From Duration Calculus To Linear Hybrid Automata*
(Extended Abstract)

Ahmed Bouajjani[1] ***, Yassine Lakhnech[2] **, and Riadh Robbana [1]***

[1] VERIMAG, Miniparc-Zirst, Rue Lavoisier 38330 Montbonnot St-Martin, France.
[2] Institut für Informatik und Praktische Mathematik Christian-Albrechts-Universität zu Kiel, Preußerstr. 1-9, D-24105 Kiel, Germany.

Abstract. We relate two different approaches for the specification and verification of hybrid systems. The first one is logic-based and uses the framework of the Calculus of Durations (CoD), the second one is automata-based and uses algorithmic analysis techniques for hybrid automata. Fragments of CoD have been identified in [13, 19] for the description of control systems and their requirements. We mainly show that the problem of verifying that a CoD control design satisfies a CoD requirement is decidable. This is proved by reducing this verification problem to the reachability problem for a subclass of linear hybrid automata where this problem is decidable.

1 Introduction

Since the last few years, methods for specifying and verifying hybrid systems are intensively investigated. Two of the prominent methods are the logic-based approach using the Calculus of Durations [10], and the automata-based approach using hybrid automata [17, 3, 18]. Our aim in this paper is to relate these two approaches.

CoD is an interval based temporal logic with a duration concept. The duration of a state property in some computation interval is the accumulated time the considered property holds. In the framework of CoD, both requirements and control systems are described by formulas. Special forms of CoD formulas have been identified for specifying control systems and their requirements [13, 19]. The fact that a control design satisfies a set of requirements is expressed as a logical entailment, which is proved using a deductive system for the validity of CoD formulas [10, 20, 19]. Almost all the existing decidability results for (dense time) CoD are negative; the unique positive result concerns the propositional fragment of CoD without timing nor duration constraints [12].

Hybrid automata are finite automata extended with a finite set of real-valued variables that change continuously at each control location. Transitions between locations are guarded by constraints on the variables and their execution may reset some of the variables. Several interesting subclasses of hybrid automata

* This work has been partially supported by ESPRIT/BRA Project 6021 (REACT)
** yl@informatik.uni-kiel.d400.de. This work has been performed while this author was visiting VERIMAG.
*** Ahmed.Bouajjani@imag.fr, Riadh.Robbana@imag.fr

have been identified in the literature according to the nature of their variables, as *linear hybrid automata* [3, 18] where the variables change at constant rates, *timed automata* [5, 1] where all the variables are clocks, i.e. their rates are always 1, or *integration automata* [3, 16] where the variables have rates 0 or 1. Several special cases of such automata have been identified for which the reachability problem is decidable [1, 4, 16, 14, 6, 8]. In the general case, which is undecidable [3, 16], methods based on approximation techniques have been proposed [2].

In this paper we show a result relating the CoD and the hybrid automata approaches. We prove that the problem of verifying that a CoD control system satisfies a set of CoD requirements (according to [13, 19]) can be reduced to the reachability problem of a class of integration automata for which this problem is decidable. This provides an algorithmic method for proving the satisfaction relation between control systems and requirements expressed in CoD.

Let us summerize how we obtain this result. Actually, we show a more general result for larger classes of control design and requirement specifications. We introduce a specification language, called DIL (Duration Interval Logic), that serves as an intermediate language between CoD and hybrid automata. DIL corresponds to the propositional fragment of CoD extended with *duration variables* introduced via a *reset quantification* that associates with each variable x a state formula whose duration is measured by x. Duration variables are used to express linear constraints on durations. We define two fragments of DIL, called CDF (Control Design Formulas) and RF (Requirement Formulas), used to specify control systems and requirements, respectively, and show that they strictly subsume their corresponding fragments of CoD that have been suggested in [13, 19].

Given a CDF formula φ_1 and an RF formula φ_2, we construct two integration automata $\mathcal{H}_1$ and $\mathcal{H}_2$ such that $\mathcal{H}_1$ recognizes the set of behaviours satisfying φ_1, and $\mathcal{H}_2$ recognizes the set of behaviours falsifying φ_2. Using these constructions, we reduce the problem of verifying that a given control system satisfies a given set of requirements, to the complement of a reachability problem in the product automaton of $\mathcal{H}_1$ and $\mathcal{H}_2$.

Then, we identify fragments of CDF and RF, called *pure-time CDF* and *simple RF*, for which our construction yields an automaton in a class of integration automata whose reachability problem can be shown to be decidable using the result in [4]. It turns out that the fragments of CoD suggested for specifying control systems and their requirements can be translated into pure-time CDF and simple RF formulas, respectively. This implies that the verification of CoD control systems w.r.t. CoD requirements is decidable. Actually, we also show that for any timed graph and any simple RF formula, the verification problem is decidable. Hence, we have identified a fragment of a linear-time duration temporal logic whose verification problem for timed graphs is decidable.

2 Preliminaries

Linear constraints Let $\mathcal{V}$ be a set of real valued variables. A *linear constraint* on $\mathcal{V}$ is a boolean combination of linear inequalities of the form $e(x_1, \cdots, x_n) \leq c$ where the x_i's are variables in $\mathcal{V}$, c is an integer constant, and $e(x_1, \cdots, x_n) = \sum_{i=1}^{n} k_i \cdot x_i$, the k_i's being integer constants. We denote by $\mathcal{C}_{\mathcal{V}}$ the set of all linear constraints on $\mathcal{V}$. A *valuation* of the variables in $\mathcal{V}$ is a function $\nu \in [\mathcal{V} \to I\!R]$. We denote by $\mathbf{0}$ the valuation that associates 0 with each variable. Given $X \subseteq \mathcal{V}$, we denote by $\nu[X \mapsto 0]$ the valuation which associates with the variables in X the value 0, and coincides with ν on the other variables. Given $g \in \mathcal{C}_{\mathcal{V}}$, we denote by $\nu \models g$ the fact that g is satisfied by ν. A *simple linear constraint* is a boolean combination of constraints of the form $e \leq c$ with e of the form x or $x - y$.

States and state formulas Let $\mathcal{P}$ be a finite set of atomic propositions. A *state* is a set of atomic propositions. Let Σ be the set of all possible states, i.e., $\Sigma = 2^{\mathcal{P}}$. A *state formula* is a boolean combination of atomic propositions. A satisfaction relation is defined as in propositional calculus between states and state formulas. We denote by $s \models \pi$ the fact that the state s satisfies the state formula π.

