribute tells us how much A there is (in our confined reaction space). And with the two procedures add_A and clear we can inject a certain amount of A (using the parameter of the method), and clear the space in which we are working. This explains the first two assertions. The third assertion incorporates the solution of the differential equation: it tells what at any time β after α the amount of A is, in terms of the amount of A at α and the elapsed time β.

For example, we can do the following. In arbitrary state s, we first clear our working space, 1 time unit later we inject 10 units of A, then 8 time units later we decide to inject another 5 units of A, and then we check 3 time units later. The result can be computed as:

$$\begin{aligned}
&\mathsf{s.clear}@0.\mathsf{add}_A(10)@1.\mathsf{add}_A(5)@8.\mathsf{amount}_A@3 \\
&\quad= \mathsf{s.clear}@0.\mathsf{add}_A(10)@1.\mathsf{add}_A(5)@8.\mathsf{amount}_A@(0+3) \\
&\quad= (\mathsf{s.clear}@0.\mathsf{add}_A(10)@1.\mathsf{add}_A(5)@8.\mathsf{amount}_A@0) \cdot e^{-3k} \\
&\quad= (\mathsf{s.clear}@0.\mathsf{add}_A(10)@1.\mathsf{amount}_A@8 + 5) \cdot e^{-3k} \\
&\quad= ((\mathsf{s.clear}@0.\mathsf{add}_A(10)@1.\mathsf{amount}_A@0) \cdot e^{-8k} + 5) \cdot e^{-3k} \\
&\quad= ((\mathsf{s.clear}@0.\mathsf{amount}_A@1 + 10) \cdot e^{-8k} + 5) \cdot e^{-3k} \\
&\quad= (((\mathsf{s.clear}@0.\mathsf{amount}_A@0) \cdot e^{-k} + 10) \cdot e^{-8k} + 5) \cdot e^{-3k} \\
&\quad= (10 \cdot e^{-8k} + 5) \cdot e^{-3k} \\
&\quad= 10 \cdot e^{-11k} + 5 \cdot e^{-3k}.
\end{aligned}$$

The first factor shows the amount of A after inserting 10 units of A and waiting 11 time units, whereas the second factor shows the amount after waiting 3 time units starting from 5 units of A. This shows that one can actually calculate with a coalgebraic specification.

A more interesting example arises when we can (independently) insert two substances A and B, which can engage in reactions $A \leftrightarrows B$, both with reaction speed proportional to the amount of transforming substance, and such that an x-amount of A (resp. B) is transformed into an x-amount of B (resp. A). This leads to the differential equation

$$\frac{dA}{d\alpha} = -kA + \ell B \qquad \text{where} \qquad A(\alpha) + B(\alpha) = A(0) + B(0).$$

In the first equation, k, ℓ are constants (in $\mathbb{R}_{\geq 0}$). The second equation tells that the total amount of A plus B must be constant (and equal to the sum at initiation). The solution of this equation is

$$\begin{aligned} A(\alpha) &= \frac{1}{k+\ell}\Big((kA(0) - \ell B(0)) \cdot e^{-(k+\ell)\alpha} + \ell(A(0)+B(0))\Big) \\ B(\alpha) &= A(0) + B(0) - A(\alpha). \end{aligned}$$

This leads to the following specification.

RT-class spec: $\mathrm{REACT}_{A \rightleftarrows B}$
methods:
$\mathsf{amount}_A \colon X \longrightarrow \mathbb{R}_{\geq 0}$
$\mathsf{add}_A \colon X \times \mathbb{R}_{\geq 0} \longrightarrow X$
$\mathsf{amount}_B \colon X \longrightarrow \mathbb{R}_{\geq 0}$
$\mathsf{add}_B \colon X \times \mathbb{R}_{\geq 0} \longrightarrow X$
$\mathsf{clear} \colon X \longrightarrow X$
assertions:
$\mathsf{s.add}_A(x)@\alpha.\mathsf{amount}_A@0 = (\mathsf{s.amount}_A@\alpha) + x$
$\mathsf{s.add}_B(x)@\alpha.\mathsf{amount}_A@0 = \mathsf{s.amount}_A@\alpha$
$\mathsf{s.clear}@\alpha.\mathsf{amount}_A@0 = 0$
$$\begin{aligned} \mathsf{s.amount}_A@(\alpha+\beta) = \tfrac{1}{k+\ell}\Big(& (k(\mathsf{s.amount}_A@\alpha) - \ell(\mathsf{s.amount}_B@\alpha)) \cdot e^{-(k+\ell)\beta} \\ & +\ell((\mathsf{s.amount}_A@\alpha) + (\mathsf{s.amount}_B@\alpha))\Big) \end{aligned}$$
$\mathsf{s.add}_B(x)@\alpha.\mathsf{amount}_B@0 = (\mathsf{s.amount}_B@\alpha) + x$
$\mathsf{s.add}_A(x)@\alpha.\mathsf{amount}_B@0 = \mathsf{s.amount}_B@\alpha$
$\mathsf{s.clear}@\alpha.\mathsf{amount}_B@0 = 0$
$\mathsf{s.amount}_B@(\alpha+\beta) = (\mathsf{s.amount}_A@\alpha) + (\mathsf{s.amount}_B@\alpha) - (\mathsf{s.amount}_A@(\alpha+\beta))$
creation:
$\mathsf{new.amount}_A@0 = 0$
$\mathsf{new.amount}_B@0 = 0$
end class spec

Let s be an arbitrary state, and put $\mathsf{t} = \mathsf{s.clear}@0.\mathsf{add}_A(10)@0$. Then one can show that, as the time β goes to infinity, the amount $\mathsf{t.amount}_A@\beta$ of A in state t at time β goes to $\frac{\ell}{k+\ell} \cdot 10$, and the amount $\mathsf{t.amount}_B@\beta$ of B goes to $\frac{k}{k+\ell} \cdot 10$. What we have is an abstract description of a mini-chemical plant, in which two substances can be put together at controlled times and quantities, and their presence over time can be monitored. We have a "passive" hybrid system, because control is on the outside. See Figure 1 for the output of a simulation in OmSim [2].

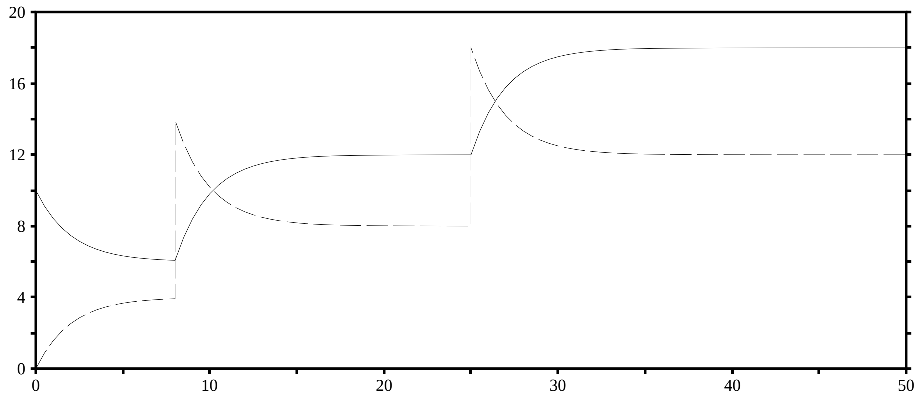

Figure 1: Initially: $A = 10, B = 0$. Additions of B: 10 at time = 8, and again 10 at time = 25. (The values of the constants in this simulation are: $k = 0.2$ and $\ell = 0.3$. Hence the eventual ratio $\frac{A}{B}$ is $\frac{3}{2}$.)

