Discrete-Time Refinement of Hybrid Automata*

Thomas Stauner

BMW Car IT, Petuelring 116, D-80809 München, Germany
stauner@in.tum.de, Phone: +49 89 3101987

Abstract. Notations like hybrid automata are highly useful in the development process of hybrid systems to document requirements in early design steps. When it comes to implementation a part of the requirements will be realized in software in a discrete-time manner. We therefore study sufficient conditions which ensure that an automaton operating in discrete-time *refines* a hybrid automaton with its underlying continuous time model. Our notion of refinement provides that vital properties which have been established for the hybrid automaton also hold for its refinement. Furthermore, we outline a method how to derive a discrete-time refinement from a hybrid automaton.

1 Introduction

In our view notations which allow a joint specification of discrete and continuous behavior, such as hybrid automata [1], are highly useful for elicitating, elaborating, discussing and documenting requirements of hybrid systems. Furthermore, such formal notations are the basis for computer-aided validation methods like simulation and model checking. In later design steps the elaborated abstract, mixed discrete and continuous model can then be refined towards implementation. [11] and [12] propose such processes.

Moving from an abstract model based on a continuous time scale to implementation amounts to changing to a discrete-time execution scheme for those components in the model which are implemented on (or in) digital hardware. Due to the cost structure and flexibility of software-based solutions, such discrete-time execution usually is desired for large parts of a hybrid system. In order to ensure that vital properties of the abstract model are satisfied in the implementation oriented discrete-time model, the change of the execution scheme must be performed in a controlled way. This is an essential prerequisite for the utility of development processes of hybrid systems which build upon hybrid notations and an early validation of the obtained hybrid models, before implementation related decisions are made. Such decisions e.g. include the partitioning of a model into submodels with different time scales.

Therefore, this paper identifies conditions under which the transition from continuous-time to discrete-time is a refinement step which preserves vital system

* This work originated at the Institut für Informatik, TU München, and was supported with funds of the DFG under reference number Br 887/9 within the priority program *Design and design methodology of embedded systems.*

C.J. Tomlin and M.R. Greenstreet (Eds.): HSCC 2002, LNCS 2289, pp. 407–420, 2002.

properties. The paper considers discrete-time refinement of hybrid automata. It is derived from [12] where a similar, more extensive theory is developed for the HyChart notation [4]. The work presented in the paper only regards isolated automata without event based communication. Automata with input and events are considered in [12].

Related Work. A refinement notion similar to ours is also used in [2]. In [7] the authors give proof rules which allow them to deduce the validity of duration calculus formulas in discrete-time from the validity of the formulas in continuous-time. In our view this can be regarded as refining the contiuous time formulas. The refinement method we develop is based on a hybrid automata dialect which allows some imprecision in the taking of transitions and the values of variables. Other variants of hybrid automata with uncertainties are introduced in [5, 3]. In particular, [3] is closed to our approach.

Overview. This paper is organized as follows. First, we explain effects of a discrete-time execution of components of a hybrid system and their implications on hybrid automata in Section 2. Section 3 defines hybrid automata, refinement and so-called discrete-time hybrid automata, which satisfy some restrictions that allow their direct implementation in discrete-time. With this basis, Section 4 presents a refinement technique for the transition from (general) hybrid automata to discrete-time hybrid automata. Finally, we draw some conclusions in Section 5. The online version of the paper, available under http://www4.in.tum.de/~stauner/papers/, additionaly contains an appendix with an example. The paper assumes some familiarity with hybrid automata [1].

2 Implementation Effects

In our view hybrid automata-like notations can beneficially be used for requirements capture and the early design steps of hybrid embedded systems. In these phases designers usually want to express that some discrete actions (or *transitions*) are taken when certain conditions are satisfied. They are not so much interested in the detailed timing, i.e. in possible small delays between the (first) satisfaction of conditions and the execution of the corresponding actions. They are, however, aware that such delays exists when the model is implemented digitally. One primary reason for such delays is that the digital components can only sense the status of their environment within the discrete time grid given by their clock. In our view sampling rates and timing uncertainties are implementation related effects. Therefore we think it is not adequate to already consider them in detail in early design phases. Nevertheless abstract models occurring in early design phases must be designed in a way that tolerates small delays without violating vital system properties. For instance, in control theory it is known that delays, such as those caused by sampling, can lead to instability. Otherwise, if the model's correctness relies on the absence of any delays, it cannot be implemented later on.

Besides timing uncertainties, there is a further kind of deviations from the ideal model. When an action which is triggered by the boundary crossing of

a continuosly evolving quantity x, i.e. triggerd by a condition like $x \geq c$, is taken by a digital component, the boundary c will usually already be exceeded. In other words, in case of actions which depend on the value of variables that evolve continuously, the timing uncertainty corresponds to an uncertainty in the actual value of the variable for which the action is executed. For given analog dynamics, one kind of uncertainty can be derived from the other.

Fig. 1 visualizes the effect of sampling on the detection of a boundary crossing for boundary value c. For trajectory x and sampling period T, the boundary crossing of x is detected with delay. The crossing of trajectory y is not detected with sampling period T, because it happens between two consecutive sampling instants. Trajectory y thus indicates that the boundary crossing cannot be detected with arbitrary precision with sampling. For hybrid automata employing such boundary crossing events as guards of transitions, this means that transitions can usually not be taken immediately when enabled, but with some delay or, under awkward circumstances, not at all. Considering the value of the trajectory when such a transition is taken, we see that in the depicted example x is already greater than c at $3T$. Obviously, the delayed execution of transitions may violate invariants which are required to hold for the system or may lead to instability, because of too late reaction.