Trajectories We consider that time ranges over the set of non-negative reals. A *trajectory* is a mapping $\rho : I\!R_{\geq 0} \to \Sigma$ associating with each time value a state, such that in every finite time interval, ρ changes its value a finite number of times (ρ has a finite number of discontinuity points). This condition is called *finite-variability*. Given a trajectory ρ and a state formula π, the *characteristic function* of π w.r.t. ρ is the function $\widetilde{\rho}_\pi : I\!R_{\geq 0} \to \{0, 1\}$ with $\widetilde{\rho}_\pi(t) =$ if $\rho(t) \models \pi$ then 1 else 0. A trajectory ρ is *right-continuous* if $\forall t \in I\!R_{\geq 0}.\ \exists \epsilon > 0.\ \forall t' \in [t, t + \epsilon).\ \rho(t) = \rho(t')$. It can be seen that for every right-continuous (finitely variable) trajectory ρ, there exists a countable sequence $\{t_i\}_{i \in \omega}$ such that $t_0 = 0$, $\forall i \in \omega.\ t_{i+1} > t_i$, $\lim_{i\to\infty} t_i = \infty$, and $\forall t \in [t_i, t_{i+1}).\ \rho(t) = \rho(t_i)$.

3 Integration Automata

We present in this section Integration Automata (IA for short) that extend the integration graphs in [3, 18, 16] by a simple acceptance condition. The acceptance condition we consider corresponds to the 1-acceptance of the ω-automata theory [21]. This condition is convenient to express the complement of the reachability problem in integration graphs as an emptiness problem. Other kinds of acceptance conditions may be considered, they are, however, not relevant for the purpose of this paper.

Let $\mathcal{P}$ be a finite set of atomic propositions and $\Sigma = 2^{\mathcal{P}}$. An *integration automaton* $\mathcal{H}$ over Σ consists of the following components:

- $\mathcal{X}$, a finite set of variables,
- $\mathcal{Q}$, a finite set of control locations,
- $\mathcal{I}nit$, a set of initial control locations ($\mathcal{I}nit \subseteq \mathcal{Q}$),
- $\mathcal{A}cc$, a set of accepting locations,
- $\mathcal{E}$, a set of edges. Each edge is a tuple (q, g, X, q') where $q, q' \in \mathcal{Q}$ are the source and target locations, $g \in \mathcal{C}_{\mathcal{X}}$ is an enabling guard, and $X \subseteq \mathcal{X}$ is the set of reset variables,
- Π, a function in $[\mathcal{Q} \to \Sigma]$, associating a state with each location,
- Γ, a function in $[\mathcal{Q} \to \mathcal{C}_{\mathcal{X}}]$, associating with each location q an invariance condition under which the computation is allowed to stay in q,
- ∂, a function in $[\mathcal{Q} \times \mathcal{X} \to \{0, 1\}]$, associating with each $q \in \mathcal{Q}$ and $x \in \mathcal{X}$, a rate in $\{0, 1\}$ at which x changes continuously while the computation is at q.

In the sequel, we define the set of right-continuous trajectories accepted by an IA. This is done by means of the notions of a configuration and a run (configuration sequence) of an automaton.

A *configuration* of $\mathcal{H}$ is a triple $\langle q, \nu, t \rangle$ where $q \in \mathcal{Q}$, $\nu \in [\mathcal{X} \to I\!R]$ is a valuation of the variables, and $t \in I\!R_{\geq 0}$ is a time stamp. An *initial configuration* is a configuration $\langle q, \mathbf{0}, 0 \rangle$ where $q \in \mathcal{I}nit$. Given a valuation ν of the variables, a control location q, and $\tau \in I\!R_{\geq 0}$, we denote by $[\nu + \tau]_q$ the valuation ν' such that $\forall x \in \mathcal{X}.\ \nu'(x) = \nu(x) + \partial(q, x) \cdot \tau$, i.e., the valuation of the variables obtained from ν by staying at location q for an amount of time equal to τ.

A *run* of $\mathcal{H}$ starting from $q \in \mathcal{Q}$ is an infinite sequence $\{\langle q_i, \nu_i, t_i \rangle\}_{i \in \omega}$ such that $\langle q, \mathbf{0}, 0 \rangle = \langle q_0, \nu_0, t_0 \rangle$, $\lim_{i \to \infty} t_i = \infty$, and $\forall i \in I\!N$, the following conditions are satisfied:

- $t_{i+1} > t_i$, and $\forall t.\ t_i \leq t < t_{i+1}.\ [\nu_i + (t - t_i)]_{q_i} \models \Gamma(q_i)$, and
- either $q_i = q_{i+1}$ and $\nu_{i+1} = [\nu_i + (t_{i+1} - t_i)]_{q_i}$, or $\exists (q_i, g, X, q_{i+1}) \in \mathcal{E}.\ \nu_{i+1} = [\nu_i + (t_{i+1} - t_i)]_{q_i}[X \mapsto 0]$ and $[\nu_i + (t_{i+1} - t_i)]_{q_i} \models g$.

Every run $\{\langle q_i, \nu_i, t_i \rangle\}_{i \in \omega}$ generates a right-continuous trajectory ρ with $\forall t \in I\!R_{\geq 0}.\ \forall i \in \omega.\ t_i \leq t < t_{i+1} \Rightarrow \rho(t) = \Pi(q_i)$. Then, a right-continuous trajectory ρ is accepted by $\mathcal{H}$ if there exists a run $\{\langle q_i, \nu_i, t_i \rangle\}_{i \in \omega}$ of $\mathcal{H}$ starting from some initial control location which generates ρ and such that $\exists i \in \omega.\ q_i \in \mathcal{A}cc$. We denote by $\mathcal{T}(\mathcal{H})$ the set of (right-continuous) trajectories accepted by $\mathcal{H}$.

Particular cases A *clock* is a variable x such that for every location q, $\partial(q, x) = 1$. A *simple pure-time constraint* is a simple linear constraint where all variables are clocks. Then, a *timed automaton* is an integration automaton where all guards and invariants are simple pure-time constraints. A *simple integration automaton* is an IA which satisfies the following conditions:

- Each invariant is a simple pure-time constraint.
- Each guard is either a simple pure-time constraint or a conjunction of such constraints with one of the form $c \sim \sum_{i=1}^{n} k_i x_i \sim d$, where $\sim \in \{<, \leq\}$, $k_i \in I\!N$, and $c, d \in \mathbb{Z}$. (The x_i's are not necessarily clocks).
- On each path from any initial location to any accepting one, there is at most one guard that is not a simple pure-time constraint, moreover, each variable that is not a clock is reset only once and never tested before that reset.