Our final hybrid example in this section involves a thermostat, and is adapted from [14, 1] (and put in coalgebraic format). We shall describe a "passive" and an "active" version. The passive thermostat PTHERM lets the user regulate the temperature in a room, via 'on' and 'off' switches of a heater (like for the earlier flip-flops). There are two attributes, namely 'val' describing whether the heater is on or off, and 'temp' describing the temperature in the room. We have to consider the following two cases.

- When the heater is off, the temperature in the room is determined by "Newton's law of cooling": the rate of change $\frac{dT}{d\alpha}$ of the temperature $T = T(\alpha)$ in the room is proportional to the difference between the temperature T in the room and the temperature of its surroundings. For convenience we assume the latter to be constantly 0, so that we have a differential equation

$$\frac{dT}{d\alpha} = -kT, \qquad \text{with solution} \qquad T(\alpha) = T(0) \cdot e^{-k\alpha}.$$

- If the heater is switched on, we assume that the change of temperature due to heating is constant. Hence we have an extra constant ℓ in our differential equation:

$$\frac{dT}{d\alpha} = -kT + \ell, \qquad \text{with solution} \qquad T(\alpha) = \left(T(0) - \frac{\ell}{k}\right) \cdot e^{-k\alpha} + \frac{\ell}{k}.$$

(These solutions are also used in [14, 1].) We thus arrive at the following specification.

RT-class spec: PTHERM
  methods:
    $\mathsf{val}\colon X \longrightarrow \{0, 1\}$
    $\mathsf{temp}\colon X \longrightarrow \mathbb{R}_{\geq 0}$
    $\mathsf{on}\colon X \longrightarrow X$
    $\mathsf{off}\colon X \longrightarrow X$
  assertions:
    $\mathsf{s.val@}(\alpha + \beta) = \mathsf{s.val@}\alpha$
    $\mathsf{s.on@}\alpha\mathsf{.val@}0 = 1$
    $\mathsf{s.off@}\alpha\mathsf{.val@}0 = 0$
    $\mathsf{s.on@}\alpha\mathsf{.temp@}0 = \mathsf{s.temp@}\alpha$
    $\mathsf{s.off@}\alpha\mathsf{.temp@}0 = \mathsf{s.temp@}\alpha$
    $\mathsf{s.val@}\alpha = 0 \vdash \mathsf{s.temp@}(\alpha + \beta) = (\mathsf{s.temp@}\alpha) \cdot e^{-k\beta}$
    $\mathsf{s.val@}\alpha = 1 \vdash \mathsf{s.temp@}(\alpha + \beta) = ((\mathsf{s.temp@}\alpha) - \frac{\ell}{k}) \cdot e^{-k\beta} + \frac{\ell}{k}$
  creation:
    $\mathsf{new.val@}0 = 1$
    $\mathsf{new.temp@}0 = 0$
end class spec

What is interesting about this example is that different states have different dynamic control laws: different formulas are used for the temperature in the room (as a function of the elapsed time) whether the heater is on (value 1) or off (value 0). In the last case only the natural loss of temperature is described: if $\beta \to \infty$, then the temperature at time $\alpha + \beta$ goes to 0. But if the heater is on there is an extra factor raising the temperature: if $\beta \to \infty$, then the temperature at $\alpha + \beta$ goes to the ratio $\frac{\ell}{k}$; this is the highest temperature that we can achieve by heating the room: it forms an equilibrium between heating and cooling. Notice that newly created thermostats have their heater on, and have a temperature equal to 0 (which is the temperature of the environment).

As an example, assume we have an arbitrary state s in which the value is 0 (heater is off), then if we switch the heater on after α time units, and then read the temperature β units later we get:

$$\begin{aligned}
\mathsf{s.on@}\alpha\mathsf{.temp@}\beta &= \mathsf{s.on@}\alpha\mathsf{.temp@}(0+\beta) \\
&= ((\mathsf{s.on@}\alpha\mathsf{.temp@}0) - \tfrac{\ell}{k}) \cdot e^{-k\beta} + \tfrac{\ell}{k} && \text{since } \mathsf{s.on@}\alpha\mathsf{.val@}0 = 1 \\
&= ((\mathsf{s.temp@}\alpha) - \tfrac{\ell}{k}) \cdot e^{-k\beta} + \tfrac{\ell}{k} \\
&= (\mathsf{s.temp@}(0+\alpha) - \tfrac{\ell}{k}) \cdot e^{-k\beta} + \tfrac{\ell}{k} \\
&= ((\mathsf{s.temp@}0) \cdot e^{-k\alpha} - \tfrac{\ell}{k}) \cdot e^{-k\beta} + \tfrac{\ell}{k} && \text{since } \mathsf{s.val@}0 = 0 \\
&= (\mathsf{s.temp@}0) \cdot e^{-k(\alpha+\beta)} + \tfrac{\ell}{k} \cdot (1 - e^{-k\beta}).
\end{aligned}$$

We call this a "passive" hybrid system because the heater will be switched on or off only as a result of an action of a client. A more user-friendly system allows a client to set the goal temperature, whereupon the system "actively" regulates the temperature. We shall specify such a system in which the temperature (after some time for adjustment) is kept in the interval $[z-1, z+1] \subseteq \mathbb{R}_{\geq 0}$ around the clients choice z. Therefore we assume that the highest possible temperature $\frac{\ell}{k}$ in the room is bigger than 2, and that the clients choice z lies in the open interval $(1, \frac{\ell}{k} - 1) \subseteq \mathbb{R}_{\geq 0}$.

The specification below has three attributes val, temp, goal for respectively the value of the heater (0=off, 1=on), the actual temperature in the room, and the goal temperature as set by the client. (Initially this goal will be set to $\frac{\ell}{2k}$, i.e. to half of the maximal temperature.) There is one procedure set, which allows a client to feed the desired temperature into the system. We shall use the abbreviations

$$\begin{aligned}
\uparrow(\mathsf{s}, \alpha) &\stackrel{\text{def}}{=} \sup\{\beta \mid ((\mathsf{s.temp@}\alpha) - \tfrac{\ell}{k}) \cdot e^{-k\beta} + \tfrac{\ell}{k} < (\mathsf{s.goal@}\alpha) + 1\} \\
&= \frac{1}{k} \ln\left(\frac{\ell - k(\mathsf{s.temp@}\alpha)}{\ell - k((\mathsf{s.goal@}\alpha) + 1)}\right) \\
\downarrow(\mathsf{s}, \alpha) &\stackrel{\text{def}}{=} \sup\{\beta \mid (\mathsf{s.temp@}\alpha) \cdot e^{-k\beta} > (\mathsf{s.goal@}\alpha) - 1\} \\
&= \frac{1}{k} \ln\left(\frac{\mathsf{s.temp@}\alpha}{(\mathsf{s.goal@}\alpha) - 1}\right)
\end{aligned}$$

for the time $\uparrow(\mathsf{s}, \alpha)$ needed in state s at α to reach the maximum $(\mathsf{s.goal@}\alpha) + 1$ by heating, and the time $\downarrow(\mathsf{s}, \alpha)$ needed to reach the minimum $(\mathsf{s.goal@}\alpha) - 1$. These abbreviations will be used for "time can proceed" (tcp) predicates—like in [13]—in the following specification.

