# Dynamic Sampling for FOCUS Hybrid Components

Alarico Campetelli

***Abstract*—The combination of embedded systems with physical components, termed Cyber-Physical Systems (CPSs), poses important challenges in the design, modelling, simulation and verification of systems, also why they combine discrete software elaborations with analogues time evolutions and different engineering disciplines. In this domain, the software part is increasing the dimension and has a more important role. An important formalism for describing continuous and discrete systems is the hybrid system formalism. We present our approach based on the FOCUS modelling theory to model hybrid systems, which have discrete transitions and continuous differential equations. The modular and logical structure of FOCUS components combined with hybrid systems improves the support for the modelling of CPSs. Anyway, a fully continuous simulation may be too complex and not represent the final hardware where the model will be deployed. Therefore, we propose two sampling techniques to transform the continuous time in discrete steps, in which the length of the sampling period it is dynamically changed.**

***Index Terms*—Hybrid systems, embedded systems, model-based development, sampling, hybrid systems simulation.**

## I. Introduction

Nowadays, safety-critical embedded systems are in use in vehicles, machines aircraft or medical instruments. In these domains, the combination of embedded systems with physical components determines the necessity of a suitable design that unite continuous and discrete behaviour. In fact, software components operate in discrete program steps, meanwhile the physical components evolve over time intervals following physical constraints. An important formalism for describing CPS from the perspective of computer scientists is known as hybrid automaton [1]. In some cases, specialised methodologies or languages are introduced to deal with CPSs, where the already existing techniques could be suitable to represent them, especially after an extension or adaptation to cover some special domain features.

FOCUS [2] is a modelling theory for the formal specification of distributed, discrete-event systems. It forms the basis foundation for our work to extend the support of the modelling also to hybrid automata. FOCUS defines a hierarchical and interconnected net of components, each with a typed i/o interface, as depicted in the Fig. 1.

Manuscript received May 9, 2013; revised July 2, 2013. This work was partially funded by the German Federal Ministry of Education and Research (BMBF), grant ``SPES XT, 01IS12005M". The responsibility for this article lies with the authors.

Alarico Campetelli is with the Technische Universität München, Institut für Informatik, Garching bei Müunchen, 85748, Germany (e-mail: campetel@in.tum.de).

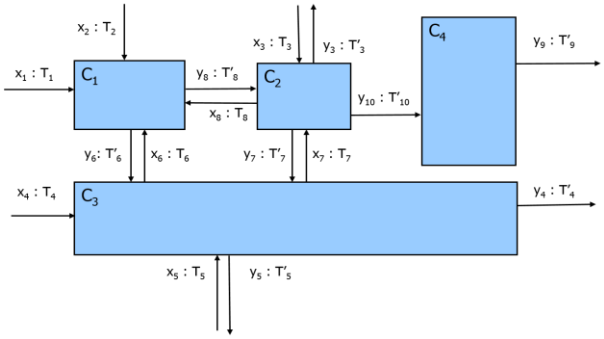


Fig. 1. An example of FOCUS component architecture.

The internal behaviour of each component can be implemented with different formalisms, as for instance functional specifications or finite state machines.

We present not only the formalization of the modelling theory for hybrid systems, but also aspects that are related to tool support and execution of them. Our models represent CPSs with a FOCUS on the software behaviour. A full continuous simulation may be too complex and anyway may not represent the final hardware, where the model will be deployed, that is basically a discrete component. We propose sampling techniques to transform the continuous time in discrete steps. The term sampling in mathematics means the transformation of an analogue signal in a digital one. We used sampling over continuous time, whereas some approaches do the sampling of the state space. Our ideas are mainly based on the work of Petreczky *et al.* [3], where is presented a static sampling over the continuous time. It is static because the discrete time step remains constant, that is usually called "equidistant sampling". Instead, we introduce two new solutions based on dynamic sampling, dynamic it is referred to the discrete time step, which is variable during the simulation. Our algorithms make the step shorter or longer, according to the values that are reached from the continuous variables or their slope. We think that such variations can produce a better simulation with respect to static sampling, because we can select important value intervals for each variable, from the requirements, and establish a corresponding length for the sampling period. On the other hand analysing the slope of the values of the variables and use a shorter period when the slope changes too repentantly can permit to better follow the variation of the variable producing more values in the same time, so in some sense increasing the precision of the simulation.