Finally, an integration automaton (resp. timed automaton) is called integration graph (resp. timed graph), if all its locations are accepting (i.e., $\mathcal{A}cc = \mathcal{Q}$).

4 Duration Interval Logic

In this section, we define the interval temporal logic DIL which is essentially an extension of the propositional Calculus of Durations (CoD) [10] with a kind of quantification that allows to introduce variables which measure the duration of state formulas. This quantification has been introduced in [7] and is similar to the reset quantification of [15]. We assume given a set $\mathcal{P}$ of atomic propositions with P, Q, ... as typical elements, and a set *Var* of *duration variables*. We use x, y, ... to range over *Var* and $f, g, \cdots$ to range over the set of linear constraints on *Var*. These constraints are called *duration constraints*. We also use $\pi, \delta, \cdots$ to range over the set of state formulas.

The set of DIL formulas is defined by the following grammar:

$$\varphi ::= \Uparrow\pi \mid f \mid [x:\pi].\ \varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi;\varphi$$

In addition to the standard abbreviations introducing boolean connectives, we use the following abbreviations $\Diamond\varphi$ for $true;\varphi;true$ and $\Box\varphi$ for $\neg\Diamond\neg\varphi$. Furthermore, we write $[x_i : \pi_i]_{i=1}^n.\ \varphi$, or $[\mathbf{x} : \boldsymbol{\pi}].\ \varphi$, where $\mathbf{x} = x_1, \cdots, x_n$ and $\boldsymbol{\pi} = \pi_1, \cdots, \pi_n$, for $[x_1 : \pi_1] \cdots [x_n : \pi_n].\ \varphi$, and $x.\ \varphi$ for $[x : true].\ \varphi$, and $\ell \sim c$ for $x.\ x \sim c$. For a state formula π, the formula $\lceil\pi\rceil$ abbreviates $[x, y : true, \pi].\ x - y = 0 \wedge x > 0$.

DIL formulas describe properties of intervals in trajectories. An interval I satisfies $\Uparrow\pi$, if π changes its value from false to true exactly at the left end-point of I. This operator has been introduced in the Mean Value Calculus [11]. The operator ";" is the chop operator of the CoD.

In the formula $\phi = [x : \pi].\ \varphi$, the duration variable x is associated with the state formula π. Consider that ϕ is interpreted in some interval I, and let f be a duration constraint containing x which appears in φ. Then, when f is interpreted in a subinterval I' of I, x represents the duration of π from the left end-point of I to the right end-point of I'. For instance, the formula "$[x, y : \pi, \delta].\ (x < y; x = y)$" expresses the fact that there exists some prefix of the considered interval where the duration of π is less than the duration of δ and that the durations for π and δ in the entire interval are equal. Notice that a variable x which is associated with the formula $true$ counts the elapsed time since the point where it has been introduced. Such a variable is called a *clock*.
In the formula $\phi = [x : \pi].\ \varphi$, the construct "$[x : \pi]$" binds the variable x in φ. Without loss of generality, we assume that every variable is bound at most once in every formula. Then, every variable appearing in some formula is either bound or free. We denote by $\mathcal{F}(\varphi)$ the set of free variables in φ. A formula φ is closed if $\mathcal{F}(\varphi) = \emptyset$. Henceforth, whenever there is no danger of ambiguity, we refer to closed formulas simply as formulas.

Given a trajectory ρ, we define a satisfaction relation $\models$ between intervals in ρ and DIL formulas. Since DIL formulas may contain free variables, the satisfaction relation is parameterized by a *position association* $\mathcal{I}$ that associates with each duration variable the time at which it has been initialized (or bound), and a *state formula association* γ that associates with each variable x the state formula π whose duration is measured by x. Let F be either $\mathcal{I}$ or γ. Then, we denote by $F[v/x]$ the function which associates to x the object v and coincides with F on the other variables. Let φ be a DIL formula, ρ a trajectory, and $a, b \in I\!R_{\geq 0}$ with $b \geq a$. We denote by $(\rho, a, b) \models_{\mathcal{I},\gamma} \varphi$ the fact that the interval of ρ with end-points a and b satisfies φ; when φ is closed, the functions $\mathcal{I}$ and γ are omitted. We define $(\rho, a, b) \models_{\mathcal{I},\gamma} \varphi$ inductively on the structure of φ.

$(\rho, a, b) \models_{\mathcal{I},\gamma} \Uparrow\pi$ iff $\exists\epsilon > 0.\ \forall t \in (a, a+\epsilon).\ \widetilde{\rho}_\pi(t) = 1$, and either $a = 0$ or $\forall t \in (a-\epsilon, a).\ \widetilde{\rho}_\pi(t) = 0$,
$(\rho, a, b) \models_{\mathcal{I},\gamma} f$ iff $\nu \models f$ where ν satisfies $\forall x \in Var.\nu(x) = \int_{\mathcal{I}(x)}^{b} \widetilde{\rho}_{\gamma(x)}(t)\, dt$,
$(\rho, a, b) \models_{\mathcal{I},\gamma} [x : \pi].\varphi$ iff $(\rho, a, b) \models_{\mathcal{I}',\gamma'} \varphi$ where $\mathcal{I}' = \mathcal{I}[a/x]$ and $\gamma' = \gamma[\pi/x]$,
$(\rho, a, b) \models_{\mathcal{I},\gamma} \neg\varphi$ iff $(\rho, a, b) \not\models_{\mathcal{I},\gamma} \varphi$
$(\rho, a, b) \models_{\mathcal{I},\gamma} \varphi_1 \vee \varphi_2$ iff $(\rho, a, b) \models_{\mathcal{I},\gamma} \varphi_1$ or $(\rho, a, b) \models_{\mathcal{I},\gamma} \varphi_2$,

$(\rho, a, b) \models_{\mathcal{I},\gamma} \varphi_1; \varphi_2$ iff $\exists m \in [a, b].\ (\rho, a, m) \models_{\mathcal{I},\gamma} \varphi_1$ and $(\rho, m, b) \models_{\mathcal{I},\gamma} \varphi_2$.

We define a satisfaction relation between trajectories and closed DIL formulas. A trajectory ρ *satisfies* a closed formula φ iff $\forall t \geq 0.\ (\rho, 0, t) \models \varphi$. We denote by $[\![\varphi]\!]$ (resp. $[\![\varphi]\!]_{rc}$) the set of trajectories (resp. right-continuous trajectories) satisfying φ. A DIL formula φ is called *satisfiable*, if there exists a trajectory that satisfies φ. It is called *valid*, if every trajectory satisfies φ. Two DIL formulas φ and φ' are called *equivalent*, if $[\![\varphi]\!] = [\![\varphi']\!]$. They are called *congruent*, if $\Box(\varphi \Leftrightarrow \varphi')$ is valid.