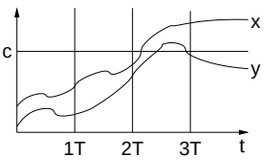

Fig. 1: Delayed or failing detection of boundary crossing events caused by sampling.

Implementation effects on analog dynamics. Let us now regard the analog dynamics of a hybrid component. The analog dynamics on the one hand affects the discrete dynamics (the state transitions) and on the other hand it can affect external components. We assume the dynamics are given by some differential equation $\dot{x} = f(x)$. In this case we distinguish two kinds of effects resulting from an implementation of the analog dynamics in discrete time. At sampling instants the discretized analog dynamics in general only yields an approximation of the exact dynamics. As an analogy, think of a numerical algorithm (the discretized analog dynamics) that computes the solution to $\dot{x} = f(x)$ for given initial value x_0. At each interpolation point kT of the algorithm, it will provide an approximation of the exact value of the solution $x(kT)$. Like in numerics, we call the deviation between the exact value and the approximating value at sampling instants the *discretization error*. It affects the discrete part as well as external components that depend on x. A further error results between sampling instants, where the discretized analog dynamics does not produce new output. In this paper, we choose to extend the (discretized) analog dynamics' output at a sampling point over the interval until the next sampling instant by holding it constant. In control systems terminology, this correponds to a so-called *zero-order hold*.[1] The effect of this strategy is that there is a further deviation of the output w.r.t. the exact dynamics between sampling instants. At the sampling

[1] Other extension strategies besides the zero-order hold would be feasible as well, but, as Ogata [10] mentions, more complicated construction schemes for continuous-time signals from discrete-time signals are usually not used in control applications.

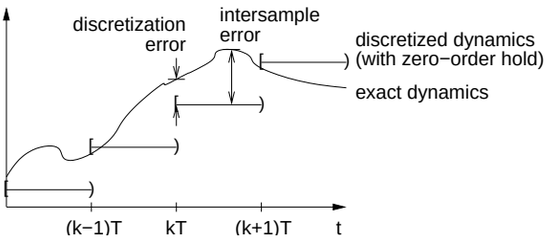

Fig. 2. Discretization error and intersample error.

points this deviation however vanishes. We therefore call it the *intersample error*. Both kinds or errors are visualized in Figure 2. We define the intersample error as the maximum deviation between the exact value and the time-extended output of the discetized analog dynamics during a sampling period. In hybrid automata, this error affects the discrete dynamics, because between sampling points the exact dynamics could enforce a transition, while no transition may be necessary for the discretized dynamics which simply remains constant during a sampling interval. Furthermore, the environment of a component is affected by this error.

Impact on verification. The effect of discrete time implementation also has to be considered when planning verification steps for a system. Namely, there is not much sense in formal, exhaustive verification of abstract models with tools like model checkers, if the properties established for the abstract model can not be transfered to the implementation, because the abstract model is unrealistically precise. In contrast, verification is helpful if the abstract model is liberal enough to comprise effects that result from its implementation, like time delays and deviations from the exact continuous dynamics. Model checking techniques which weaken the exactness with which analysis of hybrid models is performed have been proposed in [5, 6] and [3].

Methodology. As a result, the models of a system component specified in the requirements capture phase and the early design phase should allow for timing uncertainties and uncertainties in the analog dynamics. In hybrid automata, invariants which overlap with transition guards can be used to express uncertainty w.r.t. when exactly a transition is taken. In order to specify the uncertainty associated with the analog dynamics we introduce a relaxation constant into our dialect of hybrid automata (see below). To deal with the intersample error we introduce a further relaxation when we consider the traces produced by a hybrid automaton.

From an external point of view which only observes the generated traces, the uncertainties in the model leads to sets of traces. In further design steps the timing uncertainty and the permitted deviations from the analog dynamics can be used by the designers to select a sampling rate for the embedded hard- and software that guarantees (1) that transitions are taken within the delay permitted by the relaxed model and (2) that the discretized analog dynamics is

sufficiently precise. The selection process can be guided by the modeling tool and may involve standard techniques from control theory, like Shannon's sampling theorem. This way a new model which only generates a subset of the traces of the original model is obtained. The new model is therefore called a *refinement* of the original one. Properties which were true for the original model can then be transferred to the new model: If every trace of the original model is guaranteed to satisfy a given property, then all traces of the new model will also satisfy the property, because every new trace also is in the trace set of the original model. A mode detailed study of properties is given in [12].

3 Preliminaries Concerning Hybrid Automata

Hybrid automata. For our purpose, we modify the standard definition of hybrid automata [1] in some aspects. First, we omit synchronization labels. In connection with external components, synchonization labels may enforce the immediate taking of a transition in a hybrid automaton. This is too precise to allow flexible refinement of the automaton. Second, we introduce a relaxation of the continuous activities in the automaton. This means that variables need not evolve exactly as specified by the activity, but some deviation from the specified exact evolution is permitted. As we will see below, this relaxation acts as an upper bound on the permitted discretization error in a discrete-time implementation. Third, the semantics of invariants is slightly different in our model. We only require that the invariant of a location holds, if time passes in that location. When a location is left or entered via a transition the invariant need not hold. This way transient locations, i.e. locations in which no time passes, can be specified. We use this in our discrete-time dialect of hybrid automata introduced below. Finally, we do not consider any input here in order to simplify the presentation. Discrete-time refinement which also regards external input is studied in [12].