RT-class spec: ATHERM
methods:
$\mathsf{val}\colon X \longrightarrow \{0,1\}$
$\mathsf{temp}\colon X \longrightarrow \mathbb{R}_{\geq 0}$
$\mathsf{goal}\colon X \longrightarrow (1, \frac{\ell}{k} - 1)$
$\mathsf{set}\colon X \times (1, \frac{\ell}{k} - 1) \longrightarrow X$
assertions:
$\mathsf{s.temp}@\alpha < \frac{\ell}{k}$
$\mathsf{s.goal}@(\alpha+\beta) = \mathsf{s.goal}@\alpha$
$\mathsf{s.temp}@\alpha < (\mathsf{s.goal}@\alpha) - 1 \vdash \mathsf{s.val}@\alpha = 1$
$\mathsf{s.temp}@\alpha > (\mathsf{s.goal}@\alpha) + 1 \vdash \mathsf{s.val}@\alpha = 0$
$\mathsf{s.temp}@\alpha \geq a \vdash \mathsf{s.set}(a)@\alpha.\mathsf{val}@0 = 0$
$\mathsf{s.temp}@\alpha < a \vdash \mathsf{s.set}(a)@\alpha.\mathsf{val}@0 = 1$
$\mathsf{s.set}(a)@\alpha.\mathsf{temp}@0 = \mathsf{s.temp}@\alpha$
$\mathsf{s.set}(a)@\alpha.\mathsf{goal}@0 = a$
$\mathsf{s.val}@\alpha = 1, \beta < \uparrow(\mathsf{s},\alpha) \vdash \mathsf{s.val}@(\alpha+\beta) = 1$
$\mathsf{s.val}@\alpha = 1, \beta < \uparrow(\mathsf{s},\alpha) \vdash \mathsf{s.temp}@(\alpha+\beta) = ((\mathsf{s.temp}@\alpha) - \frac{\ell}{k}) \cdot e^{-k\beta} + \frac{\ell}{k}$
$\mathsf{s.val}@\alpha = 1 \vdash \mathsf{s.val}@(\alpha + \uparrow(\mathsf{s},\alpha)) = 0$
$\mathsf{s.val}@\alpha = 1 \vdash \mathsf{s.temp}@(\alpha + \uparrow(\mathsf{s},\alpha)) = (\mathsf{s.goal}@\alpha) + 1$
$\mathsf{s.val}@\alpha = 0, \beta < \downarrow(\mathsf{s},\alpha) \vdash \mathsf{s.val}@(\alpha+\beta) = 0$
$\mathsf{s.val}@\alpha = 0, \beta < \downarrow(\mathsf{s},\alpha) \vdash \mathsf{s.temp}@(\alpha+\beta) = (\mathsf{s.temp}@\alpha) \cdot e^{-k\beta}$
$\mathsf{s.val}@\alpha = 0 \vdash \mathsf{s.val}@(\alpha + \downarrow(\mathsf{s},\alpha)) = 1$
$\mathsf{s.val}@\alpha = 1 \vdash \mathsf{s.temp}@(\alpha + \downarrow(\mathsf{s},\alpha)) = (\mathsf{s.goal}@\alpha) - 1$
creation:
$\mathsf{new.val}@0 = 1$ # in fact, this can be deduced
$\mathsf{new.temp}@0 = 0$
$\mathsf{new.goal}@0 = \frac{\ell}{2k}$
end class spec

We leave it to the reader to verify that for a state s with $(\mathsf{s.goal}@\alpha) - 1 \leq \mathsf{s.temp}@\alpha \leq (\mathsf{s.goal}@\alpha) + 1$,

$$\begin{aligned} \mathsf{s.temp}@\alpha &= \mathsf{s.temp}@(\alpha + \uparrow(\mathsf{s.goal}@\alpha) + \downarrow(\mathsf{s.goal}@\alpha)) \\ \mathsf{s.val}@\alpha &= \mathsf{s.val}@(\alpha + \uparrow(\mathsf{s.goal}@\alpha) + \downarrow(\mathsf{s.goal}@\alpha)) \end{aligned}$$

where $\uparrow z$ (resp. $\downarrow z$) is the time that is required for the temperature to rise from $z-1$ to $z+1$, (resp. to fall from $z+1$ to $z-1$). Hence, once the temperature has reached the required region around the goal temperature, it will oscillate around this goal with a periodicity of $\uparrow(\mathsf{s.goal}@\alpha) + \downarrow(\mathsf{s.goal}@\alpha)$, and it will stay within this region $[(\mathsf{s.goal}@\alpha) - 1, (\mathsf{s.goal}@\alpha) + 1]$. Further, the heater will be switched on and off with the same periodicity. See Figure 2 for an OmSim simulation.

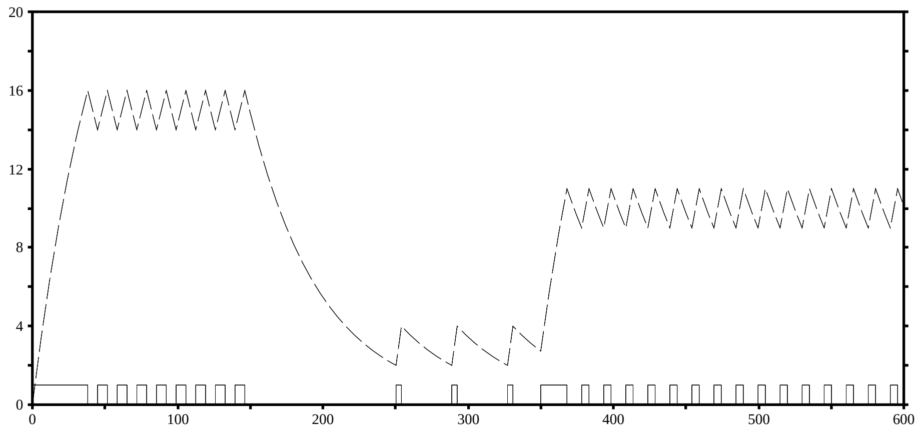

Figure 2: Initially: goal temperature = 15. Goal is set to 3 at time = 150, and to 10 at time = 350. The dashed line describes the resulting temperature, and the blocks at the bottom indicate whether the heater is on or off.

5 Models of temporal coalgebraic specifications

We now turn to semantics. In this section notions of "model" and of "terminal model" will be introduced for the temporal coalgebraic specifications from the previous section. Subsequently, terminal models will be identified for these example specifications.