Let us discuss the expressiveness of DIL and compare it with linear CoD, i.e., CoD with formulas of the form $\sum_{i=1}^{n} k_i \cdot \int \pi_i \sim c$ where the k_i's and c are integer constants, as primitives. From now on, we only refer to the linear CoD and write simply CoD. We write also PCoD for the Propositional CoD.

First of all, it can be seen that PCoD corresponds exactly to the fragment of DIL that contains formulas without "$\uparrow$" operator and such that, between every occurrence of a variable x and its binding (i.e. "$[x : \pi]$"), there exists no chop operator ";". That is, CoD formulas of the form $e(\int \pi_1, \cdots, \int \pi_n) \sim c$ can be written in DIL as $[x_i : \pi_i]_{i=1}^{n}.\ e(x_1, \cdots, x_n) \sim c$. Hence, the full CoD can be translated into first-order DIL (DIL extended with first order quantification). On the other hand, we can show that each formula in DIL can be translated into CoD using first-order quantification. In fact, DIL is strictly more expressive than PCoD since the construct "$[x : \pi]$" is a kind of quantification that allows the comparison of durations of state formulas measured in different intervals. For instance, the formula $\varphi = [x : \pi].\ (x \leq 1; [y : \delta].\ x - y < 0)$ is not expressible in PCoD. It is equivalent, however, to the CoD formula $\forall x.\ (x = \int \pi) \Rightarrow (\int \pi \leq 1; x - \int \delta < 0)$. Then, we have the following strict inclusions: PCoD $\subset$ DIL $\subset$ CoD.

5 Control Design and Requirement Specification

We define in this section two subsets of DIL, called CDF (Control Design Formulas) and RF (Requirement Formulas) and show that they subsume the CoD fragments that have been identified in [13, 19] for the description of control designs and requirements of hybrid systems.

Formulas in CDF as well as in RF are positive boolean combinations of invariance formulas of the form $\Box(\varphi \Rightarrow \varphi')$ where φ and φ' satisfy some specific conditions we define below. To do so, we first introduce a positive fragment of DIL, called DIL$^+$. The set of DIL$^+$ formulas is defined by the following grammar:

$$\varphi ::= \neg{\uparrow}\pi \mid {\uparrow}\pi \mid \lceil \pi \rceil \mid f \mid [x : \pi].\ \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi; \varphi$$

We define a *canonical form* for DIL$^+$ formulas. It corresponds to the set of formulas described by the following grammar:

$$\begin{aligned}
\varphi &::= \bigvee \phi \\
\phi &::= [\mathbf{x} : \boldsymbol{\pi}].\ ((\beta \wedge \xi); \phi) \mid [\mathbf{x} : \boldsymbol{\pi}].\ (\beta \wedge \xi) \\
\beta &::= \mu \mid \mu; \beta \\
\mu &::= \{\textstyle\bigwedge {\uparrow}\pi \wedge\}\{\textstyle\bigwedge \neg{\uparrow}\pi \wedge\}\lceil \pi \rceil \mid true \\
\xi &::= \textstyle\bigwedge e \sim c, \sim \in \{<, >, \leq, \geq\}
\end{aligned}$$

We can prove that every DIL$^+$ formula has a congruent formula in canonical form. The proof uses straightforward laws relating boolean connectives and the chop operator; mainly the fact that the conjunction of two chop formulas is equivalent to a disjunction of chop formulas.

Proposition 1. *Every DIL$^+$ formula has a congruent formula in canonical form.*

Now, we define subclasses of DIL$^+$ formulas by introducing conditions on their canonical form. For that, we need the following notation. Given a formula of the form μ introduced above, we define $\Phi(\mu) =$ (if $\mu = \{\bigwedge \uparrow\pi \wedge\}\{\bigwedge \neg\uparrow\pi'\wedge\}\lceil\pi''\rceil$ then $\{\bigwedge \pi\wedge\}\pi''$ else $true$).

Let ϕ be a DIL$^+$ formula in canonical form given by

$$[\mathbf{x}_1 : \boldsymbol{\pi}_1].\ ((\mu_1 \wedge \xi_1); \cdots [\mathbf{x}_j : \boldsymbol{\pi}_j].\ ((\mu_j \wedge \xi_j); \cdots [\mathbf{x}_n : \boldsymbol{\pi}_n].\ (\mu_n \wedge \xi_n) \cdots)) \cdots) \quad (1)$$

The first subclass we consider, consists of what we call *tail-constraint-free* formulas, which are defined as formulas of the form (1) with $\mu_n = true$ and $\xi_n = true$. It can be seen that for every tail-constraint-free formula, if it is satisfied by some interval $[a, b]$, then it is satisfied by every longer interval $[a, d]$.

Now, we define the class of π-*alternation-free* formulas, which is a subset of the class of tail-constraint-free formulas. Intuitively, a formula ϕ in the form of (1) is π-alternation-free, if for every interval satisfying $\Phi(\mu_1); \cdots; \Phi(\mu_{n-1})$, the formula $\uparrow\pi$ can only be true at the beginning of the interval. The formal definition is as follows. We say that a tail-constraint-free formula ϕ (1) is π-*alternation-free* if $\forall j \in \{1, \cdots, n-1\}$, the conjunction $\pi \wedge \Phi(\mu_j)$ is either equivalent to $false$ or to $\Phi(\mu_j)$, and furthermore, $\pi \wedge \Phi(\mu_j) \neq false$ implies that $\forall k.\ 1 \leq k < j$, $\pi \wedge \Phi(\mu_k) \neq false$.

The next subclass of DIL$^+$ formulas we define is the class of *overlap-free formulas*. Consider a formula ϕ of the form (1), and an interval I that satisfies ϕ. From the semantics of DIL, there are n points in the interval I that delimit subintervals where the formulas $\mu_i \wedge \xi_i$ are satisfied. Intuitively, the overlap-free condition implies that the choice of such points is unique, except for the last one when ϕ is tail-constraint-free. Then, we have the following definition. A formula ϕ of the form (1) is called *overlap-free* if $\forall j \in \{1, \cdots, n-1\}$, one of the following conditions holds:

1. $\Phi(\mu_j) \wedge \Phi(\mu_{j+1}) = false$, or
2. $j = n - 1$, and ϕ is tail-constraint-free, or
3. ξ_j is equivalent to $\xi \wedge (x = c)$, for some ξ, some clock x in $\mathbf{x}_j$, and some constant c.

