Functional Specification of Real-Time and Hybrid Systems *

Olaf Müller Peter Scholz
Institut für Informatik, Technische Universität München
D-80290 München, Germany
E-mail: {mueller,scholzp}@informatik.tu-muenchen.de

Abstract. Functional specifications have been used to specify and verify designs of a number of reactive, discrete systems. In this paper we extend this specification style to deal with real-time and hybrid systems. As mathematical foundation we employ Banach's fixed point theory in metric spaces. The goal is to show that the theory used for discrete functional specifications smoothly carries over to real-time and hybrid systems. An example of a thermostat specification illustrates the method.

1 Introduction

Hybrid systems are dynamical systems consisting of both discrete and continuous components. They are used to model the behavior of embedded real-time systems in a physical environment. Recently, a number of description and specification languages for reactive and/or real-time systems together with their proposed methodology for analysis, verification, and refinement were extended to deal with hybrid systems. For example, for model checking purposes a theory of hybrid automata has been developed [ACH+95], TLA has been extended to TLA+ [Lam93], I/O Automata have been extended to describe hybrid systems [LSVW95], and there are many other [GNRR93, AKNS95] hybrid description techniques.

In this paper we extend the formalism of *functional specification* [BDD+93, Bro93] to deal with real-time and hybrid systems. Functional specifications describe the behavior of a system as a network of functions, where every function processes infinite streams of incoming messages and yields infinite streams of outgoing messages. In the discrete setting, several approaches have been taken to give functional specifications a semantics:

- In [Bro93] domain theory is used to develop a semantic model for discrete stream processing functions together with a tailored refinement methodology.
- In [GS96] metric spaces are employed to give a semantics for functionally specified, discrete mobile data-flow networks.

We follow the second approach and extend the static parts of [GS96] to a description and specification method for hybrid systems. Our goal is to show that only

* This work is partially sponsored by the German Federal Ministry of Education and Research (BMBF) as part of the compound project "KorSys" and by BMW (Bayerische Motoren Werke AG).

slight modifications must be carried through, so that the whole theory smoothly carries over to the hybrid world.

The paper is organized as follows: Section 2 introduces stream processing functions and relates them to the corresponding notions in the theory of metric spaces. In Section 3 composition operators are defined that are used to build networks out of single functions. In particular, the mathematical foundation of the feedback operator is presented. Finally, Section 4 illustrates the specification method with the simple example of a thermostat.

2 Specification with Stream Processing Functions

We regard a distributed system as a network of components that exchange messages via directed channels. On every input or output channel messages are received from, or sent to, the environment. Therefore, every channel reflects an input or output communication history of the system. The system itself is described by a set of functions, where each function processes input histories and produces output histories according to its specification. To describe underspecification or nondeterminism we use sets of functions instead of single functions.

2.1 Dense Communication Histories

Communication histories of discrete systems can be modeled by sequences of messages, i.e., functions of type $\mathbb{N} \to M$, where M denotes the set of all messages [Bro93, BDD$^+$93]. For hybrid systems this model has to be extended to incorporate real time. One possibility is to add real time stamps. In the literature this is known as *sampling* semantics [MP93]. Here, instead, we develop a *super dense* semantics and therefore introduce real time or *dense* streams.

Let M be the (potentially infinite) set of all messages. A *dense stream* x over a set M is represented by a total function $x : \mathbb{R}_+ \to M$, where $\mathbb{R}_+$ denotes the set of all non-negative real numbers. Since we describe reactive systems, which continuously respond to stimuli from the environment, time never halts, and we use $\mathbb{R}_+$ as the time scale instead of time intervals. The set of all dense streams is denoted by $M^{\mathbb{R}_+}$. For every dense stream x we abbreviate the restriction $x|_{[0,t]}$ by $x{\downarrow}t$.

In order to motivate the usefulness of this definition we have adapted the example of a thermostat from [ACH$^+$95], where it is presented by means of hybrid automata.