First, we reconsider the specification DTFF of discrete-time flip-flops as described in the beginning of the previous section. A model of such a specification will first of all be a model of the "underlying" coalgebraic specification as in Section 4—obtained by ignoring aspects of time; that is, by taking the time parameters α, β equal to 0 in the specification. Thus we should have a carrier set U of states, together with operations $\mathsf{val}: U \to \{0,1\}$ and $\mathsf{on}: U \to U$, and an initial state $u_0 \in U$. What we further need (in this discrete time case) is an $\mathbb{N}$-action $\mu: U \times \mathbb{N} \to U$ describing the influence of time on the state space. Then we can interpret timed methods as:

$$\mathsf{s.val@}\alpha = \mathsf{val}(\mu(\mathsf{s}, \alpha)) \qquad \text{and} \qquad \mathsf{s.on@}\alpha = \mathsf{on}(\mu(\mathsf{s}, \alpha)).$$

This interpretation corresponds with our earlier operational explanation: $\mathsf{s.val@}\alpha$ means: first wait α units of time and take the resulting state $\mu(\mathsf{s}, \alpha)$ emerging after this period of time, then apply the val-method. Notice that $\mathsf{s.val@}0 = \mathsf{s.val}$, because μ is an action. Similarly for the procedure on.

In order to find concrete examples of models it is useful to think of elements of the carrier set U as internal states needed to display the specified behaviour. In this case we can take as internal states the elements $\mathsf{s} \in \mathbb{N}$ with $\mathsf{s} \leq 10$. Such a state s can be seen as the number of units of time before the value of the flip-flop becomes 0. This explains the maximum 10. We thus take $U = \{0, 1, \ldots, 10\} \subseteq \mathbb{N}$ as underlying state state space, with action

$$\mu: \{0, 1, \ldots, 10\} \times \mathbb{N} \longrightarrow \{0, 1, \ldots, 10\} \qquad \text{given by} \qquad (\mathsf{s}, \alpha) \longmapsto s \dot{-} \alpha.$$

The interpretations of the methods are then:

$$\{0, 1, \ldots, 10\} \xrightarrow{\mathsf{val}} \{0, 1\} \;\text{ is }\; \mathsf{s} \mapsto \begin{cases} 0 & \text{if } \mathsf{s} = 0 \\ 1 & \text{else} \end{cases} \quad \text{and} \;\; \{0, 1, \ldots, 10\} \xrightarrow{\mathsf{on}} \{0, 1, \ldots, 10\} \;\text{ is }\; \mathsf{s} \mapsto 10.$$

And as initial state in this model we take $0 \in \{0, 1, \ldots, 10\}$. We verify that the assertions of the DTFF specification hold in this model.

(i) The first assertion $\mathsf{s.val@}\alpha = 0 \vdash \mathsf{s.val@}(\alpha + \beta) = 0$ holds since

$$\mathsf{s.val@}\alpha = 0 \Rightarrow \mu(\mathsf{s}, \alpha) = \mathsf{s} \dot{-} \alpha = 0 \Rightarrow \alpha \geq \mathsf{s} \Rightarrow \alpha + \beta \geq \mathsf{s} \Rightarrow \mathsf{s.val@}(\alpha + \beta) = 0.$$

(ii) The second assertion $\beta \geq 10 \vdash \mathsf{s.val}@\beta = 0$ obviously holds, since for $\mathsf{s} \in \{0, 1, \ldots, 10\}$ and $\beta \geq 10$ one has $\mathsf{s} \dot{-} \beta = 0$.

(iii) And the last assertion $\beta < 10 \vdash \mathsf{s.on}@\alpha.\mathsf{val}@\beta = 1$ holds since

$$\beta < 10 \Rightarrow 10 \dot{-} \beta > 0 \Rightarrow (\mathsf{s.on}@\alpha) \dot{-} \beta > 0 \Rightarrow \mathsf{s.on}@\alpha.\mathsf{val}@\beta = 1.$$

(iv) Finally, the initial state $0 \in \{0, 1, \ldots, 10\}$ satisfies the creation condition, because $0.\mathsf{val}@0 = 0$, since $\mu(0, 0) = 0 \dot{-} 0 = 0$.

There are other models of this DTFF specification besides $\{0, 1, \ldots, 10\} \subseteq \mathbb{N}$. One can also take the closed intervals $[0, 10] \subseteq \mathbb{Q}$ and $[0, 10] \subseteq \mathbb{R}$ of (positive) rational and real numbers below 10. The definitions of the action and methods are as above. But in these models of rationals and reals there are "too many" states[1]. The minimality of the model $\{0, 1, \ldots, 10\} \subseteq \mathbb{N}$ can be expressed mathematically using terminality. This will be formulated next.

5.1. Definition. Consider a (discrete or real time) specification S as above, with attributes $X \longrightarrow A_1, \ldots, X \longrightarrow A_n$ and procedures $X \times B_1 \longrightarrow X, \ldots, X \times B_m \longrightarrow X$.

(i) A **model** of this specification consists of four parts:

(a) a "state space" or "carrier set" U, serving as interpretation $U = [\![X]\!]$ of the unknown type X in the specification S; elements of U will be called states;

(b) a monoid action $\mu: U \times M \to U$, where M is the monoid of discrete or real time (in accordance with whether S is a discrete or real time specification);

(c) functions $U \to A$, $U \times B \to U$, where $A = A_1 \times \cdots \times A_n$ is the product of the sets of attribute values and $B = B_1 + \cdots + B_m$ is the coproduct (disjoint union) of the procedure parameter sets, giving combined interpretations $U \to A_i$ of the attributes and of the procedures $U \times B_j \to U$, in such a way that the assertions of the specification S are satisfied;

(d) An initial state $u_0 \in U$ satisfying the creation conditions in the specification S.

We notice that the interpretations of the attributes and of the procedures form a *coalgebra* $U \to A \times U^B$ on the state space U. Hence we are describing coalgebraic models of temporal specifications.

(ii) Such a model $\langle U \xrightarrow{\mathsf{at}} A, U \times B \xrightarrow{\mathsf{proc}} U, U \times M \xrightarrow{\mu} U, u_0 \in U \rangle$ is called **terminal** if for every model $\langle V \xrightarrow{\mathsf{at}} A, V \times B \xrightarrow{\mathsf{proc}} V, V \times M \xrightarrow{\nu} V, v_0 \in V \rangle$ there is a unique function $f: V \to U$ making the following three diagrams commute.

$$\begin{tikzcd}
V \times M \arrow[r, "\nu"] \arrow[d, "f \times id"'] & V \arrow[r, "\mathsf{at}"] & A \arrow[d, equal] \\
U \times M \arrow[r, "\mu"'] & U \arrow[r, "\mathsf{at}"'] & A
\end{tikzcd}
\qquad
\begin{tikzcd}
(V \times M) \times B \arrow[r, "\nu \times id"] \arrow[d, "(f \times id) \times id"'] & V \times B \arrow[r, "\mathsf{proc}"] & V \arrow[d, "f"] \\
(U \times M) \times B \arrow[r, "\mu \times id"'] & U \times B \arrow[r, "\mathsf{proc}"'] & U
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, "v_0"] \arrow[d, equal] & V \arrow[d, "f"] \\
1 \arrow[r, "u_0"'] & U
\end{tikzcd}$$

That is, f satisfies for $v \in V$, $\alpha \in M$ and $b \in B$:

$$f(v).\mathsf{at}@\alpha = v.\mathsf{at}@\alpha \qquad \text{and} \qquad f(v).\mathsf{proc}(b)@\alpha = f(v.\mathsf{proc}(b)@\alpha) \qquad \text{and} \qquad f(v_0) = u_0.$$

A function f satisfying these three requirements will sometimes be called a **homomorphism** (of models).