In this paper a *hybrid automaton* A is a 7-tuple $(X, V, inv, init, Act, \varepsilon_{dis}, E)$ with the following meaning:

- X is a finite ordered set $\{x_1, ..., x_n\}$ of variables over $\mathbb{R}$. A valuation s is a point in $\mathbb{R}^n$, the value of variable x_i in valuation s is the i-th component of s, s_i. Often we also use notation $s.x_i$ to denote the value of variable x_i, instead of writing s_i. $\mathbb{R}^n$ is also called the *data-state space* of the automaton. When Φ is a predicate over the variables in X, we write $[\![\Phi]\!]$ to denote the set of valuations s for which $\Phi[X := s]$ is true.
- V is a finite set of locations. A state of automaton A is a tuple (v, s) consisting of a location $v \in V$ and a valuation s.
- inv is a mapping from the set of locations V to predicates over the variables in X. $inv(v)$ is the *invariant* of location v. When time passes in v, only valuations $s \in [\![inv(v)]\!]$ are allowed.
- $init$ is a mapping from the locations in V to predicates over X. $init(v)$ is called the initial condition of v. (v, s) is an initial state if $init(v)$ is true for s.

Act is a mapping from the locations in V to convex predicates over $X \cup \dot{X}$, where $\dot{X} = \{\dot{x}_1, ..., \dot{x}_n\}$ and $\dot{x}_i$ denotes the first time derivative of x_i, dx_i/dt. A predicate over variables in a set Y is called *convex* if it is a finite conjunction of inequalities $<, \leq, >, \geq$ between arithmetic terms over Y.[2] When control is in location v the variables evolve according to differentiable functions which approximatly satisfy the activity $Act(v)$.

ε_{dis} is a mapping from the locations in V to the nonnegative real numbers $\mathbb{R}_+$. For each location $\varepsilon_{dis}(v)$ specifies a permitted error by which the continuous evolution (according to *Act*) may be affected.

A finite set E of edges called transitions. A transition $(v, jp, v') \in E$ has source location v and target location v'. jp is called the transition's jump condition. It is a convex predicate over variables $X \cup X'$, where $X' = \{x_1', ..., x_n'\}$. x_i' stands for the value of variable x_i after the transition is taken, while x_i refers to the value before the transition. Jump conditions are structured into a guard and a body. The guard is the precondition for the transition and if the transition is taken, the body determines the next value of the variables. The guard is a convex predicate over variables X and the body is a convex predicate over variables $X \cup X'$. If the guard is true the body is satisfiable.

All of the above mappings are total.

Semantics. The *transition-step relation* $\multimap$ is defined as $(v, s) \multimap (v', s')$ iff there is a $(v, jp, v') \in E$ such that the variables change according to the jump condition, i.e. $jp[X, X' = s, s']$ is true. Transition steps are discrete, no time passes while they are taken. Unlike standard hybrid automata we do not require that the location invariants hold in the source or destination location. For our purpose, this is more convenient for enforcing transitions.

The δ *time step relation* $\rightsquigarrow_\tau^\delta$ is defined as $(v, s) \rightsquigarrow_\tau^\delta (v, s')$ iff there are two differentiable functions $\rho, \tau \in [0, \delta] \to \mathbb{R}^n$, $\delta \geq 0$, with

- $\tau(0) = s$ and $\tau(\delta) = s'$
- the invariant of v is satisfied during $[0, \delta)$: $\tau(t) \in [\![inv(v)]\!]$ for $0 \leq t < \delta$
- τ is within $\varepsilon_{dis}(v)$ distance of ρ: $d(\tau(t), \rho(t)) \leq \varepsilon_{dis}(v)$ for $t \in [0, \delta]$, where $d(.,.)$ is the Euclidean distance on $\mathbb{R}^n$.
- the activity $Act(v)$ is satisfied for ρ: $Act(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)]$ is true for $t \in [0, \delta]$, where $\dot{\rho}(t) = (\dot{\rho}_1(t), ..., \dot{\rho}_n(t))$.

Note that the regarded location's invariant need not hold at the end point δ. A function τ for which there is a function ρ such that the last two items from above hold, is said to *satisfy* $Act(v)$ *up to* ε_{dis}.

The *hybrid step relation* $\blacktriangleright$ is defined as $(v, s) \blacktriangleright_\tau^\delta (v', s')$ iff there is a $s'' \in \mathbb{R}^n$ such that $(v, s)(\multimap)^*(v', s'')\rightsquigarrow_\tau^\delta (v', s')$, where R^* is the arbitrary but finite iteration of relation R (including the 0-fold iteration, i.e. the identity relation).

A *(continuous-time) execution* of a hybrid automaton is a finite sequence of hybrid steps $(v_0, s_0) \blacktriangleright_{\tau_0}^{\delta_0} (v_1, s_1) \blacktriangleright_{\tau_1}^{\delta_1} \ldots \blacktriangleright_{\tau_\ell}^{\delta_\ell} (v_{\ell+1}, s_{\ell+1})$, where $s_0 \in [\![init(v_0)]\!]$ and $\ell \in \mathbb{N}$. A *trace* is obtained from an execution by mapping the execution to a function over (a subset of) the time axis $\mathbb{R}_+$. For an execution $(v_0, s_0) \blacktriangleright_{\tau_0}^{\delta_0} (v_1, s_1) \blacktriangleright_{\tau_1}^{\delta_1} \ldots \blacktriangleright_{\tau_\ell}^{\delta_\ell} (v_{\ell+1}, s_{\ell+1})$, the corresponding trace τ is defined by: $\tau(t) = \tau_i(t -$

[2] Equality can be defined as a macro.