Example 1 Dense Stream. The temperature of a room in a cool environment can be modeled by a dense stream x. We assume that without the presence of any heater, the temperature decreases according to the exponential function $x(t) = \Theta e^{-Kt}$, where t denotes the time, Θ the initial temperature, and K is a positive constant determined by the room.

A mathematical treatment of functional specifications requires dealing with feedback loops. In the discrete case, dealing with streams of type $\mathbb{N} \to M$, the semantics of such loops has been successfully described as least fixed points of functions over domains [Bro93, BDD+93]. The underlying mathematical model is Scott's domain theory [SG90, Win93]. Fixed points of stream processing functions over dense streams, however, are more naturally and elegantly described by the fixed point theory of Banach. It is based upon the mathematical background of metric spaces. In order to specify loops of stream processing functions in Section 3, we therefore introduce the main concepts of metric space theory.

Definition 1 Metric Space. A *metric space* is a pair (D, d) consisting of a nonempty set D and a mapping $d : D \times D \to \mathbb{R}$, called a *metric* or a *distance,* which has the following properties:

(**1**) $\forall x, y \in D: \quad d(x, y) = 0 \quad \Leftrightarrow \quad x = y$
(**2**) $\forall x, y \in D: \quad d(x, y) = d(y, x)$
(**3**) $\forall x, y, z \in D : d(x, y) \leq d(x, z) + d(z, y)$.

We need a metric for dense streams, which is defined in the sequel.

Definition 2 The Baire Metric of Streams. The Baire metric space of dense streams $(M^{\mathbb{R}_+}, d)$ is for all $x, y \in M^{\mathbb{R}_+}$ defined as follows (see [Eng77]):

$$d(x, y) = inf\{2^{-t} \mid t \in \mathbb{R}_+ \wedge x{\downarrow}t = y{\downarrow}t\}.$$

From this definition a metric $d^{(n)}$ for n-tuples of streams $(M^{\mathbb{R}_+})^n$ can be easily derived. Let $n \in \mathbb{N}$ and $x, y \in (M^{\mathbb{R}_+})^n$ then $d^{(n)}(x, y)$ is defined as

$$d^{(n)}(x, y) = max\{d(x_i, y_i) \mid 1 \leq i \leq n\}.$$

A metric space (D, d) is called *complete* whenever each Cauchy sequence converges to an element of D [Eng77]. The Baire metric space on stream tuples $((M^{\mathbb{R}_+})^n, d^{(n)})$, we consider in this paper, is complete [Eng77]. Complete metric spaces are a presupposition for Banach's fixed point theorem. This theorem, which will be explained later on, guarantees — under certain assumptions — the existence of a unique fixed point of loops in functional specifications.

2.2 Stream Processing Functions

Components of real time or hybrid systems can be functionally specified by stream processing functions over dense streams. First ideas in this area come from system theory [MT75]. Components are connected by directed channels to form a network. Each channel links an *input port* to an *output port.* A (m, n)-ary stream processing function with m input and n output ports is a function f with

$$f : (M_1^{\mathbb{R}_+})^m \to (M_2^{\mathbb{R}_+})^n$$

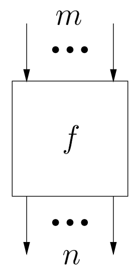

Fig. 1. Stream Processing Function

where M_1 and M_2 represent two (not necessarily different) sets of messages. The graphic notation of f is pictured in Fig. 1. If we want to express some kind of nondeterminism we describe components by a set of stream processing functions rather than by a single function.

Our operational understanding that stream processing functions model interacting components leads to a basic requirement for them. An interactive component is not capable to take back an output message that it has already emitted. This requirement can be fulfilled by a certain kind of stream processing functions, namely behaviors.

A stream processing function is said to be a *behavior* if its input until time t completely determines its output until time t. It is said to be a *delayed behavior* if its input until time t completely determines its output until time $t+\delta$ for $\delta > 0$. In other words, a delayed behavior imposes a delay of at least an arbitrarily small real value between input and output. Here, δ denotes the delay of f. It is quite realistic to assume components to be delayed because reactive systems always need a certain time to react. Instantaneous reactions, however, can be expressed by (non-delayed) behaviors.