## II. Focus Hybrid Components

The FOCUS approach covers the specification of distributed, discrete systems with communication histories or traces represented by infinite sequences of messages. As consequence, FOCUS modelling approach has the time represented with a discrete granularity. This can represent a

DOI: 10.7763/IJMO.2013.V3.308

limiting restriction for reactive systems, which interact in continuous real-time with physical components and their environment. Consequently, the system has to have continuous data types and to be able to provide an instrument to define the continuous evolution of such data types. Considering the discrete character of FOCUS models and the continuous dynamic of the hybrid automata, we define *hybrid components* based on dense streams [4]. The internal behavior of each hybrid component is implemented by a special version of hybrid automaton [2]. The channels are described with stream functions over continuous time.

**Definition 1 (Hybrid stream):** Let M be the set of all messages (potentially infinite) and M* a finite sequence of messages, a hybrid discrete stream x over M is described by the following total function:

$$x : \mathsf{R}_+ \to M^*$$

where as a hybrid continuous stream $y$ over a set N of messages is:

$$y : \mathsf{R}_+ \to N$$

We assign to each channel a data type by a function $t : c \to T$, where $c$ is a channel and $T$ is a set of types $\tau \in T$. $M$ is the universe of all messages:

$$M = \bigcup \{\tau : \tau \in T\}$$

We define now a hybrid component based on the definition of hybrid automata and our notion of component in FOCUS .

**Definition 2 (I/O Hybrid Component):** An I/O hybrid component is a tuple

$$H = (\Sigma, Var, Init, I, O, Dom, E, f, G, R)$$

1) A state space $\sum$ = (Q × V) where Q is a set of discrete states Q = {q1, q2, …} and V a set of continuous states $V \subseteq (M_1^*)^{\mathsf{R}_+} \times \ldots \times (M_l^*)^{\mathsf{R}_+} \times (M_{l+1})^{\mathsf{R}_+} \times \ldots \times (M_n)^{\mathsf{R}_+}$ to each element of $V$ corresponds a hybrid stream as specified in Definition 1, that is associated to a variable in the set Var.

2) A set of initial states $Init \subseteq \Sigma$.

3) A set of input $I \subseteq V$ and output $O \subseteq V$ state space, respectively for input and output channels, whereas $Int \subseteq V$ are the internal continuous state space and $Int_d \subseteq V$ the internal discrete state space both with no channels associated, any variable in Var can only be in one of these set. $O_d \subseteq O$ and $O_c \subseteq O$, with $O_c \cap O_d = \varnothing$ are respectively the continuous and discrete output state space, we denote with $W = Int \cup O_c$ the set of internal and output state spaces.

4) A domain function $Dom : Q \to \mathsf{P}(V)$.

A set of edges $E \subseteq Q \times Q$ that represent the discrete state transition.

5) A vector field function $f : Q \times V \to W$.

6) A guard condition function $G : E \to \mathsf{P}(V)$.

7) A reset map function $R : E \times V \to \mathsf{P}(W \cup O_d)$ that represent with the vector field the continuous dynamics.

The set $V$ of continuous state space is subdivided in internal variables, output and input subsets. Continuous states are defined considering each discrete state plus the *Dom* function, when the system is in one of these states the system evolves in the continuous space according to such function. From an initial state $(q_0, v_0)$ the evolution of the system is guided from the differential equation over $w \in W$

$$\dot{w} = f(q_0, v)$$

$$v(0) = v_0$$

and remain constant the discrete state $q$

$$q(t) = q_0.$$

If the actual state $v$ remains in $Dom(q_0)$ then the continuous elaboration is active and executed. A discrete transition to $q_1$ is determined for the current value of $v$, when is in the guard $G(q_0, q_1) \subseteq (M_1^*)^{\mathsf{R}_+} \times \ldots \times (M_l^*)^{\mathsf{R}_+} \times (M_{l+1})^{\mathsf{R}_+} \times \ldots \times (M_n)^{\mathsf{R}_+}$ of some edge $(q_0, q_1) \in E$. After a discrete transition the reset function sets some values in $R((q_0, q_1), v) \subseteq (M_1^*)^{\mathsf{R}_+} \times \ldots \times (M_k^*)^{\mathsf{R}_+} \times (M_{k+1})^{\mathsf{R}_+} \times \ldots \times (M_m)^{\mathsf{R}_+}$ for the variables in $W \cup O_d$ with $m \le n$, this way the continuous evolution can be reset.