$\Sigma_{k=0}^{i-1}\delta_k)$ for $t \in [\Sigma_{k=0}^{i-1}\delta_k, \Sigma_{k=0}^{i}\delta_k)$, where $\Sigma_{k=0}^{-1}x$ is defined to be 0. τ is a piecewise differentiable function[3] if the execution is non-zeno, i.e. if it does not contain infinitely many transition steps within a finite time interval. Its domain is $[0, \Sigma_{k=0}^{\ell}\delta_i)$. We write $Tr(A)$ to denote the set of all traces of a hybrid automaton A.

The relaxation R_ε of a set of traces T is defined by $R_\varepsilon(T) = \{\sigma \in PD \mid \exists \tau \in T.\, dom(\tau) = dom(\sigma) \wedge \forall t \in dom(\tau).\, d(\tau(t), \sigma(t)) \leq \varepsilon\}$, where $dom(\tau)$ is the domain of function τ and PD is the set of piecewise differentiable functions from (subsets of) $\mathbb{R}_+$ to $\mathbb{R}^n$.

Refinement. We say that a hybrid automaton B is a *refinement* of hybrid automaton A iff the set of traces of B is contained in that of A, i.e. $Tr(B) \subseteq Tr(A)$. In order to consider a discrete-time execution scheme as a refinement of a hybrid automaton, some more freedom is necessary due to the intersample error. We therefore use relaxation R_ε: A hybrid automaton B is an ε *refinement* of a hybrid automaton A iff $Tr(B) \subseteq R_\varepsilon(Tr(A))$.

On an abstract level, we can regard universal properties, i.e. properties which have to be satisfied for all executions of a system, as sets of traces (cf. [9]).In this point of view a universal property P is satisfied by a hybrid automaton A iff $Tr(A) \subseteq P$. It is satisfied by the ε relaxation of A, respectively, iff $R_\varepsilon(Tr(A)) \subseteq P$. By the definition of refinement it is obvious that a refinement (ε refinement) of an automaton A is guaranteed to satisfy a universal property P if A (the ε relaxation of A) satisfies P. Note that a more comprehensive study of properties of hybrid systems is given in [12].

Discrete-time hybrid automata. In order to talk about discrete time systems we need a way to specify such systems first. As we want as much similarity to hybrid automata as possible, we do not introduce a new automata model but introduce syntactic restrictions on hybrid automata which ensure that an automaton satisfying these restrictions can in essence only produce traces which are step functions over an equidistant discrete time grid.

A hybrid automaton $A = (X, V, inv, init, Act, \varepsilon_{dis}, E)$ is called a *discrete-time hybrid automaton (DHA)* of time granularity $T > 0$ iff:

- $X = \{x_1, ..., x_n, c\}$ for $n \in \mathbb{N}$. X contains the special variable c called the *clock* of A.
- The invariant of every location is of the form $(c = 0 \wedge p) \vee c \neq T$, where p is a predicate over $X \setminus \{c\}$.
- The initial condition of every location is of the form $p \ \wedge \ c = 0$, where p is a predicate over $X \setminus \{c\}$.
- The activity of every location is $\dot{c} = 1 \ \wedge \ \bigwedge_{x \in X \setminus \{c\}} \dot{x} = 0$. Hence, only the clock changes in time steps.
- $\varepsilon_{dis} = 0$
- The set of transition E can be split into two disjoint sets E_l and E_u such that the following holds. The guard of every transition in E_l is a predicate of the form $c = 0 \ \wedge \ p$ where p is a convex predicate over $X \setminus \{c\}$. The body

[3] A function is piecewise differentiable iff every finite interval can be partitioned in finitely many subintervals such that the function is differentiable on each subinterval.

of every transition in E_l does not modify c. The guard of every transition in E_u is of the form $c = T \ \wedge \ p$ where p is a convex predicate over $X \setminus \{c\}$ and the body of every transition in E_u is of the form $q \ \wedge \ c' = 0$ where q is a convex predicate over $X' \setminus \{c'\}$. Furthermore, for transitions in E_u source location and destination location coincide.

Transitions in E_u reset the clock c. The rest q of their body is intended to perform some numerical computation and determine the value of the variables at the next discrete time instant $k \cdot T$, $k \in \mathbb{N}$. Thus q is intended to replace the continuous update of the variables which is performed by the activities in a hybrid automaton by a discrete update. Therefore, transitions in E_u are also called update transitions. While the activities are differential equations, q will usually result from a difference equation or a more complex numerical computation.

The guards of the transitions and the invariants are chosen in a way which ensures that time T after the last transition was taken, a transition in E_u must be taken (in order to update the variables' values according to an approximation of a differential equation) and then transitions in E_l can be taken or time can progress. Transitions in E_l can only be taken after a transition in E_u, because their guard requires that c is 0. Transitions in E_l are called logic transitions, since they are intended to model the logic-based switching between control modes. Furthermore, we point out that due to the form of the invariants and since activities only modify c, the following holds: If an invariant is true for a valuation s with $s.c = 0$ then a time step of duration T is possible.

Executions and traces of DHA. The executions of a DHA A result from the hybrid step relation as above. It is easy to see that the traces generated by a DHA A and projected on $X \setminus \{c\}$ are piecewise constant functions in which points of discontinuity can only occur at time instants $k \cdot T$, $k \in \mathbb{N}$. In order to simplify proofs in the following we now introduce further step relations which more directly correspond to the operational behavior of DHA.