Finally, we introduce a subclass of overlap-free formulas. An overlap-free formula ϕ of the form (1) is called *strongly overlap-free*, if ϕ is either not tail-constraint-free, or ξ_{n-1} is equivalent to true, or ξ_{n-1} is equivalent $\xi \wedge (x = c)$, for some ξ, some clock x in $\mathbf{x}_j$, and some constant c.

A formula $\bigvee_i \phi_i$ in canonical form is called *overlap-free* (resp. *strongly overlap-free*, *tail-constraint-free*, π-*alternation-free*) if each ϕ_i is overlap-free (resp. strongly overlap-free, tail-constraint-free, π-alternation-free).

We are now able to define the two classes of DIL formulas corresponding to control designs and requirements.

Control Design Formulas A *control design formula* (CDF for short) is a conjunction of formulas of the form $\Box((\uparrow\pi \wedge (\ell \sim c)) \Rightarrow \varphi)$, where $\sim \in \{>, \geq\}$, and φ is a DIL$^+$ formula in canonical form that is overlap-free, and π-alternation-free.

Requirement Formulas A *requirement formula* (RF for short) is a disjunction of conjunctions of formulas of the form $\Box(\varphi \Rightarrow \varphi')$, where φ and φ' are DIL$^+$ formulas such that φ' is in canonical form and strongly overlap-free.

Particular cases We call *simple clock-constraint* any simple linear constraint which contains only clocks. A *lower-bound* duration constraint (lbd-constraint, for short), is any constraint of the form $\sum_{i=1}^{n} k_i x_i \sim d$, where $\sim \in \{>, \geq\}$, $k_i \in \mathbb{N}$, and $d \in \mathbb{N}$. Then, we call *pure-time CDF formula*, any CDF formula such that all its duration constraints are simple clock-constraints. Moreover, a *simple RF formula* is a conjunction of RF formulas of the form $\Box(\phi \Rightarrow \bigvee_{i=1}^{n} \phi_i)$ such that all the duration constraints appearing in the formulas $\phi, \phi_1, \cdots, \phi_n$ are simple clock-constraints, except at most one, which can be a conjunction of lbd-constraints, and appears only once in one of the formulas $\phi_1, \cdots, \phi_n$.

In [13, 19], CoD formulas have been identified for specifying control designs and requirements. It is not difficult to show that the properties expressible by these formulas can be also expressed in CDF and RF respectively. In the full paper, we show the following result.

Proposition 2. *For every CoD control design (resp. CoD requirement) there exists an equivalent pure-time CDF (resp. simple RF) formula.*

6 The Verification Problem

If D is the set of trajectories of a system (control design) and R a set of trajectories expressing the requirements, the verification problem consists on checking whether $D \subseteq R$ holds. We show in this section that when D and R are represented as two formulas φ_D and φ_R in CDF and RF respectively, the verification problem $[\![\varphi_D]\!] \subseteq [\![\varphi_R]\!]$ reduces to the emptiness problem in IA's.

Let us present the steps of our reduction of the verification problem to the emptiness problem of IA's. Remember that we have defined the languages recognized by IA's as sets of right-continuous trajectories. Therefore, we first show that the verification problem $[\![\varphi_D]\!] \subseteq [\![\varphi_R]\!]$ can be solved by considering only right-continuous trajectories.

By induction on the structure of DIL formulas, we can show that the interpretation of a DIL formula in some interval of some trajectory does not depend on isolated points. Therefore, for every trajectory ρ we can find a right-continuous trajectory ρ' such that the interpretation of each DIL formula in each interval along ρ or along ρ' leads to the same value. Consequently, we have the following result.

Proposition 3. *For every DIL formulas φ and φ', $[\![\varphi]\!] \subseteq [\![\varphi']\!]$ iff $[\![\varphi]\!]_{rc} \subseteq [\![\varphi']\!]_{rc}$.*

Then, we construct two IA's $\mathcal{H}_1$ and $\mathcal{H}_2$ such that $[\![\varphi_D]\!]_{rc} = \mathcal{T}(\mathcal{H}_1)$ and $\overline{[\![\varphi_R]\!]_{rc}} = \mathcal{T}(\mathcal{H}_2)$, where $\overline{[\![\varphi]\!]_{rc}}$ denotes the set of right-continuous trajectories which do not satisfy φ. So, we have $[\![\varphi_D]\!]_{rc} \subseteq [\![\varphi_R]\!]_{rc}$ iff $\mathcal{T}(\mathcal{H}_1) \cap \mathcal{T}(\mathcal{H}_2) = \emptyset$, which is equivalent to the emptiness of the product automaton of $\mathcal{H}_1$ and $\mathcal{H}_2$.

6.1 Control Design Formulas

We present in this subsection the construction of an IA recognizing all the right-continuous trajectories satisfying some given CDF formula. Actually, it suffices to consider only CDF formulas which are conjunctions of formulas of the form $\Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \phi)$, since we can show that every CDF formula is equivalent to another one in this form.

First of all, we define for every CDF formula φ an equivalent formula denoted φ^* which is suitable for the construction of an IA recognizing $[\![\varphi]\!]_{rc}$.

Let us start by considering the simpler case of a CDF formula φ of the form $\Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \phi)$, where ϕ is given by

$$[\mathbf{x}_1 : \boldsymbol{\pi}_1].\ (\underbrace{(\mu_1 \wedge \xi_1)}_{\text{Phase 1}}; \cdots [\mathbf{x}_n : \boldsymbol{\pi}_n].\ (\underbrace{(\mu_n \wedge \xi_n)}_{\text{Phase } n}; true) \cdots) \tag{2}$$

Let ρ be a trajectory and $a \in I\!\!R_{\geq 0}$ a time point where $\uparrow\pi$ holds. By definition, if ρ satisfies the formula φ, then ϕ is satisfied by every interval $[a, b]$ such that $b > a + c$. Let us suppose first that $c > 0$. Since ϕ is overlap-free, only three situations may arise:

1. The phase n starts (strictly) before and ends before or at $a + c$.
2. The phase n starts (strictly) before and ends (strictly) after $a + c$.
3. The phase n starts exactly at $a + c$.