Definition 3 (Delayed) Behavior. A (m, n)-ary stream processing function f is called a *behavior* if

$$\forall x, y \in (M^{\mathbb{R}_+})^m,\ t \in \mathbb{R}_+ : x{\downarrow}t = y{\downarrow}t \Rightarrow f(x){\downarrow}t = f(y){\downarrow}t$$

and *a delayed behavior (with delay $\delta > 0$)* if

$$\forall x, y \in (M^{\mathbb{R}_+})^m,\ t \in \mathbb{R}_+ : x{\downarrow}t = y{\downarrow}t \Rightarrow f(x){\downarrow}(t+\delta) = f(y){\downarrow}(t+\delta).$$

Note that the operator $\downarrow$ is overloaded to stream tuples in a point-wise style, i.e., $x{\downarrow}t$ for a stream tuple $x \in (M^{\mathbb{R}_+})^m$ denotes the tuple we get by applying $\downarrow t$ to each component of x.

The equivalent property in Scott's theory is monotonicity. From a theorem by Knaster and Tarski it is well-known that monotonic functions over complete partial orders have a least fixed point [Win93].

We model specifications by sets of (delayed) behaviors. They can be composed into networks of functions, which themselves behave as (delayed) behaviors. For this purpose, we will introduce three composition operators in the next section. For one of them, the feedback operator, the existence of a unique fixed point of the feedback loop is guaranteed only for *delayed* behaviors. To prove this formally we introduce a notion corresponding to delayed behaviors in metric space theory.

Definition 4 Lipschitz Functions. Let (D_1, d_1) and (D_2, d_2) be metric spaces and let $f : D_1 \to D_2$ be a function. We call f a *Lipschitz function* if there is a constant $c \geq 0$ such that the following condition is satisfied for all $x, y \in D_1$:

$$d_2(f(x), f(y)) \leq c \cdot d_1(x, y).$$

The Lipschitz constant $Lip(f)$ of a Lipschitz function f is denoted by the infimum of all c that fulfill the above mentioned inequation. If $Lip(f) \leq 1$ we call f *non-expansive*. If $Lip(f) < 1$ we call f *contractive*.

The following theorem relates the notions of behaviors and delayed behaviors to non-expansiveness and contractivity. Whereas the first ones have a operational justification, the latter ones represent their transfer to metric space theory and will be used as a requirement for Banach's fixed point theorem.

Theorem 5. *A stream processing function is a delayed behavior iff it is contractive with respect to the metric of stream tuples. A stream processing function is a behavior iff it is non-expansive with respect to the metric of stream tuples.*

Proof. We prove the first statement of the theorem. First, we prove the only-if-direction. Suppose that $d^{(m)}(x, y) = 2^{-t_0}$ and that f is a delayed behavior with delay δ. $d^{(m)}(x, y) = 2^{-t_0}$ implies that $x \downarrow t_0 = y \downarrow t_0$. Therefore, $f(x) \downarrow (t_0 + \delta) = f(y) \downarrow (t_0 + \delta)$. Finally, we get $inf\{2^{-t} \mid t \in I\!R_+ \wedge f(x) \downarrow t = f(y) \downarrow t\} \leq 2^{-(t_0+\delta)} = 2^{-\delta} \cdot d^{(m)}(x, y)$. Since $2^{-\delta} < 1$ for all $\delta > 0$, f is contractive.
Now, we prove the if-direction. Suppose that $d^{(m)}(x, y) = 2^{-t_1}$, $d^{(n)}(f(x), f(y)) = 2^{-t_2}$, and that f is contractive, i.e., $\exists c < 1 : \forall x, y : d^{(n)}(f(x), f(y)) \leq c \cdot d^{(m)}(x, y)$. As $c < 1$ we can find a positive δ such that $2^{-\delta} = c$. Then $2^{t_1 - t_2} \leq c = 2^{-\delta}$. This implies because of the monotonicity of the logarithmic function that $t_1 + \delta \leq t_2$. As a consequence we get $x \downarrow t_1 = y \downarrow t_1 \Rightarrow f(x) \downarrow (t_1 + \delta) = f(y) \downarrow (t_1 + \delta)$ because $f(x) \downarrow t_2 = f(y) \downarrow t_2$ implies $f(x) \downarrow (t_1 + \delta) = f(y) \downarrow (t_1 + \delta)$. In other words, f is a delayed behavior. The second equivalence can be proven accordingly.