## III. Static Sampling Architecture

In communication/signal theory the term *sampling* indicates the operation to make an analogue signal to be a discrete signal. The sampling may be *periodic* if the period is constant, denoted as $\Delta$, and *variable* if the period is not constant, denoted $\delta(n)$ with $n \in \mathsf{N}_+$. We build an approach based on the work in [4], where the authors apply a sampling approach to hybrid automata with a constant period, but we introduce a variable period. We define now the architecture of our sampling approach, defining the hybrid component and the controller. We start defining sequences of sampled values from a continuous stream, which represent the input or output data produced in the discretized architecture.

**Definition 3 (Sampled Stream):** Considering a finite set $M$ where $\perp \notin M$ and a finite or infinite timed sequence of elements of $M$ associated to a continuous stream:

$$s = (m_1, t_1)(m_2, t_2)\ldots(m_k, t_k)\ldots$$

where $0 \le t_1 < t_2 < \ldots < t_k < \ldots$, $m_{i+1} \in M$ ( $m_{i+1} \in M^*$ if associated to a discrete stream), $t_{i+1} \in \mathsf{R}_+$ for $i \in \mathsf{N}$, $i < |s|$. The sequence $s$ may be infinite or finite and we associate a continuous stream $x$ as specified in Definition 1 to a sampled sequence:

$$\alpha : t \in \mathsf{R}_+ \mapsto \begin{cases} m_{i+1} \in M & \text{if } t = t_{i+1} \text{ for some } i \in \mathsf{N} \\ (m_{i+1} \in M^* & \text{if discrete stream}) \\ \perp & \text{otherwise} \end{cases} \quad (1)$$

This time-event function $\alpha$ returns at each time instance values of the message set $M$ and $\perp$ in case of absence of messages. We denote with $\mathsf{S}_M$ the set of all such sampled streams. We define now the architecture of our sampling approach, defining the hybrid component and the controller.

**Definition 4 (Sampled Hybrid Component):** A hybrid sampled i/o hybrid component corresponding to the i/o hybrid component $H$ is the following map $\nu_H : \mathsf{S}_{N_1} \times \ldots \times \mathsf{S}_{N_n} \to \mathsf{S}_{M_1} \times \ldots \times \mathsf{S}_{M_m}$

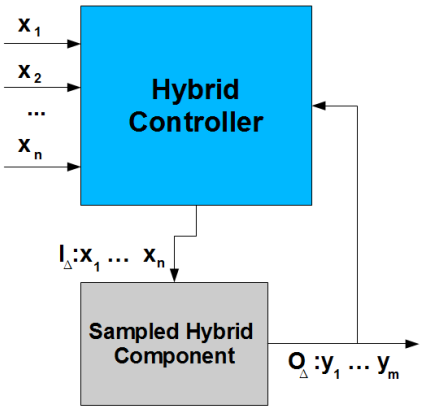


Fig. 2. Sampling architecture for hybrid components.

**Definition 5 (Hybrid Controller):** A hybrid controller is a map

$$\mathsf{C} : \mathsf{S}_{M_1} \times \ldots \times \mathsf{S}_{M_m} \to \mathsf{S}_{N_1} \times \ldots \times \mathsf{S}_{N_n}$$

where the $N_i$ and $M_i$ are respectively the input and output message sets for i/o channels of $H$.

The set of outputs that arrive to the controller are received at sampling intervals, so the controller is activated at a sampling rate $\Delta > 0$ and provides the corresponding input for the next sampled period. We have a representation of the i/o sampled component and its controller in Fig. 2, where the symbols $I_\Delta$ and $O_\Delta$ indicates the sampled i/o streams deduced from the streams specified in Definition 3.

## IV. Dynamic Sampling Algorithms

We introduce in this section our approach based on a variable period. The basic idea is to adapt the length of the period, in the sampled architecture, to the necessary precision for the elaboration of the values produced from the differential equations and therefore for the continuous output of the component. At the beginning the period is set to the maximum value $d_0$.

The first approach is based on the calculation of the slope for the functions defined by differential equations and a correspondent acceptance value for it. The sampling is considered stable if the slope is smaller or equal than acceptance value. As shown in Fig. 3, when the slope between the value at time $t_2$ and at the actual time $t_3$ is not in the acceptance value, the period is halved. At time $t_9$ since the slope was in the acceptance value for a period of time greater to the stabilization time the period is again doubled, until eventually the maximum value. In this architecture we associate a hybrid component $H$ to the dynamic sampling controller $\mathsf{C}_H$.