Then, given an overlap-free formula ϕ (2), we define a formula $\phi^*_{>c}$ which reflects the situations above. First of all, when $c = 0$, $\phi^*_{>c} =$ if $n > 1$ then $false$ else ϕ. For, $c > 0$, the formula $\phi^*_{>c}$ is defined as follows. Let $\phi_t = t.\ \phi$ and suppose $n > 1$ (if $n = 0$ then $\phi^*_{>c} = true$). Given a constant c, we define the formula $\phi^*_{>c}$ as a disjunction of three formulas ϕ_1, ϕ_2, and ϕ_3 such that

1. ϕ_1 is obtained from ϕ_t by replacing ξ_n by $\xi_n \wedge (t \leq c)$,
2. ϕ_2 is obtained from ϕ_t by replacing $[\mathbf{x}_n : \boldsymbol{\pi}_n].\ (\mu_n \wedge \xi_n; true)$ by $[\mathbf{x}_n : \boldsymbol{\pi}_n].\ (\mu_n \wedge (t = c); \mu_n \wedge \xi_n \wedge (t > c); true)$, and
3. ϕ_3 is obtained from ϕ_t by replacing ξ_{n-1} by $\xi_{n-1} \wedge (t = c)$, and ξ_n by $\xi_n \wedge (t > c)$. (If $n = 1$ the disjunct ϕ_3 is omitted.)

Then, we have the following result.

Lemma 4. *Let ϕ be an overlap-free of the form (2), ρ a trajectory, and $a \in I\!\!R_{\geq 0}$. Then, $\forall b > a + c.\ (\rho, a, b) \models \phi$ iff $\forall b > a + c.\ (\rho, a, b) \models \phi^*_{>c}$.*

Consequently, we obtain $[\![\Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \phi)]\!]_{(rc)} = [\![\Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \phi^*_{>c})]\!]_{(rc)}$. We consider now the more general case of a CDF formula $\varphi = \Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \bigvee_{i=1}^{n} \phi_i)$. Let us denote by φ^* the formula $\Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \bigvee_{i=1}^{n} (\phi_i)^*_{>c})$. This definition generalizes straightforwardly to conjunctions of formulas.

We prove hereafter that the formulas φ and φ^* characterize the same sets of trajectories. It is not difficult to see that each interval that satisfies $\phi^*_{>c}$, satisfies ϕ too. Therefore, $[\![\varphi^*]\!] \subseteq [\![\varphi]\!]$. The proof of the inclusion $[\![\varphi]\!] \subseteq [\![\varphi^*]\!]$ is sketched next. It can be shown that because of the finite-variability condition and since duration constraints are linear, for each overlap-free formula ϕ and $a, c \in I\!\!R_{\geq 0}$ with $a > c$ there exists $b > a + c$ such that either $\forall b' \in (a + c, b).\ (\rho, a, b') \models \phi$ or $\forall b' \in (a + c, b).\ (\rho, a, b') \not\models \phi$. Therefore, if the disjunction $\bigvee_{i=1}^{n} \phi_i$ is satisfied by every interval $[a, b]$ with $b > a + c$, there exists necessarily some $\phi \in \{\phi_1, \cdots, \phi_n\}$ and $d > a + c$, such that every interval $[a, b']$ with $b' \in (a + c, d)$ satisfies ϕ. Since the ϕ_i's are tail-constraint-free, this implies that every interval $[a,b]$ of length $> c$ satisfies ϕ. Using Lemma 4, we deduce that every interval $[a,b]$ of length $> c$ satisfies $\phi^*_{>c}$. Hence, we obtain that *varphi* and φ^* are equivalent. Finally, it can be seen that for every DIL formulas φ and φ', $[\![\varphi \wedge \varphi']\!] = [\![\varphi]\!] \cap [\![\varphi']\!]$. Then, we have the following result.

Lemma 5. *For every CDF formula* φ, $[\![\varphi]\!]_{(rc)} = [\![\varphi^*]\!]_{(rc)}$.

Consider a CDF formula $\varphi = \Box(\uparrow\!\pi \wedge (\ell > c) \Rightarrow \bigvee_{i=1}^{n} \phi_i)$, $\varphi^* = \Box(\uparrow\!\pi \wedge (\ell > c) \Rightarrow \bigvee_{i=1}^{m} \phi'_i)$, and a trajectory ρ. By Lemma 5, proving that ρ satisfies φ is equivalent to proving that ρ satisfies φ^*. As it is argued in the justification of this lemma, if ρ satisfies φ, then for every point a where $\uparrow\!\pi$ holds, there exists a $\phi \in \{\phi_1, \cdots, \phi_n\}$ and some point $d > a + c$, such that $\phi^*_{>c}$ is satisfied by every interval $[a,b]$ with $b \in (a+c, d)$. Remember that $\phi^*_{>c}$ may be a disjunction of three formulas (when $c > 0$), and notice that in this case, these formulas are mutually exclusive. Thus, we deduce that, in order to verify that some trajectory ρ satisfies φ, we have to find for every point a where $\uparrow\!\pi$ holds, some $\phi' \in \{\phi'_1, \cdots, \phi'_m\}$ and $d > a+c$, such that ϕ' is satisfied by every interval $[a,b]$ with $b \in (a+c, d)$. Now, for a fixed formula ϕ' and point d, we have to check the truth of ϕ for infinitely many intervals, namely all intervals $[a,b]$ with $b \in (a+c, d)$. The solution to this problem is given by the next lemma which says that it suffices to consider only the interval $[a,d]$ and check a stronger property that is described in the next lemma.

Lemma 6. *Let* φ *be a CDF formula with* $\varphi^* = \Box(\uparrow\!\pi \wedge (\ell > c) \Rightarrow \bigvee_{i=1}^{m} \phi_i)$, *and* ρ *a right-continuous trajectory. Then* ρ *satisfies* φ^* *iff for every* $a, b \in \mathbb{R}_{\geq 0}$ *such that* $(\rho, a, b) \models \uparrow\!\pi$, *there exist* $d \in \mathbb{R}_{\geq 0}$ *with* $d > a + c$, *and* $\phi \in \{\phi_1, \cdots, \phi_m\}$ *such that*