3 Composition Operators

The definition of networks is the main structuring principle on the functional specification level. There is no (semantical) difference in principle between a single component and a network of components. A network can be defined either by recursive equations or by special composition operators. We choose the second alternative and consider three basic composition operators, namely *sequential/parallel composition* and *feedback*.

In our functional specification technique, networks of components can be represented by directed graphs, where the nodes represent components and the edges represent point-to-point, directed communication channels (see, for instance, Fig. 2).

3.1 Sequential Composition

Sequential composition is simply defined by functional composition of two stream processing functions. The graphic representation of this composition is pictured in Fig. 2.

Definition 6 Sequential Composition. Let f and g be (m, n)-ary and (n, k)-ary stream processing functions, respectively. Then $f \circ g$ is the (m, k)-ary stream processing function defined by $(f \circ g)(x) = g(f(x))$.

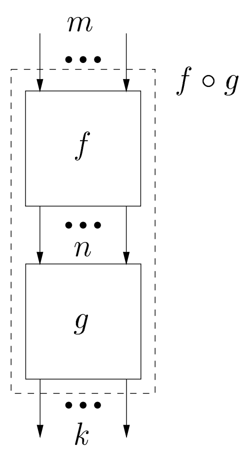

Fig. 2. Sequential Composition

The following theorem and corollary depict important properties of the sequential composition:

Theorem 7. *The sequential composition of two Lipschitz functions $f : D_1 \to D_2$ and $g : D_2 \to D_3$ is a Lipschitz function with constant $Lip(f) \cdot Lip(g)$.*

Proof.

$$\begin{aligned} d_3(g(f(x_1)), g(f(x_2))) &\leq Lip(g) \cdot d_2(f(x_1), f(x_2)) \quad (1) \\ &\leq Lip(g) \cdot Lip(f) \cdot d_1(x_1, x_2). \quad (2) \end{aligned}$$

Corollary 8. *The sequential composition of two behaviors is a behavior. The sequential composition of two delayed behaviors with delays δ_1 and δ_2, respectively, is a delayed behavior with delay $\delta_1 + \delta_2$. The sequential composition of a behavior and a delayed behavior is a delayed behavior.*

Due to the above theorem, the proof of this corollary is obvious.

3.2 Parallel Composition

The parallel composition is defined intuitively. Sticking two components orthogonally together yields a component which input/output ports consists of all input/output ports of the composed components (see Fig. 3). Formally:

Definition 9 Parallel Composition. Let f and g be (m, n)-ary and (k, l)-ary stream processing functions. Then $f \| g$ is the $(m+k, n+l)$-ary stream processing function defined by

$$(f \| g)(x_1, \ldots, x_{m+k}) = (f(x_1, \ldots, x_m), g(x_{m+1}, \ldots, x_{m+k})).$$

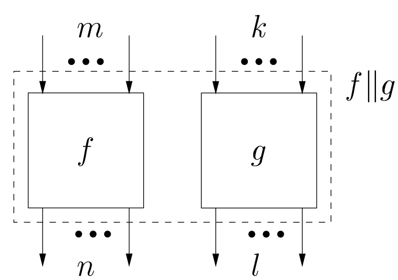

Fig. 3. Parallel Composition

As for the sequential composition, an equivalent property can also be formulated for the parallel composition:

Theorem 10. *The parallel composition of two behaviors is a behavior. The parallel composition of two delayed behaviors with delays δ_1 and δ_2, respectively, is a delayed behavior with delay $min(\delta_1, \delta_2)$. The parallel composition of a behavior and a delayed behavior is a behavior.*