The *logic-step relation* $\dashrightarrow_l$ is defined as $(v,s) \dashrightarrow_l (v',s')$ iff $(v,s) \bullet\!\!\rightarrow (v',s')$ by a transition in E_l. We define the update relation $\dashrightarrow_u$ as $(v,s) \dashrightarrow_u (v',s')$ iff $(v,s) \bullet\!\!\rightarrow (v',s')$ by a transition in E_u. Note that $v' = v$ holds due to the form of the transitions in E_u. The *control-step relation* $\rightsquigarrow$ is defined as $(v,s) \rightsquigarrow^{\ell}_{(\mu_i)} (v,s')$ iff (μ_i) is a sequence of functions and there are s_i for $i \in \{0,\ldots,\ell-1\}$, $\ell \geq 1$, and s_ℓ such that $(v,s_i)(\bullet\!\!\rightarrow^{T}_{\mu_i}; \dashrightarrow_u)(v,s_{i+1})$, where ; denotes the sequential composition of relations, $s_0 = s$, $s_\ell = s'$ and every function $\mu_i \in [0,T] \to \mathbb{R}^{n+1}$, $i \in \{0,\ldots,\ell-1\}$, in the sequence (μ_i) is defined by $\mu.c(t) = t$ and $\mu.x(t) = s''_x$ for $x \in X \setminus \{c\}$. Informally, a control step $\rightsquigarrow^{\ell}_{(\mu_i)}$ spans ℓ sampling intervals without being interrupted by logic steps. It may be regarded as the execution of a discrete-time control law. The *discrete-time hybrid step relation* $\triangleright^{\ell}_{(\mu_i)}$ is defined as $(v,s) \triangleright^{\ell}_{(\mu_i)} (v',s')$ iff there is a $s'' \in \mathbb{R}^{n+1}$ such that $(v,s)(\bullet\!\!\rightarrow)^*(v',s'') \rightsquigarrow^{\ell}_{(\mu_i)} (v',s')$. We remark that the invariant of v' must be true for s'', because no control step is possible otherwise.

Every finite trace τ of a DHA A can be extended to a trace τ' which is defined on $[0, k \cdot T)$ where k is the smallest natural number such that $dom(\tau) \subseteq dom(\tau')$. In other words, τ can be extended to a trace which ends at a sampling step. The

reason is that invariants do not become false for values of c in $(0, T)$. Therefore, we now restrict our attention to traces which end at sampling steps.

A *discrete-time execution* of a DHA is a finite sequence of discrete-time hybrid steps $(v_0, s_0) \overset{\ell_0}{\underset{(\mu_{i,0})}{\triangleright}} (v_1, s_1) \overset{\ell_1}{\underset{(\mu_{i,1})}{\triangleright}} \dots \overset{\ell_m}{\underset{(\mu_{i,m})}{\triangleright}} (v_{m+1}, s_{m+1})$, where $s_0 \in [\![init(v_0)]\!]$. Corresponding to the traces of continuous-time executions, the trace τ of such a discrete-time execution is defined by: $\tau(t) = \mu_{i,j}(t - t')$ for $t \in [t', t' + T)$ and $t' = (\Sigma_{k=0}^{j-1} \ell_k + i) \cdot T$. If we restrict our attention to traces which end at sampling instants, the set of traces generated for a DHA by discrete-time hybrid steps and by continuous-time hybrid steps coincide. The proof is simple and omitted here. The similar property does not hold on the level of executions, since the hybrid step relation includes idle steps and time steps of duration less than T while a single discrete-time hybrid step lasts at least time T.

4 Refinement to Discrete Time

We start with a theorem that establishes a sufficient condition for a DHA B to be an ε refinement of a hybrid automaton A. Then, a systematic way to construct such a DHA B from a specific type of hybrid automata is introduced.

4.1 Refinement Theorem

The way we formulate the theorem below is motivated by the desire to be able to consider invariants (and transitions) in separation from activities when refinement is regarded. Therefore, we do not explicitly relate control steps of a DHA to time steps of a hybrid automaton, but consider the satisfaction of invariants (item 3 below) and activities (item 4 below) separately. This requires to abstract the set of functions (or evolutions) permitted by the activities of a hybrid automaton by a set of evolution constraints. Typically, such an abstraction can be the set of all Lipschitz continuous functions with Lipschitz constant less or equal to a given constant.[4] In the following we write S for the set of functions which abstracts the activities and call it the set of evolution constraints. We write S_t for the set of all functions in S whose domain is a superset of $[0, t)$.

Theorem 1. *Let $A = (X, V, inv_A, init_A, Act_A, \varepsilon_{dis}, E_A)$ be a hybrid automaton and let $B = (X \cup \{c\}, V, inv_B, init_B, Act_B, 0, E_B)$ be a DHA of time granularity T. B is an ε refinement of A if there is a set of evolution constraints S for A such that the following holds.*

1. *the initial conditions in B result from those of A by adding conjunct $c = 0$ (clock initialization) to them: $\forall v \in V.\, init_B(v) \equiv init_A(v) \ \wedge \ c = 0$*

[4] A function τ is Lipschitz continuous iff there is a constant $L \geq 0$ such that $\forall x, y \in dom(\tau).\, d'(\tau(x), \tau(y)) \leq L \cdot d(x, y)$ where d and d' are metrics on domain and rage, respectively, of function τ.