Definition 6 (Period variation based on the slope): Given a hybrid component $H$, a set of acceptance values $L_i$ and a set of stabilization times $S_i$, with $0 \le i < |W|$ we define the dynamic variation of the period as follows: we calculate the slope from the second step at time $t = 2d_0$ and consequently for the following steps. The evolution of the system is guided from the differential equation over the variable in $w \in W$, that is $\dot{w} = f(q,v)$, where $(q,v)$ is the actual state and $\delta(n-1)$ is the actual period of the component $H$ for the $n$-step of elaboration. At the elaboration step $n$ for each variable $w \in W$ we use the precedent value of $w$ at the computation step $n-1$ to compute the slope:

$$slope(w,n) = \frac{w(n) - w(n-1)}{\delta(n-1)}$$

We update the period for the next computation step only if the slope is greater than the acceptance value $L_w$ or was in such interval for the stabilization time $S_w$ and the slope is not the maximum value $d_0$. This control is done for each variables $w \in W$ and at the end is chosen for the component the shortest value for the period. The map $\delta$ is so inductively defined, for an initial state $(q_0, v_0)$ the map is set for the first step of elaboration to the maximum value $d_0$:

$$\delta(0) = d_0$$

$$\mathrm{w}_{\delta_{i+1}} = \begin{cases} \delta(\mathrm{i})/2 & \text{if } |\mathrm{slope(w,i)}| > \mathrm{L_w} \\ \delta(\mathrm{i}) \times 2 & \text{if } (|\mathrm{slope(w,i)}| \le \mathrm{L_w} \wedge |\mathrm{slope(w,i-1)}| \\ & \le \mathrm{L_w} \wedge \ldots \wedge |\mathrm{slope(w,i-k)}| \le \mathrm{L_w}) \\ & \wedge(\delta(\mathrm{i}) + \delta(\mathrm{i-1}) + \ldots + \delta(\mathrm{i-k}) \ge \mathrm{S_w}) \\ & \wedge \delta(\mathrm{i-1}) < \mathrm{d_0} \quad \text{with } \mathrm{k} \le \mathrm{i} \\ \delta(\mathrm{i}) & \text{otherwise} \end{cases} \quad (2)$$

$$\delta(i+1) = min\{w_{\delta_{i+1}} \mid w \in W\}$$

Acceptance values and stabilization periods for the continuous variables has to be based on the range of values that the variables reach in the elaboration. The acceptance value of the slope for a variable is in fact related to the values assumed from the variable. If a variable has a relative small interval of values, the slope should also be not too big, in order to follow the fluctuations with a finer production of values.

We consider the variables independently from the actual differential equation associated to the actual state. In this way when a transition is taken and the variable can obtain a

different differential equation, the change of values can also determine a high slope with the precedent value. Therefore, when the state is changed probably the slope determines a change in a smaller period, and then there is a higher production of values. A change in the state is normally a change in the modus of the system, so a better precision can guarantee a reactive simulation.

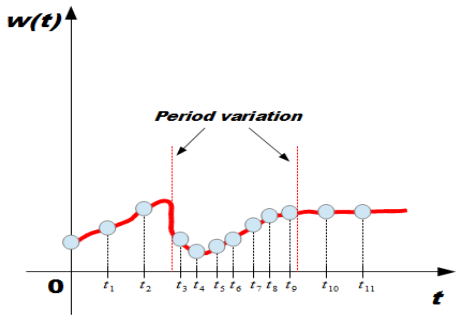


Fig. 3. Period variation based on the slope.

The second idea for the period variation is conceived around predefined value intervals, said critical. Basically during the simulation of the hybrid component, the period varies according with the current values of the variables in $W$, respect to predefined critical intervals, as depicted in Fig. 4.

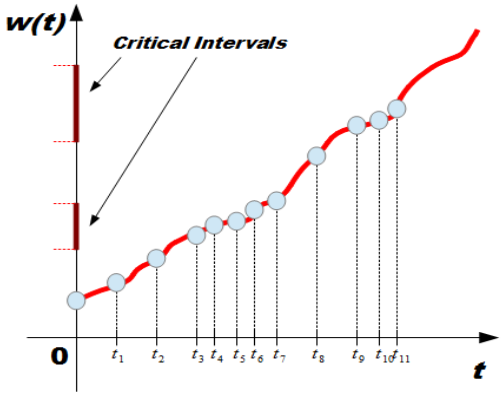


Fig. 4. Period variation based on the critical intervals.

If the value of a variable is in a critical interval and the actual period is greater than the acceptance period associated to the interval, then the period is set to this value, as for instance from the step at time $t_4$ and $t_{10}$. On the other hand if the value of a variable is no more in a critical interval, then the period is set again to the maximum value $d_0$, as at time $t_7$. The period value for the next computation step is calculated as said for each output or internal variable, and finally will be chosen the minimum period between the values elaborated for each variable.