Proof. We prove the second statement of the theorem. Let f be a (m, n)-ary delayed behavior with delay δ_1 and g be a (k, l)-ary delayed behavior with delay δ_2. Without loss of generality we assume that $\delta_1 < \delta_2$. Let $x, y \in (M^{\mathbb{R}_+})^k$, then $g(x)\downarrow(t + \delta_2) = g(y)\downarrow(t + \delta_2)$ implies that $g(x)\downarrow(t + \delta_1) = g(y)\downarrow(t + \delta_1)$. The other statements can be proven accordingly.

Note that the sequential composition of a behavior and a delayed behavior is a delayed behavior, whereas the parallel composition of a behavior and a delayed behavior is "only" a behavior.

3.3 Feedback Operator

Systems described by functional specifications may contain loops. In the graphic notation, this is denoted by circular graphs (Fig. 4). The feedback operator feeds k output channels back to k input channels of a $(m+k, n+k)$-ary delayed behavior.

Definition 11 Feedback Operator. Let $f : (M_1^{\mathbb{R}_+})^m \times (M^{\mathbb{R}_+})^k \to (M_2^{\mathbb{R}_+})^n \times (M^{\mathbb{R}_+})^k$ be a $(m+k, n+k)$-ary delayed behavior. Then $\mu^k f$ is a (m, n)-ary delayed behavior such that the value $(z_1, \ldots, z_n)$ of $(\mu^k f)(x_1, \ldots, x_m)$ is calculated as follows:

$$(z_1, \ldots, z_n, y_1, \ldots, y_k) = f(x_1, \ldots, x_m, y_1, \ldots, y_k)$$

where $(y_1, \ldots, y_k)$ is the solution of the equation

$$(y_1, \ldots, y_k) = g_{(x_1, \ldots, x_m)}(y_1, \ldots, y_k).$$

Here $g_{(x_1, \ldots, x_m)}$ is defined as a (k, k)-ary delayed behavior:

$$g_{(x_1, \ldots, x_m)}(y_1, \ldots, y_k) = \pi_{n+1, n+k}(f(x_1, \ldots, x_m, y_1, \ldots, y_k))$$

where $\pi_{n+1, n+k}$ denotes the projection on the last k ports.

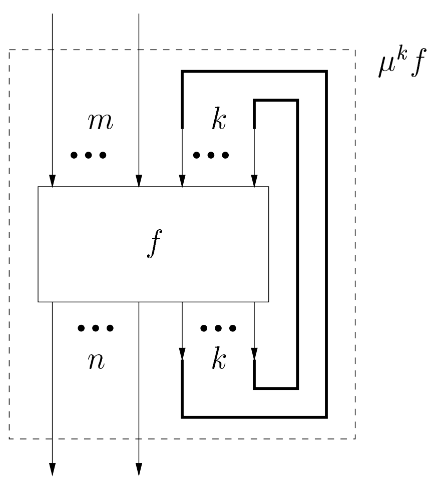

Fig. 4. Feedback Operator

The central issue of our contribution is that the fixed point operator is well-defined, i.e., that the unique solution of

$$(y_1, \ldots, y_k) = g_{(x_1,\ldots,x_m)}(y_1, \ldots, y_k)$$

exists. The existence of this fixed point is guaranteed by Banach's fixed point theorem:

Theorem 12 Banach's Fixed Point Theorem. *Let (D, d) be a complete metric space and $f : D \rightarrow D$ a contractive function. Then there exists an $x \in D$, such that the following holds:*

(**1**) $x = f(x)$ *(x is a fixed point of f)*
(**2**) $\forall y \in D : \; y = f(y) \Rightarrow y = x$ *(x is unique)*
(**3**) $\forall z \in D : \; x = lim_{n\rightarrow\infty} f^n(z)$ *where*

$$f^0(z) = z$$
$$f^{n+1}(z) = f(f^n(z))$$

Proof. For instance, see [Sut75].