In the notion of homomorphism used in the definition, the internal time steps are not preserved directly, but only indirectly via their observable effect. As an aside we mention that every model $\langle V \xrightarrow{\mathsf{at}} A, V \times B \xrightarrow{\mathsf{proc}} V, V \times M \xrightarrow{\nu} V, v_0 \in V \rangle$ yields a coalgebra $V \to A^M \times V^{B \times M}$ by $v \mapsto \langle \lambda\alpha.\, v.\mathsf{at}@\alpha, \lambda(b, \alpha).\, v.\mathsf{proc}(b)@\alpha \rangle$. The notion of homomorphism used in this definition corresponds to a morphism of coalgebras for the functor $X \mapsto A^M \times X^{B \times M}$. And associated with this functor is a notion of *bisimulation* relation. It is a relation $R \subseteq V \times V$ on the carrier of a model V such that for all x, y with $R(x, y)$ one has $R(\nu(x, \alpha), \nu(y, \alpha))$ and $x.\mathsf{at}@\alpha = y.\mathsf{at}@\alpha$ and $R(x.\mathsf{proc}(b)@\alpha, y.\mathsf{proc}(b)@\alpha)$, for all $\alpha \in M$ and $b \in B$. By a standard result, bisimilar elements become equal when mapped to the terminal model.

[1]But for a client who can only use the specified methods, these differences of implementation are not noticeable.

We have already seen examples of models. We now present an example of a terminal model.

5.2. Example (Discrete-time flip-flop). Consider the discrete time flip-flop specification DTFF, with its model $U = \{0, 1, \ldots, 10\} \subseteq \mathbb{N}$ as described in the beginning of this section. This model can be characterized as terminal model of the specification. And this formalizes our earlier intuition that it involves the minimal set of states (or: forms a "minimal realization"). For any model with carrier set V, action $\nu: V \times \mathbb{N} \to V$, methods $\mathsf{val}: V \to \{0, 1\}$, $\mathsf{on}: V \to V$ and initial state $v_0 \in V$, there is a function

$$f: V \longrightarrow \{0, 1, \ldots, 10\} \qquad \text{given by} \qquad f(v) = \inf\{\beta \in \{0, 1, \ldots, 10\} \mid v.\mathsf{val}@\beta = 0\}.$$

Thus f maps a state $v \in V$ to the first unit of time where the value of state v is 0. We show that f is a homomorphism.

(i) Commutation with val:

$$\begin{aligned} f(v).\mathsf{val}@\alpha = 0 \;&\Leftrightarrow\; \mu(f(v), \alpha) = f(v) \mathbin{\dot{-}} \alpha = 0 \\ &\Leftrightarrow\; \alpha \geq f(v) = \inf\{\beta \in \{0, 1, \ldots, 10\} \mid v.\mathsf{val}@\beta = 0\} \\ &\Leftrightarrow\; v.\mathsf{val}@\alpha = 0. \end{aligned}$$

The direction ($\Leftarrow$) of the last step is easy, by definition of infimum. For ($\Rightarrow$), assume $\alpha \geq f(v)$, say $\alpha = f(v) + \beta$. Since $v.\mathsf{val}@f(v) = 0$—because $f(v)$ is the first time that the value is 0—we get $v.\mathsf{val}@(f(v) + \beta) = v.\mathsf{val}@\alpha = 0$ by the monotony assertion in the DTFF specification.

(ii) Commutation with on:

$$\begin{aligned} f(v.\mathsf{on}@\alpha) &= \inf\{\beta \in \{0, 1, \ldots, 10\} \mid v.\mathsf{on}@\alpha.\mathsf{val}@\beta = 0\} \\ &= \inf\{\beta \in \{0, 1, \ldots, 10\} \mid \mu(v.\mathsf{on}@\alpha, \beta) = (v.\mathsf{on}@\alpha) \mathbin{\dot{-}} \beta = 10 \mathbin{\dot{-}} \beta = 0\} \\ &= \inf\{\beta \in \{0, 1, \ldots, 10\} \mid \beta \geq 10\} \\ &= 10 \\ &= f(v).\mathsf{on}@\alpha. \end{aligned}$$

(iii) Preservation of the initial state:

$$f(v_0) = \inf\{\beta \in \{0, 1, \ldots, 10\} \mid v_0.\mathsf{val}@\beta = 0\} = \inf\{\beta \in \{0, 1, \ldots, 10\} \mid \beta \geq 0\} = 0.$$

Finally we have to show that f is unique with these properties. If also $g: V \to \{0, 1, \ldots, 10\}$ satisfies $g(v).\mathsf{val}@\alpha = v.\mathsf{val}@\alpha$, $g(v).\mathsf{on}@\alpha = g(v.\mathsf{on}@\alpha)$ and $g(v_0) = 0$, then $g(v) = f(v)$ since:

- $g(v) \leq f(v)$ because $g(v)$ is a lower bound of the set $\{\beta \in \{0, 1, \ldots, 10\} \mid v.\mathsf{val}@\beta = 0\}$:

$$v.\mathsf{val}@\beta = 0 \;\Rightarrow\; g(v).\mathsf{val}@\beta = 0 \;\Rightarrow\; \mu(g(v), \beta) = g(v) \mathbin{\dot{-}} \beta = 0 \;\Rightarrow\; g(v) \leq \beta.$$

- $g(v) \geq f(v)$ because $v.\mathsf{val}@g(v) = 0$. This follows since $g(v).\mathsf{val}@g(v) = 0$ since $\mu(g(v), g(v)$ $= g(v) \mathbin{\dot{-}} g(v) = 0$.

5.3. Example (Real-time flip-flop). We now consider a real-time version of the above timed flip-flop. Its specification is the same as the discrete time specification (in the beginning of Section 4) except that in order to deal with boundary problems we add an extra "denseness" assertion

$$\mathsf{s.val}@\alpha = 1 \vdash \exists \beta > 0\; \mathsf{s.val}@(\alpha + \beta) = 1.$$

It tells us that if the value at time α is 1, then we can always find a (possibly very small) non-zero positive real number β such that β units of time later the value is still 1. As a consequence, if $v.\mathsf{val}@\alpha = 1$, then the set $\{\beta \mid \mathsf{s.val}@(\alpha + \beta) = 1\}$ is an upwardly open interval $[0, \gamma) \subseteq \mathbb{R}_{\geq 0}$.