Definition 7 (Period variation based on intervals): Given a hybrid component $H$, a set of critical intervals $K_i$ and a set of period times $P_i \in \mathbb{R}_+$, with $0 \le i < |W|$, we build the dynamic variation of the period as follow. For the initial states as for instance $(q_0, v_0)$ the period is set to the maximum value $d_0$, from the second step of elaboration (at time $t = 2d_0$) starts the dynamic choice of the period. The evolution of the system is guided from the differential equation over the variable in $w \in W$, that is $\dot{w} = f(q,v)$, where $(q,v)$ is the actual state.

The modification of the actual period is made analysing the actual value of the variables in $W$, that is if a variable $w$ is in a critical interval $K_w$, then the corresponding $P_w$ acceptance period is the necessary value of $w$ for the period. It is finally chosen the minimum acceptance period between the variables that are in a critical interval. The period map $\delta$ is inductively over the elaboration steps as follow determined:

$$\delta(0) = d_0$$

$$w_{\delta_{i+1}} = \begin{cases} P_w & if\ w(i) \in K_w \\ \delta(i) & otherwise \end{cases}$$

$$\delta(i+1) = min\{w_{\delta_{i+1}} \mid w \in W\} \tag{3}$$

Critical intervals have to be established before the execution of the sampled hybrid components and they should have appropriate sampling period, said acceptance periods. These information is strictly bound to the needs and real-time restrictions of the variables, therefore they can be delineated from the formal requirements of each variable of the systems or from the modelled hybrid component considering the implementation behaviours.

## V. Conclusion and Future Work

We presented an approach that comprises modelling and simulation of hybrid systems. The presentation of our results starts with Focus theory fundamentals, based on these modelling structure we developed our notion of hybrid component and then we propose two dynamic sampling solutions. These solutions consider the values reached from the variables and adapt the period of the sampling, in order to follow sudden variations or critical intervals reached from the variables. Focus theory is supported and implemented in the modelling tool Auto Focus 3[5], developed at our research group and specialized for reactive and discrete embedded systems, it integrates specification and verification tightly into the model-based development process, based on the model checker NuSMV [6].

We aim to implement hybrid components, the algorithms for their sampling and their formal verification in Auto Focus 3. Through these verification techniques may be also useful for the sampling algorithms, checking value invariants of the possible value ranges reached from the variables and so deriving congruent slope acceptance values. Finally, we plan to conduct more case studies with industrial partners involved in the automotive and embedded system domain, in common research projects.

## Acknowledgment

I would like to thank Mario Gleirscher and Maximilian Junker, who provided valuable comments during the elaboration of the research presented in this work.

## REFERENCES


[1] T. A. Henzinger, "The theory of hybrid automata," in *Proc. The 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, IEEE Computer Society, Washington, DC, USA, 1996, pages 278-.

[2] M. Broy, F. Dederich, C. Dendorfer, M. Fuchs, T. Gritzner, and R. Weber, "The design of distributed systems - An introduction to FOCUS," Technical Report TUM-I9202, Technische Universität München, 1992.

[3] M. Petreczky, D. A. van Beek, J. E. Rooda, P. J. Collins, and J. H. van Schuppen, "Sampled-Data control of hybrid systems with discrete inputs and outputs," in *Proc. the 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, A. Giua, M. Silva, and J. Zaytoon, Eds, International Federation of Automatic Control, 2009

[4] O. Müller and P. Scholz, "Functional Specification of Real-Time and Hybrid Systems," in *Proc. Hybrid and Real-Time Systems*, pp. 26-28, Springer-Verlag, 1997.

[5] AF3Wiki. [Online]. Available: http://autofocus.in.tum.de

[6] NuSMV: a new symbolic model checker. [Online]. Available: http://nusmv.fbk.eu



**Alarico Campetelli** received the MSc and BSc (Hons) degrees in Computer Science, with specialization in formal languages, from University "La Sapienza" in Rome, respectively in 2007 and 2004. I had worked from 2007 to 2008 at Gruppo Repubblica in Rome as software developer, and from 2008 to 2009 at Artificial Technology in Munich as researcher and software developer. In 2009 I joined at the Technische Universität Munich, where I am currently a PhD student at the chair of Software & Systems Engineering. I had worked in research projects with industrial partners and I had also lectured. My current research interests are in formal methods, formal verification, hybrid systems and model-based development.