In the context of this paper, we can apply Banach's theorem in the following way. First of all, the metric space $((M^{\mathbb{R}_+})^k, d^{(k)})$ is complete. Secondly, f is a $(m + k, n + k)$-ary delayed behavior and therefore contractive. Remember that f need not to be a basic stream processing function, but can also be a composed, delayed behavior. Moreover, also $g_{(x_1,\ldots,x_m)} : (M^{\mathbb{R}_+})^k \rightarrow (M^{\mathbb{R}_+})^k$ is by definition a contractive function. Altogether, all assumptions of Banach's fixed point theorem are fulfilled and the existence of a unique fixed point $(y_1, \ldots, y_k)$ of $g_{(x_1,\ldots,x_m)}$ is ensured. Hence, the feedback part of every delayed behavior has a unique fixed point.

Banach's fixed point theorem is the counterpart of Knaster/Tarski's fixed point theorem in the theory of metric spaces. However, note that Knaster/Tarski's theorem only guarantees the existence of a *least* fixed point, i.e., that potentially more than one fixed point can exist. In contrast, Banach's fixed point theorem guarantees the existence of a unique fixed point.

Again it is a straightforward proof to show that the feedback $\mu^k f$ is a delayed behavior, provided that f is a delayed behavior.

4 Example

In this section we give a functional specification of a thermostat, a simple hybrid system used as an introductory example in [ACH$^+$95]. The temperature of a room is controlled by a thermostat, which continuously senses the temperature and turns a heater on and off. The temperature is governed by differential equations. When the heater is off, the temperature $Temp$ of the environment, denoted by the dense stream x, decreases according to the function $x(t) = \Theta e^{-Kt}$ (see Example 1). When the heater is on, the temperature of the environment follows the function $x(t) = \Theta e^{-Kt} + h(1 - e^{-Kt})$, where h is a constant that depends on the power of the heater, Θ is the initial temperature of the room, and K is a constant determined by the environment. K can be considered to be direct proportional to the geometric size of the room. We wish to keep the temperature between min and max degrees and turn the heater on and off accordingly.

4.1 Thermostat as Open System

The controlling part of the resulting system for this informal description is shown in Fig. 5. The system consists of the two components *Control* and *Heater*. The first one is described by a function f_C of type

$$f_C : Temp^{\mathbb{R}_+} \rightarrow \{\mathsf{on}, \mathsf{off}\}^{\mathbb{R}_+}$$

that produces signals off or on, if the incoming stream of temperature signals overshoots max or undershoots min, respectively. These signals serve as an input stream for the *Heater* f_H:

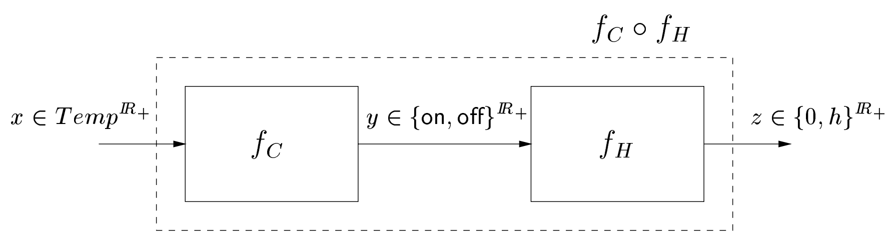

Fig. 5. Thermostat Modeled as Open System

$$f_H : \{\mathsf{on}, \mathsf{off}\}^{\mathbb{R}_+} \rightarrow \{0, h\}^{\mathbb{R}_+}$$

that produces the corresponding heating power, which can be 0 or h. Note that we model only the heating power of the heater, but not the resulting absolute temperature. The temperature of the room is regarded as part of the system's environment. This is different from [ACH$^+$95], where the temperature is an inherent part of the system description. Therefore, the environment is there modeled as part of the system.