- *if* $\phi = t.\, [\mathbf{x}_1 : \boldsymbol{\pi}_1].\, (\cdots [\mathbf{x}_n : \boldsymbol{\pi}_n].\, (\mu_n \wedge \xi_n \wedge (t \leq c); true) \cdots)$, *then there exist* $a_1, \cdots, a_{n+1} \in \mathbb{R}_{\geq 0}$ *with* $a = a_1$, *a position association* $\mathcal{I}$ *satisfying* $\mathcal{I}(t) = a_1$ *and* $\forall j \in \{1, \cdots, n\}.\, \forall x \in \mathbf{x}_j.\, \mathcal{I}(x) = a_j$, *and a state formula association* γ *satisfying* $\forall j \in \{1, \cdots, n\}.\, \gamma(\mathbf{x}_j) = \boldsymbol{\pi}_j$ *such that*
  1. $\forall j \in \{1, \cdots, n\}.\, (\rho, a_j, a_{j+1}) \models_{\mathcal{I},\gamma} \mu_j \wedge \xi_j$, *and*
  2. $(\rho, a_n, a_{n+1}) \models_{\mathcal{I},\gamma} t \leq c$,
- *if* $\phi = t.\, [\mathbf{x}_1 : \boldsymbol{\pi}_1].\, (\cdots [\mathbf{x}_n : \boldsymbol{\pi}_n].\, (\mu_n \wedge (t = c); \mu_n \wedge \xi_n \wedge (t > c); true) \cdots)$, *then there exist* $a_1, \cdots, a_{n+1} \in \mathbb{R}_{\geq 0}$ *with* $a = a_1$, $\mathcal{I}$ *satisfying* $\mathcal{I}(t) = a_1$ *and* $\forall j \in \{1, \cdots, n\}.\, \forall x \in \mathbf{x}_j.\, \mathcal{I}(x) = a_j$, *and* γ *satisfying* $\forall j \in \{1, \cdots, n\}.\, \gamma(\mathbf{x}_j) = \boldsymbol{\pi}_j$ *such that*
  1. $\forall j \in \{1, \cdots, n-1\}.\, (\rho, a_j, a_{j+1}) \models_{\mathcal{I},\gamma} \mu_j \wedge \xi_j$,
  2. $(\rho, a_n, a_{n+1}) \models_{\mathcal{I},\gamma} \mu_n \wedge (t = c)$, *and*
  3. $\forall u \in (a_{n+1}, d).\, (\rho, a_{n+1}, u) \models_{\mathcal{I},\gamma} \mu_n \wedge \xi_n$,
- *if* $\phi = t.\, [\mathbf{x}_1 : \boldsymbol{\pi}_1].\, (\cdots [\mathbf{x}_{n-1} : \boldsymbol{\pi}_{n-1}].\, (\mu_{n-1} \wedge \xi_{n-1} \wedge (t = c); [\mathbf{x}_n : \boldsymbol{\pi}_n].\, \mu_n \wedge \xi_n \wedge (t > c); true) \cdots)$, *then there exist* $a_1, \cdots, a_n \in \mathbb{R}_{\geq 0}$ *with* $a = a_1$, $\mathcal{I}$ *satisfying* $\mathcal{I}(t) = a_1$ *and* $\forall j \in \{1, \cdots, n\}.\, \forall x \in \mathbf{x}_j.\, \mathcal{I}(x) = a_j$, *and* γ *satisfying* $\forall j \in \{1, \cdots, n\}.\, \gamma(\mathbf{x}_j) = \boldsymbol{\pi}_j$ *such that*
  1. $\forall j \in \{1, \cdots, n-1\}.\, (\rho, a_j, a_{j+1}) \models_{\mathcal{I},\gamma} \mu_j \wedge \xi_j$,
  2. $(\rho, a_{n-1}, a_n) \models_{\mathcal{I},\gamma} t = c$, *and*

3. $\forall u \in (a_n, d).\, (\rho, a_n, u) \models_{\mathcal{I},\gamma} \mu_n \wedge \xi_n$,

Hereafter, using Lemma 6, we show that for a given CDF formula φ with $\varphi^* = \Box(\uparrow\pi \wedge (\ell > c) \Rightarrow \bigvee_{i=1}^{m} \phi_i)$, we can construct an automaton $\mathcal{H}$ that recognizes $[\![\varphi]\!]_{rc}$. For sake of presentation, we assume that the ϕ_i's do not contain subformulas of the form $\uparrow\delta$. It is not difficult to see how to handle the general case. This automaton consists of two parts. Within the first one, which we call the *waiting part*, it looks for the next point at which $\uparrow\pi$ holds. Once such a time point a is detected, the computation of $\mathcal{H}$ enters the second part of the automaton, called the *guessing part*, where it nondeterministically guesses a $\phi \in \{\phi_1, \cdots, \phi_m\}$ and a $d > a + c$, such that one of the situations described by Lemma 6 applies to the interval $[a, d]$ and ϕ. The π-alternation-free condition ensures that during this procedure, no other points where $\uparrow\pi$ holds can occur. When this guessing procedure terminates, the computation returns back to the waiting part of $\mathcal{H}$, unless some new point where $\uparrow\pi$ holds is immediately detected, and in this case, it reenters directly the guessing part.

To explain the construction of $\mathcal{H}$ in more details, we first introduce some notions. Given a state formula π we call a location q a π-location, if $\Pi(q) \models \pi$, and we call a transition π-dangerous, if it leads from a $\neg\pi$-location to a π-location.

The initial locations of $\mathcal{H}$ are the $\neg\pi$-locations of the waiting part, and the target locations of the π-dangerous transitions, in the guessing part. All locations are accepting, hence $\mathcal{H}$ is an integration graph.

The waiting part contains for each state $s \in 2^{\mathcal{P}}$ a location labeled by s and with true as invariant. All the transitions between these locations are allowed except π-dangerous ones. The rates at which the variables change in this part of the automaton are not relevant. Transitions that are π-dangerous lead from the waiting to the guessing part. The structure of the guessing part is induced by the formulas ϕ_i appearing in φ^*. Consider for instance the following formula:

$$t.\,[\mathbf{x}_1 : \boldsymbol{\pi}_1].\,(\underbrace{(\lceil\delta_1\rceil \wedge \xi_1)}_{\text{Phase }1};\cdots[\mathbf{x}_n : \boldsymbol{\pi}_n].\,\underbrace{(\lceil\delta_n\rceil \wedge (t = c)}_{\text{Phase }n};\underbrace{\lceil\delta_n\rceil \wedge \xi_n \wedge (t > c)}_{\text{Phase }n+1};true)\cdots$$

The guessing part associated with this formula is represented in figure 1. It is constructed in the following way: we associate with each phase i, with $i \in \{1, \cdots, n\}$, a complete graph where the locations are labeled by all the possible states satisfying the corresponding state formula δ_i. To the phase $n+1$, we associate two such graphs labeled by either $\delta_n \wedge \pi$ or $\delta_n \wedge \neg\pi$. The rate of each variable x which is bound to δ, is 1 (resp. 0) in every δ-location (resp. $\neg\delta$-location).