2. *if a logic step is possible in B, a similar step is possible in automaton A:* $\forall u, v.\, u \dashrightarrow_l v \Rightarrow \pi_X(u) \bullet\!\!\rightarrow \pi_X(v)$, *where* $\dashrightarrow_l$ *is the logic step relation of B and* $\bullet\!\!\rightarrow$ *is the transition step relation of A. π_X denotes projection of valuations of B to valuations of A, i.e. the projection eliminates the valuation of variable c. Note that c must be* 0 *by definition of logic steps, which means that logic steps can only occur at sampling instants.*
3. *the invariant of a location v of A is guaranteed to remain true during a sampling period, if the invariant of v in B is true at a sampling instant, i.e. for $c = 0$. In other words, for a start state (v, s) no activity of A can violate the invariant of v during time T, if the invariant of v in B for valuation s' with $\pi_X(s') = s$ and $s'.c = 0$ is true:*
$\forall v \in V, s' \in \mathbb{R}^{n+1}.\, s'.c = 0 \ \wedge \ s' \in [\![inv_B(v)]\!] \Rightarrow (\forall \tau \in S_T.\, \tau(0) = \pi_X(s') \Rightarrow \forall t \in [0, T).\, \tau(t) \in [\![inv_A(v)]\!])$
4. *for every control step $(v, s) \triangleright^{\ell}_{(\mu_i)} (v', s')$ of B, (1) there is a function $\tau \in [0, \ell \cdot T] \to \mathbb{R}^n$ which is in the ε_{dis} neighborhood of a function ρ that satisfies $Act_A(v)$ and (2) τ and the respective μ_i agree at sampling instants and (3) between sampling instants they agree up to ε. Formally:*
$(v, s) \triangleright^{\ell}_{(\mu_i)} (v', s') \Rightarrow \exists \tau, \rho \in [0, \ell \cdot T] \to \mathbb{R}^n.$
$(\forall t \in [0, \ell \cdot T].\, Act(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)] \ \wedge \ d(\tau(t), \rho(t)) \leq \varepsilon_{dis}) \ \wedge$
$\forall i \in \{0, \ldots, \ell - 1\}.\, \pi_X(\mu(iT)) = \tau(iT) \ \wedge$ (sampling instants OK)
$\forall t \in [0, \ell \cdot T).\, d(\pi_X(\mu(t)), \tau(t)) \leq \varepsilon \ \wedge$ (intersample response OK)
$\pi_X(s') = \tau(\ell \cdot T)$ (last value OK)
where function $\mu \in [0, \ell \cdot T) \to \mathbb{R}^{n+1}$ is defined by $\mu(t) = \mu_i(t - i \cdot T)$ for $i \in \{0, \ldots, \ell - 1\}$ and $t \in [iT, (i + 1)T)$.
5. *set S of evolution constraints indeed is an abstraction of the trajectories produced by the activities of A. In other words, every function τ which satisfies $Act_A(v)$ up to ε_{dis} for a $v \in V$ is in S. Formally:* $\forall \tau.\, (\exists \rho.\, \forall t \in dom(\tau).\, Act(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)] \ \wedge \ d(\tau(t), \rho(t)) \leq \varepsilon_{dis}) \Rightarrow \tau \in S$

B is also called a discrete-time ε refinement *of A.*

The proof of the theorem proceeds by induction over the discrete-time hybrid step relation. (Remember that the traces of B, which – without loss of generality – end at sampling instants, can be constructed from iterating the discrete-time hybrid step relation starting from a state that satisfies the initial condition of B.) Assumption 1 of the theorem assures that every initial state of B is an initial state of A (disregarding variable c). Assumption 2 assures that for every logic step of B there is a corresponding transition step of A. Assumptions 3 to 5 guarantee that for every control step of B there is a corresponding time step of A which produces a trace that is similar to that of B up to ε. Technically, assumption 4 assures that for a control step of B there is a corresponding function τ satisfying the respective activity of A. Assumption 3 then ensures that the invariant of the location in which the control step takes places is not violated by any activity, i.e. that a time step is indeed possible. In the formal proof, assumption 3 must be applied inductively ℓ times for a control step spanning ℓ sampling intervals. A detailed proof of a similar, more complex theorem in the context of *HyCharts* is given in [12].

4.2 Constructing Discrete-Time Refinements

In the following we introduce a method how to derive a DHA B and a time granularity $T > 0$ from a hybrid automaton A such that B satisfies assumptions 1 to 4 of Theorem 1. The method requires that A has a certain form described below.

First, note that Theorem 1 already determines that B must have the same locations as A, the same variables plus clock c and $\varepsilon_{dis} = 0$ for B. Furthermore, assumption 1 of the theorem derives the initial conditions in B from those in A by adding a clock initialization. The activities of B are fixed by the definition of DHA. Thus, what is left, is to derive the invariants, the sets of transitions E_l and E_u and the time granularity T for DHA B from automaton A.

4.2.1 Invariants and Logic Transitions We start with the invariants and logic transitions E_l and derive constrains on T.