In fact, the model of hybrid automata does not emphasize on an interface concept to the environment, so that [ACH$^+$95] describes merely closed systems without dividing the overall specification into system and environment. The advantage of our approach is its modularity, which allows us to separate the environment from the system specification. This is one of the essential issues of our approach. The application of our functional specification method to the thermostat example shows that indeed only the environment behaves continuously. The system itself, i.e., *Controller* and *Heater* behave as value-discrete components. They produce signals on, off, 0, and h. The environment, however, is characterized by the temperature, which is denoted by a real-valued ($Temp$) stream. In the sequel, we give the precise specifications of the components *Control* and *Heater*. First of all, we define *Control*:

$$f_C(x) = y$$

where the output stream $y \in \{\mathsf{on}, \mathsf{off}\}^{\mathbb{R}_+}$ is for all $t \in \mathbb{R}_+$ defined as follows:

$$\begin{array}{ll} x(t) \leq \mathsf{min} & \Rightarrow y(t + \delta_C) = \mathsf{on} \\ x(t) \geq \mathsf{max} & \Rightarrow y(t + \delta_C) = \mathsf{off} \\ \mathsf{min} < x(t) < \mathsf{max} & \Rightarrow y(t + \delta_C) = y(t). \end{array}$$

Here $\delta_C > 0$ denotes the delay of the component *Control*. However, this specification leaves the value $y(t)$ in the interval $[0, \delta_C)$ unspecified. We can abolish this under-specification by simply defining $y(t) = \mathsf{off}$ in this interval. Now, we specify the *Heater*:

$$f_H(y) = z$$

where the output stream $z \in \{0, h\}^{\mathbb{R}_+}$ is for all $t \in \mathbb{R}_+$ defined as follows:

$$\begin{array}{l} y(t) = \mathsf{off} \Rightarrow z(t + \delta_H) = 0 \\ y(t) = \mathsf{on} \Rightarrow z(t + \delta_H) = h. \end{array}$$

Again, to avoid under-specification, we define $z(t) = 0$ for $t \in [0, \delta_H)$. The whole thermostat can then be described using the sequential composition

$$f_C \circ f_H.$$

This function has delay $\delta_C + \delta_H$ according to Corollary 1.

4.2 Thermostat as Closed System

To model the continuous part of the specification, we add the environment to it, yielding a closed system (Fig. 6):

$$f_E : \{0, h\}^{\mathbb{R}_+} \rightarrow Temp^{\mathbb{R}_+}$$

Env is specified as a component that cools the temperature down according to the exponential function Θe^{-Kt} (see also Example 1), if the *Heater* is off. When it is on, the temperature follows the function $\Theta e^{-Kt} + h(1 - e^{-Kt})$. We combine these two functions to one function $x(t) = \Theta e^{-Kt} + z(t) \cdot (1 - e^{-Kt})$ and get:

$$f_E(z) = x$$

where the output stream $x \in Temp^{\mathbb{R}_+}$ is defined by the differential equation:

$$x'(t) = z(t) - K\Theta x(t)$$

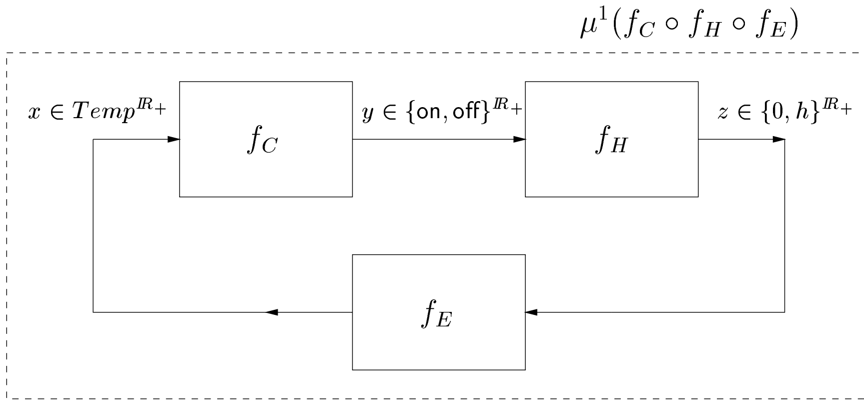

Fig. 6. Thermostat Modeled as Closed System

where $x'(t)$ denotes the first differentiation of $x(t)$. f_E and $f_C \circ f_H$ form a closed system in the shape of a feedback:

$$\mu^1(f_C \circ f_H \circ f_E).$$

This function is well-defined, as the occurring fixed point is uniquely determined according to our theory in Section 3: as $f_C \circ f_H$ is contractive with delay $\delta_C + \delta_H$, $f_C \circ f_H \circ f_E$ is contractive according to Corollary 1, even if f_E has no delay at all. Therefore Banach's fixed point theorem can be applied.