We claim that the terminal model satisfying this RTFF specification is the closed interval $[0, 10] \subseteq \mathbb{R}$ of positive reals less than or equal to 10. The idea is that a state $\mathsf{s} \in [0, 10]$ represents the state of the flip-flop in which the value will be 0 in s units of time. The (real-time) action $\mu\colon [0,10] \times \mathbb{R}_{\geq 0} \to [0,10]$ is $(\mathsf{s}, \alpha) \mapsto \mathsf{s} \mathbin{\dot{-}} \alpha$. And the method interpretations are

$$\mathsf{val}\colon [0,10] \longrightarrow \{0,1\} \quad \text{is} \quad \mathsf{s} \mapsto \begin{cases} 0 & \text{if } \mathsf{s}=0 \\ 1 & \text{else} \end{cases} \qquad \text{and} \qquad \mathsf{on}\colon [0,10] \longrightarrow [0,10] \quad \text{is} \quad \mathsf{s} \mapsto 10.$$

with $0 \in [0,10]$ as initial state—much as in the discrete case. We check the validity of the extra assertion $\mathsf{s.val}@\alpha = 1 \vdash \exists \beta > 0\ \mathsf{s.val}@(\alpha+\beta) = 1$. If for some state $\mathsf{s} \in [0,10]$ and time $\alpha \in \mathbb{R}_{\geq 0}$ we have $\mathsf{s.val}@\alpha = 1$, then $\mathsf{s} \mathbin{\dot{-}} \alpha > 0$, so that $\alpha < \mathsf{s}$. But then we can find a $\beta > 0$ with $\alpha + \beta < \mathsf{s}$ because $\mathbb{R}_{\geq 0}$ is dense. This means that $\mathsf{s.val}@(\alpha+\beta) = 1$.

In order to show terminality of the model $[0,10]$, assume another model consisting of a carrier set V, with action $\nu\colon V \times \mathbb{R}_{\geq 0} \to V$, method interpretations $\mathsf{val}\colon V \to \{0,1\}$, $\mathsf{on}\colon V \to V$ and initial state $v_0 \in V$. Then we can define a function $f\colon V \to [0,10]$ by $f(v) = \inf\{\beta \in [0,10] \mid v.\mathsf{val}@\beta = 0\}$. We show that $f(v).\mathsf{val}@\alpha = v.\mathsf{val}@\alpha$. Indeed

$$f(v).\mathsf{val}@\alpha = 0 \;\Leftrightarrow\; f(v) \mathbin{\dot{-}} \alpha = 0 \;\Leftrightarrow\; \alpha \geq f(v) \;\overset{(*)}{\Leftrightarrow}\; v.\mathsf{val}@\alpha = 0.$$

The marked implication ($\Leftarrow$) is easy by definition of infimum. For ($\Rightarrow$) we use that $v.\mathsf{val}@f(v) = 0$. Suppose not, i.e. $v.\mathsf{val}@f(v) = 1$. Then we can find a $\gamma > 0$ with $v.\mathsf{val}@(f(v)+\gamma) = 1$, by the additional assertion mentioned above. But $f(v) + \gamma$ is a lower bound for the $\beta \in [0,10]$ with $v.\mathsf{val}@\beta = 0$. Hence $f(v) + \gamma \geq f(v)$, because $f(v)$ is infimum. But this is impossible.

The remaining details that f is the unique homomorphism $V \to [0,10]$ are as in the previous example, and are left to the reader.

5.4. Example (Chemical reactions). We consider the specifications REACT_A and $\mathrm{REACT}_{A\rightleftarrows B}$ from the previous section. In the first case a model has to keep track of the amount of the chemical substance A. This is done most economically by taking as state space the set $\mathbb{R}_{\geq 0}$ of positive reals, elements of which represent this amount of A. The associated action $\mu\colon \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ sends a pair (x, α) consisting of the present amount x of A and the time α, to the amount $\mu(x,\alpha) = x \cdot e^{-k\alpha}$ after α time units. This is an action, since $\mu(x,0) = x \cdot e^0 = x \cdot 1 = x$, and $\mu(\mu(x,\alpha),\beta) = \mu(x,\alpha)\cdot e^{-k\beta} = (x \cdot e^{-k\alpha}) \cdot e^{-k\beta} = x \cdot e^{-k(\alpha+\beta)} = \mu(x, \alpha+\beta)$. The interpretations of the methods amount_A, add_A and clear are then simply:

$$\begin{array}{ccc} \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{add}_A} & \mathbb{R}_{\geq 0} \\ \langle x, y \rangle & \longmapsto & x + y \end{array} \qquad \begin{array}{ccc} \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{amount}_A} & \mathbb{R}_{\geq 0} \\ x & \longmapsto & x \end{array} \qquad \begin{array}{ccc} \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{clear}} & \mathbb{R}_{\geq 0} \\ x & \longmapsto & 0. \end{array}$$

As initial state we have to take $0 \in \mathbb{R}_{\geq 0}$. This is the terminal model, since for an arbitrary model V with action $\nu\colon V \times \mathbb{R}_{\geq 0} \to V$, attribute $\mathsf{amount}_A\colon V \to \mathbb{R}_{\geq 0}$, procedures $\mathsf{add}_A\colon V \times \mathbb{R}_{\geq 0} \to V$, $\mathsf{clear}\colon V \to V$ with initial state $v_0 \in V$, we get a unique homomorphism $f\colon V \to \mathbb{R}_{\geq 0}$, namely $f(v) = v.\mathsf{amount}_A@0$. Then

$$\begin{array}{rl} & f(v).\mathsf{amount}_A@\alpha \\ = & \mu(f(v), \alpha) \\ = & f(v) \cdot e^{-k\alpha} \\ = & (v.\mathsf{amount}_A@0) \cdot e^{-k\alpha} \\ = & v.\mathsf{amount}_A@(0+\alpha) \\ = & v.\mathsf{amount}_A@\alpha \end{array} \qquad \begin{array}{rl} & f(v).\mathsf{add}_A(x)@\alpha \\ = & \mu(f(v),\alpha) + x \\ = & v.\mathsf{amount}_A@\alpha + x \\ = & v.\mathsf{add}_A(x)@\alpha.\mathsf{amount}_A@0 \\ = & f(v.\mathsf{add}_A(x)@\alpha) \end{array} \qquad \begin{array}{rl} & f(v).\mathsf{clear}@\alpha \\ = & 0 \\ = & v.\mathsf{clear}@\alpha.\mathsf{amount}_A@0 \\ = & f(v.\mathsf{clear}@\alpha). \end{array}$$

And also $f(v_0) = v_0.\mathsf{amount}_A@0 = 0$. Uniqueness is obvious.

A model of the second specification $\mathrm{REACT}_{A\rightleftarrows B}$ must keep track of both the amounts of A and of B. This suggests $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ as carrier of the terminal model. We only define the action, and leave further details to the reader. This action $\mu\colon (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}) \times \mathbb{R}_{\geq 0} \to (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0})$ is given by

$$\begin{array}{l} \mu(\langle x_A, x_B \rangle, \alpha) = \langle \mu_A(\langle x_A, x_B\rangle, \alpha), x_A + x_B - \mu_A(\langle x_A, x_B \rangle, \alpha) \rangle \\ \qquad\qquad \text{where } \mu_A(\langle x_A, x_B \rangle, \alpha) = \dfrac{1}{k+\ell}\Big((kx_A - \ell x_B)\cdot e^{-(k+\ell)\alpha} + \ell(x_A + x_B)\Big) \end{array}$$

Notice that $\mu_A(\langle x_A, x_B\rangle, 0) = \frac{1}{k+\ell}(\ell(x_A + x_B) + kx_A - \ell x_B) = \frac{1}{k+\ell}((\ell + k)x_A) = x_A$, and thus $\mu(\langle x_A, x_B\rangle, 0) = \langle x_A, x_A + x_B - x_A\rangle = \langle x_A, x_B\rangle$. The other action-equation $\mu(\mu(\langle x_A, x_B\rangle, \alpha), \beta) = \mu(\langle x_A, x_B\rangle, \alpha + \beta)$ is left to the interested reader.