Ideal and relaxed invariants. For our method, we require that the invariants of A are constructed from the transitions by negating the transition guards and introducing a relaxation of these "ideal" invariants. In more detail, we require that the invariant $inv_A(v)$ of a location v in A is constructed in the following way. First, the ideal invariant of v is defined as the conjunction of the negated guards of all transitions with source location v. As all guards are convex predicates over X, the ideal invariant ϕ can schematically be rewritten as $\bigwedge_{i=1}^{g}(\bigvee_{j=1}^{h_i} at_{i,j})$ where each $at_{i,j}$ is a negated comparison of arithmetic expressions over X. The relaxed invariant ψ results from relaxing the negated atoms. In detail, each negated comparison $at_{i,j}$ is rewritten in the form $a_{i,j}\#0$, where $a_{i,j}$ is an arithmetic expression and $\# \in \{<, \leq\}$. Then this predicate is replaced by atom $at'_{i,j} \equiv a\#\varepsilon_{i,j}$, where $\varepsilon_{i,j} > 0$ is a constant. Thus, ψ is given by $\bigwedge_{i=1}^{g}(\bigvee_{j=1}^{h_i} at'_{i,j})$. The constants $\varepsilon_{i,j}$ provide that the relaxed invariant remains true for some trespassing of threshold 0 in the expressions $a_{i,j}\#0$. This means that the relaxed invariant does not become false at once when a transition guard becomes true, but it remains true for some more time, until the threshold is trespassed by amount $\varepsilon_{i,j}$. On essential idea in time-discretization is to choose the time granularity T in a way which ensures that a clock tick lies within every time interval in which the value of such expressions $a_{i,j}$ rises from 0 to $\varepsilon_{i,j}$.

Definition of B's invariants and logic transitions. With these preliminaries we define the invariants and logic transitions of B as follows. For every location v with relaxed invariant ψ in automaton A, we define $inv_B(v) \equiv (c = 0 \wedge \phi) \vee c \neq T$, where ϕ is the ideal invariant corresponding to ψ (resulting from the negation of transition guards in A, as explained above). Furthermore, the set of logic transitions E_l of B is defined by adapting the transitions of A by adding $c = 0$ to all transition guards and adding $c' = c$ to the bodies such that c remains constant: $E_l = \{(v, g \wedge c = 0 \wedge b \wedge c' = c, v') \mid (v, g \wedge b, v') \in E_A\}$. With these guards and invariants, a logic transition in B must be taken, if at a sampling instant, i.e. for $c = 0$, the current location's ideal invariant ϕ is false.

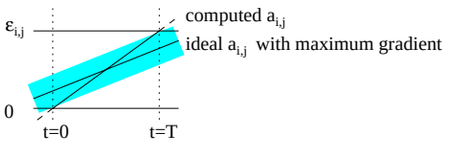

Fig. 3. Worst case in the evolution of $a_{i,j}$: Due to ε_{dis}, the evolution of $a_{i,j}$ is affected by an error $\delta_{i,j}$ (in grey in the figure). The computed value of $a_{i,j}$ therefore crosses the interval $[0, \varepsilon_{i,j}]$ faster than it could in the ideal case.

Constraints on T. We now assume that all invariants in A are relaxed invariants as explained above. Furthermore, we assume that maximal gradients $l_{i,j}$ limiting the evolution of the arithmetic expressions $a_{i,j}$ during any period of continuous evolution are given. We require that the $l_{i,j}$ are derived from a set S of evolution constraints which limits the continuous evolution of the variables of A (cf. assumption 5 in Theorem 1). In order to provide that a sampling tick lies in every interval in which a transition in automaton A must be taken, we synthesize constraints on T from the (relaxed) invariants of A and the maximal gradients. Let ψ be a relaxed invariant of A and let $at'_{i,j} \equiv a_{i,j} \# \varepsilon_{i,j}$ be its (relaxed) atoms, as above. From each such atom we derive the constraint $T \leq \frac{\varepsilon_{i,j} - 2 \cdot \delta_{i,j}}{l_{i,j}}$, where $\delta_{i,j}$ is the error by which the computation of $a_{i,j}$ is affected (cf. Fig. 3). It results from the error ε_{dis} with which the continuous evolution of the variables is affected. For instance, the computation of expression $x + y$ is affected by the error in x plus that in y.

Correctness. Assumption 2 and 3 of Theorem 1 are satisfied, if B is derived as described, the constraints on T are satisfied and S is the set of evolution constraints used in Theorem 1. For assumption 3 we have to show that the invariant of A cannot be violated in a time step of duration T, if the invariant of the corresponding location of B is true before the step. In other words, no transition of A is enforced between two sampling instants where B can react. We prove this as follows. Let $v \in V$ and $s' \in \mathbb{R}^{n+1}$ with $s'.c = 0$ and $inv_B(v)$ is true for s'. By construction of the invariants of B this means that the ideal invariant ϕ of v must be true for $\pi_X(s')$. Let τ be an evolution of the variables which (1) satisfies the assumptions S on maximal gradients used for the computation of the $l_{i,j}$, which (2) starts with $\pi_X(s')$ and which (3) last for at least time T. We show that no atom $at'_{i,j}$ of the (relaxed) invariant ϕ of v in A becomes false for evolution τ on the interval $[0, T)$, if the corresponding ideal atom $at_{i,j}$ is true for $\pi_X(s')$. At time $t = 0$, $\tau(0) = \pi_X(s')$ and $at_{i,j}$ therefore holds by assumption, i.e. $a_{i,j} \# 0$, $\# \in \{<, \leq\}$. The worst case is that τ rises with its maximum gradient and that the errors in its computation ε_{dis} sum up. During time T, $a_{i,j}$ can only increase by $l_{i,j} \cdot T$ due to the assumptions on the maximal gradients. Thus, allowing for the worst case effect of $a_{i,j}$ being already above 0 by the worst case error $\delta_{i,j}$ at time 0 and being $\delta_{i,j}$ below the computed value at time T, we get that at time T the computed value of $a_{i,j}$ can at most be $\delta_{i,j} + l_{i,j} \cdot T + \delta_{i,j}$ (see Fig. 3). Provided the constraints on time granularity T are satisfied, this is still