5 Conclusion and Further Work

We have shown that the specification formalism of discrete timed stream processing functions can easily be extended to deal with real-time and hybrid systems. We could give functional specifications with feedback a semantical foundation by introducing the concept of delayed behaviors that allows us to employ Banach's fixed point theorem. Characteristic of our approach is that our functional model naturally reflects the physical and conceptual structure of the system and its environment. In particular, it is possible to distinguish clearly between system and environment. In the thermostat example this structural clarity has been documented. Furthermore, we have the impression that the concept of well-known mathematical functions leads to a simple and clear specification style. Moreover, this generic approach fulfills the major requirement for any reasonable modeling formalism, namely modularity. In the discrete case a verification methodology by (structural, behavioral, and interface) refinements is well studied and understood. It is work in progress to carry over these results to our setting. It would also be interesting to analyze another type of streams as functions of type $\mathbb{N} \to M \times \mathbb{R}$, yielding a sampling semantics.

Acknowledgment

Thanks are owed to Manfred Broy who provided first ideas concerning both dense streams and behaviors. The authors have benefited from many discussion with Ketil Stølen and from the technical report by Radu Grosu and Ketil Stølen.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[AKNS95] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry. *Hybrid Systems II*, volume 999. Springer Verlag, 1995. Lecture Notes in Computer Science.

[BDD+93] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. The Design of Distributed Systems: An Introduction to Focus — Revised Version. Technical Report TUM-I9202-2, Technische Universität München, Fakultät für Informatik, 80290 München, Germany, 1993.

[Bro93] M. Broy. Interaction Refinement – The Easy Way. In M. Broy, editor, *Program Design Calculi*, volume 118 of *NATO ASI Series F: Computer and System Sciences*. Springer, 1993.

[Eng77] R. Engelking. *General Topology*. PWN - Polish Scientific Publishers, 1977.

[GNRR93] R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel. *Hybrid Systems*, volume 736. Springer Verlag, 1993. Lecture Notes in Computer Science.

[GS96] R. Grosu and K. Stølen. A Model for Mobile Point-to-Point Dataflow Networks without Channel Sharing. In *Proc. of the 5th International Conference on Algebraic Methodology and Software Technology AMAST'96, Munich*, volume 1101 of *Lecture Notes in Computer Science*, pages 513–519, 1996. Also available as Technical Report TUM-I9527, Technische Universität München.

[Lam93] L. Lamport. Hybrid Systems in TLA+. In R.L. Grossman et al., editor, *[GNRR93]*, 1993.

[LSVW95] N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. Technical Report CS-R9578, CWI, Computer Science Department, Amsterdam, 1995. Available under `http://www.cs.kun.nl/~fvaan/`.

[MP93] Z. Manna and A. Pnueli. Verifying Hybrid Systems. In Grossman et al., editor, *[GNRR93]*, 1993.

[MT75] M.D. Mesarovic and Y. Takahara. *General Systems Theory: Mathematical Foundations*, volume 113. Academic Press, 1975. Mathematics in Science and Engineering.

[SG90] D. Scott and C. Gunter. Semantic Domains and Denotational Semantics. In *Handbook of Theoretical Computer Science*, chapter 12, pages 633 – 674. Elsevier Science Publisher, 1990.

[Sut75] W. A. Sutherland. *Introduction to metric and topological spaces*. Claredon Press - Oxford, 1975.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, 1993.

This article was processed using the LaTeX macro package with LLNCS style