5.5. Example (Thermostats). The passive and active thermostat PTHERM and ATHERM in the previous section also involve non-trivial actions. The terminal model in the passive case PTHERM has carrier set $\{0,1\} \times \mathbb{R}_{\geq 0}$, where the first component describes whether the heater is on (1) or off (0), and the second component is the current temperature. This implementation contains all the information we need, and nothing more. The action $\mu: (\{0,1\} \times \mathbb{R}_{\geq 0}) \times \mathbb{R}_{\geq 0} \to \{0,1\} \times \mathbb{R}_{\geq 0}$ is given by

$$\mu(\langle 0, x\rangle, \alpha) = \langle 0, x \cdot e^{-k\alpha}\rangle \qquad \text{and} \qquad \mu(\langle 1, x\rangle, \alpha) = \langle 1, \left(x - \frac{\ell}{k}\right) \cdot e^{-k\alpha} + \frac{\ell}{k}\rangle$$

where k, ℓ are the constants as used in the PTHERM-specification. It is not hard to check that μ is an action. The interpretations of the methods on the state space $\{0,1\} \times \mathbb{R}_{\geq 0}$ are given by

$$\begin{array}{rcl} \{0,1\} \times \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{val}} & \{0,1\} \\ \langle z, x\rangle & \longmapsto & z \end{array} \qquad \begin{array}{rcl} \{0,1\} \times \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{temp}} & \mathbb{R}_{\geq 0} \\ \langle z, x\rangle & \longmapsto & x \end{array}$$

$$\begin{array}{rcl} \{0,1\} \times \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{on}} & \{0,1\} \times \mathbb{R}_{\geq 0} \\ \langle z, x\rangle & \longmapsto & \langle 1, x\rangle \end{array} \qquad \begin{array}{rcl} \{0,1\} \times \mathbb{R}_{\geq 0} & \xrightarrow{\mathsf{off}} & \{0,1\} \times \mathbb{R}_{\geq 0} \\ \langle z, x\rangle & \longmapsto & \langle 0, x\rangle. \end{array}$$

As initial state we take $\langle 1, 0\rangle \in \{0,1\} \times \mathbb{R}_{\geq 0}$, as prescribed by the specification. We leave it to the reader to verify that the assertions in the PTHERM-specification hold in this model.

If we have another PTHERM-model with carrier set V, action $V \times \mathbb{R}_{\geq 0} \to V$, method interpretations $\mathsf{val}: V \to \{0,1\}$, $\mathsf{temp}: V \to \mathbb{R}_{\geq 0}$, $\mathsf{on}: V \to V$, $\mathsf{off}: V \to V$ and initial state $v_0 \in V$. Then there is a unique homomorphims $f: V \to \{0,1\} \times \mathbb{R}_{\geq 0}$, namely $f(v) = \langle v.\mathsf{val}@0, v.\mathsf{temp}@0\rangle$. We only check that f commutes with the temperature attributes. If $v.\mathsf{val}@0 = 0$, then

$$f(v).\mathsf{temp}@\alpha = \mathsf{snd}\mu(f(v), \alpha) = (\mathsf{snd}f(v)) \cdot e^{-k\alpha} = (v.\mathsf{temp}@0) \cdot e^{-k\alpha} = v.\mathsf{temp}@\alpha.$$

And if $v.\mathsf{val}@0 = 1$, then

$$f(v).\mathsf{temp}@\alpha = \left(\mathsf{snd}f(v) - \frac{\ell}{k}\right) \cdot e^{-k\alpha} + \frac{\ell}{k} = \left((v.\mathsf{temp}@0) - \frac{\ell}{k}\right) \cdot e^{-k\alpha} + \frac{\ell}{k} = v.\mathsf{temp}@\alpha.$$

We turn to the semantics of the active thermostat ATHERM. In a model of this specification one has to keep track of (1) whether the heater is on or off, (2) the current temperature in the room, and (3) the goal temperature. The minimal set of these data is

$$U = \{\langle x, y, z\rangle \in \{0,1\} \times [0, \tfrac{\ell}{k}) \times (1, \tfrac{\ell}{k} - 1) \mid y < z - 1 \Rightarrow x = 1 \text{ and } y > z + 1 \Rightarrow x = 0\}.$$

The restriction in this definition deals with the states of adjustment, when the temperature y in the room is outside the region $[z-1, z+1]$ around the goal temperature z. The method interpretations on U are as follows.

$$\begin{array}{rcl} U & \xrightarrow{\mathsf{val}} & \{0,1\} \\ \langle x, y, z\rangle & \longmapsto & x \end{array} \qquad \begin{array}{rcl} U & \xrightarrow{\mathsf{temp}} & [0, \frac{\ell}{k}) \\ \langle x, y, z\rangle & \longmapsto & y \end{array}$$

$$\begin{array}{rcl} U & \xrightarrow{\mathsf{goal}} & (1, \frac{\ell}{k} - 1) \\ \langle x, y, z\rangle & \longmapsto & z \end{array} \qquad \begin{array}{rcl} U \times (1, \frac{\ell}{k} - 1) & \xrightarrow{\mathsf{set}} & U \\ (\langle x, y, z\rangle, a) & \longmapsto & \begin{cases} \langle 0, y, a\rangle & \text{if } y \geq a \\ \langle 1, y, a\rangle & \text{if } y < a \end{cases} \end{array}$$

The action $\mu \times \mathbb{R}_{\geq 0} \to U$ is more difficult. We first define, for a goal temperature $z \in (1, \frac{\ell}{k} - 1)$, a history function $h_z: \mathbb{R}_{\geq 0} \to [0, \frac{\ell}{k})$ describing the periodic oscillation of the temperature in the room around the goal temperature z, as function of time $\alpha \in \mathbb{R}_{\geq 0}$. Therefore we first need the times

$$\uparrow z \stackrel{\text{def}}{=} \frac{1}{k}\ln\left(\frac{\ell - k(z-1)}{\ell - k(z+1)}\right) \qquad \text{and} \qquad \downarrow z \stackrel{\text{def}}{=} \frac{1}{k}\ln\left(\frac{z+1}{z-1}\right)$$

that it takes for the temperature in the room to rise from $z-1$ to $z+1$, respectively to fall from $z+1$ to $z+1$. The periodicity of h_z is then $\uparrow(z) + \downarrow(z)$, through the definition:

$$h_z(x) = \begin{cases} ((z-1) - \frac{\ell}{k}) \cdot e^{-kx} + \frac{\ell}{k} & \text{if } x \in [0, \uparrow z) \\ (z+1) \cdot e^{-kx} & \text{if } x \in [\uparrow z, \uparrow z + \downarrow z) \\ h_z(x - n(\uparrow z + \downarrow z)) & \text{otherwise, where } n \in \mathbb{N} \text{ is least with } x \geq n(\uparrow z + \downarrow z). \end{cases}$$