less or equal to $\varepsilon_{i,j}$ for $t = T$ and strictly less than $\varepsilon_{i,j}$ for $0 < t < T$. Thus, $at'_{i,j}$ is true during the interval $[0, T)$. Applying this argument to all atoms that are true at $t = 0$, the relaxed invariant is guaranteed to remain true during that interval. (Remember that the relaxed invariant is a conjunction of disjunctions of atoms. Thus, one atom must have been true for $t = 0$ in every disjunction, if the relaxed invariant was true at that time.)

4.2.2 Update Transitions For the construction of the set E_u of update transitions, we suggest to use techniques from numerical mathematics or control theory. Numerical techniques for the solution of initial value problems can be used to approximate the variable evolution according to the activities in hybrid automaton A. In order to use such techniques in our context, we require that the respective numerical algorithm can be written as the iterated application of a jump condition at time steps kT, $k \in \mathbb{N}$. Thus, the values of the variables at time $(k+1)T$ must be computed from their values at time kT and this computation must be specifiable as a convex predicate p over variables $X \cup X'$, where $X' = \{x_1', ..., x_n'\}$, the x_i refer to the variable values at the last sampling step kT, $k \in \mathbb{N}$, and the x_i' refer to the values at the next step $(k+1)T$. For simple algorithms, like the Euler backward method or the trapezoidal method this is possible [8, 10]. Depending on the problem, more complex methods may also be applicable if Theorem 1 is modified by allowing extensions of the state space of B (see [12] for a more detailed discussion).

Definition of B's update transitions. Given a convex predicate p whose iterated application at time steps kT computes an approximation to the activity of location v in A, a corresponding update transition in B is defined by $(v, c = T \wedge p \wedge c' = 0, v)$. Set E_u of B consists of one such transition for each location v.

Constraints on T and applicable algorithms. In order to satisfy assumption 4 of Theorem 1 the algorithm used to define the update transition of a location v must satisfy some restrictions on its accuracy. First, the approximating values at time instants kT and the exact value according to the activity of v in A must agree up to ε_{dis}. In terms of numerical mathematics this means that T must be chosen small enough to guarantee that the so-called *global discretization error* is less or equal to ε_{dis}. In the context of control theory methods, Shannon's sampling theorem may also be applicable to obtain a further constraint on T. Second, the computed approximating value for time kT must be within $\varepsilon + \varepsilon_{dis}$ of the exact values throughout the following sampling interval $[kT, (k+1)T)$, where ε is the constant with which the traces of A are relaxed in Theorem 1. Provided a variable x evolves at most with gradient l, this means that $\varepsilon + \varepsilon_{dis} \leq l \cdot T$ must hold. Thus, the sampling rate T is constrained further due to the intersample error.

Note that if the activities of A merely specify a linear evolution of the variables in X, then simple numerical integration methods like Euler backward are exact. No discretization error occurs. This is utilized in the example in the appendix of the online version of this paper.

5 Conclusion

The paper summarized some of the results in [12]: It identified sufficient conditions under which an abstract specification of a hybrid system given by a hybrid automaton can be refined to execution in discrete-time. A method was described that allows us to derive the discrete time model from the hybrid model provided some restrictions are satisfied. In the development process of hybrid systems such a method can be applied to move from early abstract models of the system towards implementation.

For application is practice we suggest to build specializations of the general method in [12], which also regards input signals and events. Furthermore, we remark that automata which require a high precision w.r.t. the time when transitions are taken and w.r.t. their continuous activities can seriously hamper discrete-time refinement and hence implementation. From a methodological point of view, this suggests to avoid such precise models in early development steps.

Acknowledgment. We thank Jianjun Deng for his valuable feedback on a draft version of this paper.

References

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

[2] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional refinement of hierarchical hybrid systems. In *Proc. of HSCC'01*, LNCS 2034. Springer-Verlag, 2001.

[3] M. Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *Proc. of Computer Science Logic (CSL'99)*, LNCS 1683. Springer, 1999.

[4] R. Grosu, T. Stauner, and M. Broy. A modular visual model for hybrid systems. In *Proc. of FTRTFT'98*, LNCS 1486. Springer-Verlag, 1998.

[5] V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proc. of HART 97*, LNCS 1201. Springer-Verlag, 1997.

[6] T.A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *Proc. of HSCC'00*, LNCS 1790. Springer-Verlag, 2000.

[7] D. V. Hung and P. H. Giang. Sampling semantics of duration calculus. In *Proceedings of FTRTFT '96*, LNCS 1135. Springer-Verlag, 1996.

[8] M. K. Jain. *Numerical Solution of Differential Equations.* Wiley Eastern Ltd., New Delhi, 1979.

[9] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[10] K. Ogata. *Discrete-Time Control Systems.* Prentice-Hall, 1987.

[11] D. Sinclair. Using an object-oriented methodology to bring a hybrid system from initial concept to formal definition. In *Proc. Int. Workshop on Hybrid and Real-Time Systems (HART'97)*, LNCS 1201. Springer-Verlag, 1997.

[12] T. Stauner. *Systematic Development of Hybrid Systems.* PhD thesis, Technische Universität München, 2001.