Now we define the action $\mu : U \times \mathbb{R}_{\geq 0} \to U$ as follows. We first deal with the adjustment phases: if $y < z - 1$, then

$$\mu(\langle x, y, z\rangle, \alpha) = \begin{cases} \langle x, (y - \frac{\ell}{k}) \cdot e^{-k\alpha} + \frac{\ell}{k}, z\rangle & \text{if } \alpha < \frac{1}{k}\ln(\frac{\ell - ky}{\ell - k(z-1)}) \\ \mu(\langle x, z-1, z\rangle, \alpha - \frac{1}{k}\ln(\frac{\ell - ky}{\ell - k(z-1)})) & \text{otherwise.} \end{cases}$$

And if $y > z + 1$, then

$$\mu(\langle x, y, z\rangle, \alpha) = \begin{cases} \langle x, y \cdot e^{-k\alpha}, z\rangle & \text{if } \alpha < \frac{1}{k}\ln(\frac{y}{z+1}) \\ \mu(\langle x, z+1, z\rangle, \alpha - \frac{1}{k}\ln(\frac{y}{z+1})) & \text{otherwise.} \end{cases}$$

Finally, if we are in the "stability" phase $z - 1 \leq y \leq z + 1$, then we can use the history function h_z to define μ.

$$\begin{aligned} \mu(\langle 0, y, z\rangle, \alpha) &= \langle x, h_z(h_z^{-1}(y) + \alpha), z\rangle \qquad \text{where} \\ &\qquad h_z^{-1}(y) \in [\uparrow z, \uparrow z + \downarrow z) \text{ is unique with } h_z(h_z^{-1}(y)) = y, \\ &\qquad \text{and } x = 0 \text{ if the derivative } h_z'(h_z^{-1}(y) + \alpha) < 0, \text{ and } x = 1 \text{ else.} \\ \mu(\langle 1, y, z\rangle, \alpha) &= \langle x, h_z(h_z^{-1}(y) + \alpha), z\rangle \qquad \text{where} \\ &\qquad h_z^{-1}(y) \in [0, \uparrow z) \text{ is unique with } h_z(h_z^{-1}(y)) = y, \\ &\qquad \text{and } x = 0 \text{ if the derivative } h_z'(h_z^{-1}(y) + \alpha) < 0, \text{ and } x = 1 \text{ else.} \end{aligned}$$

It is laborious, but in essence straightforward, to check that U with this action is a model of the active thermostat specification ATHERM; and also that it is the terminal model: for an arbitrary model V there is a unique homomorphism $f : V \to U$ given by $f(v) = \langle v.\mathsf{val}@0, v.\mathsf{temp}@0, v.\mathsf{goal}@0\rangle$.

6 Final remarks

Terminal models play a special role in (coalgebraic) specification as minimal realizations in which all observationally indistinguishable states are identified. We have introduced (non-obvious) notions of model and of homomorphism of models for temporal coalgebraic specifications, and have shown in various examples that the resulting terminal models are the intended minimal models, thereby achieving the modest aim of this paper: to show that terminality applies in these situations as well.

We have not explained where the terminal models came from. We used the intuitive (and quite useful) heuristics that terminal models are "minimal realizations", i.e. consist of the minimal set of states needed for the specified behaviour. There is a more mathematical way to find these terminal models by following the recipe of [8]: first find the terminal model of operations only, and then carve out the appropriately universal submodel satisfying the assertions, using (temporal) mongruences. Due to lack of space, we only give a sketch: in a situation with attribute $X \longrightarrow A$, procedure $X \times B \longrightarrow X$ and monoid M, this terminal model has as carrier the function space $A^{(B \times M)^\star \times M}$ of "sampling observations". And for the second step, one uses the greatest "temporal mongruence" which is contained in the subset determined by the equations, where a temporal mongruence is a subset of the carrier set of a model, which is closed under the monoid action and under the procedure.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comp. Sci.*, 138(1):3–34, 1995.
2. M. Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems.* PhD thesis, Dep. of Automatic Control, Lund Inst. of Techn., 1994.
3. M. Barr and Ch. Wells. *Category Theory for Computing Science.* Prentice Hall, 1990.
4. H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification I: Equations and Initial Semantics.* Number 6 in EATCS Monographs. Springer, Berlin, 1985.
5. J.A. Goguen. Realization is universal. *Math. Syst. Theor.*, 6(4):359–374, 1973.
6. T.A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lect. Notes Comp. Sci., pages 226–251. Springer, Berlin, 1992.
7. M.W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, New York, 1974.
8. B. Jacobs. Mongruences and cofree coalgebras. In V.S. Alagar and M. Nivat, editors, *Algebraic Methods and Software Technology*, number 936 in Lect. Notes Comp. Sci., pages 245–260. Springer, Berlin, 1995.
9. B. Jacobs. Inheritance and cofree constructions. In P. Cointe, editor, *European Conference on Object-Oriented Programming*, number 1098 in Lect. Notes Comp. Sci., pages 210–231. Springer, Berlin, 1996.
10. B. Jacobs. Objects and classes, co-algebraically. In B. Freitag, C.B. Jones, C. Lengauer, and H.-J. Schek, editors, *Object-Orientation with Parallelism and Persistence.* Kluwer Acad. Publ., 1996, to appear.
11. R.E. Kalman, P.L. Falb, and M.A. Arbib. *Topics in Mathematical System Theory.* McGraw-Hill Int. Series in Pure & Appl. Math., 1969.
12. R. Koymans, R. Kuiper, and E. Zijlstra. Paradigms for real-time systems. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 331 in Lect. Notes Comp. Sci., pages 159–174. Springer, Berlin, 1988.
13. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, number 736 in Lect. Notes Comp. Sci., pages 149–178. Springer, Berlin, 1993.
14. X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, number 600 in Lect. Notes Comp. Sci., pages 549–572. Springer, Berlin, 1992.
15. A. Pnueli. Development of hybrid systems. In H. Langmaack, W.P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 863 in Lect. Notes Comp. Sci., pages 159–174. Springer, Berlin, 1994.
16. H. Reichel. An approach to object semantics based on terminal co-algebras. *Math. Struct. Comp. Sci.*, 5:129–152, 1995.
17. J. Rutten and D. Turi. Initial algebra and final coalgebra semantics for concurrency. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency*, number 803 in Lect. Notes Comp. Sci., pages 530–582. Springer, Berlin, 1994.
18. P. Wegner. The object-oriented classification paradigm. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 479–560. The MIT Press series in computer systems, 1987.
19. W. Yi. Real-time behaviour of asynchronous agents. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR '90. Theory of Concurrency: unification and extension*, number 458 in Lect. Notes Comp. Sci., pages 502–520. Springer, Berlin, 1990.