This guessing part is connected with the waiting part as follows: each location in the graph labeled by $\delta_n \wedge \pi$ has a transition to every location in the waiting part of the automaton, the locations in the graph labeled by $\delta_n \wedge \neg\pi$ are connected to $\neg\pi$-locations of the waiting part, and to all the locations in the graph of phase 1.

We have explained the construction of the guessing part corresponding to one of the possible forms that ϕ can take. The other forms can be considered in a similar way. Then, we have the following result.

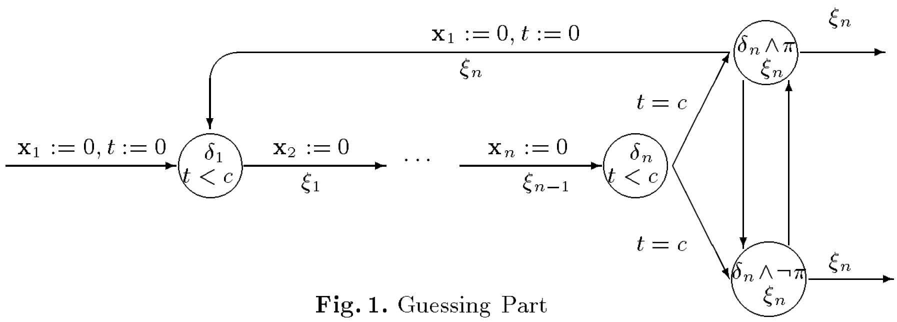

Fig. 1. Guessing Part

Theorem 7. *For every CDF formula φ, we can effectively construct an integration graph $\mathcal{H}$ such that $\mathcal{T}(\mathcal{H}) = [\![\varphi]\!]_{rc}$.*

6.2 Requirement Formulas

We show in this subsection that for every RF formula φ, we can construct an IA recognizing all the right-continuous trajectories which do not satisfy φ.

First of all, we have the following fact, which is to prove.

Lemma 8. *For every DIL formula $\varphi = \bigvee\limits_{i=1}^{n} \bigwedge\limits_{j=1}^{m_i} \Box\varphi_i^j$, $\overline{[\![\varphi]\!]}_{rc} = \bigcap_{i=1}^{n} \bigcup_{j=1}^{m_i} \overline{[\![\Box\varphi_i^j]\!]}_{rc}$.*

Next, we give the key lemma behind our construction. It says that the class of strongly overlap-free formulas is closed under complementation.

Lemma 9. *For every strongly overlap-free DIL$^+$ formula φ, there exists a strongly overlap free DIL$^+$ formula which is congruent to $\neg\varphi$.*

Then, our construction is based on the fact that, given any DIL$^+$ formula, we can define an integration automaton which, for every trajectory ρ, guesses nondeterministically the beginning of some interval in ρ which satisfies φ. When such an interval is found, the automaton enters an accepting location. Then, we have the following result.

Proposition 10. *For every DIL$^+$ formula φ, we can effectively construct an IA $\mathcal{H}$ such that $\mathcal{T}(\mathcal{H})$ is the set of right-continuous trajectories ρ such that $\exists a, b \in \mathbb{R}_{\geq 0}.\ (\rho, a, b) \models \varphi$.*

To illustrate how the automaton $\mathcal{H}$ is constructed, consider a formula $\phi = [\mathbf{x}_1 : \boldsymbol{\pi}_1].\ (([\delta_1] \wedge \xi_1); \cdots [\mathbf{x}_n : \boldsymbol{\pi}_n].\ ([\delta_n] \wedge \xi_n)) \cdots)$. The corresponding automaton $\mathcal{H}$ is represented in figure 2. It has one accepting location, marked by a double circle. Its initial states are marked by vertical arrows. For all locations of $\mathcal{H}$, we have $\Gamma(q) = true$, and for each variable x that is bounded to π, $\partial(q, x) = 1$ if $\Pi(q) \models \pi$, and $\partial(q, x) = 0$ otherwise.

Now, let $\varphi = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \Box(\varphi_{(i,j)} \Rightarrow \varphi'_{(i,j)})$ be a RF formula. By Lemma 8, $\overline{[\![\varphi]\!]}_{rc} = \bigcap_{i=1}^{n} \bigcup_{j=1}^{m_i} \mathcal{T}_i^j$, where the $\mathcal{T}_i^j$'s are the sets of right-continuous trajectories ρ such that $\exists a, b \in \mathbb{R}_{\geq 0}.\ (\rho, a, b) \models \varphi_{(i,j)} \wedge \neg\varphi'_{(i,j)}$, and moreover, since the $\varphi_{(i,j)}$'s are DIL$^+$ formulas, and by Lemma 9, the $\neg\varphi'_{(i,j)}$'s are congruent to DIL$^+$ formulas, then all the $\varphi_{(i,j)} \wedge \neg\varphi'_{(i,j)}$'s are congruent to DIL$^+$ formulas. Therefore, using Proposition 10, we obtain the following result.

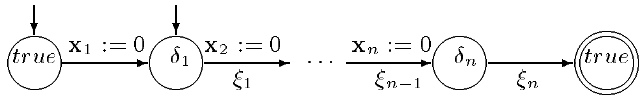

Fig. 2. The automaton of the set of trajectories having an interval satisfying ϕ.

Theorem 11. *For every RF formula φ, we can effectively construct an integration automaton $\mathcal{H}$ such that $\mathcal{T}(\mathcal{H}) = \overline{[\![\varphi]\!]_{rc}}$.*

6.3 Discussion

The constructions we have presented previously are based on some conditions we have considered in the definitions of CDF and RF formulas. We show in the full paper that without these conditions, such constructions are in general impossible. Indeed, we exhibit examples of formulas that violate these conditions, and argue that for these formulas, we need automata with infinitely many variables. The examples we provide are inspired by an example given in [5] to show that the class of regular timed languages is not closed under complementation. In this extended abstract, we only consider the π-alternation-freeness condition. Consider the formula $\varphi = \Box(\uparrow\pi \wedge (\ell > 1) \Rightarrow (\lceil true \rceil \wedge \ell = 1); \lceil \neg\pi \rceil; true)$.
Clearly, φ is not in CDF since it is not π-alternation-free. It expresses the property that whenever π changes from false to true, then after exactly one time unit, π is false. An automaton that recognizes this property must be able to keep track of all time points where $\uparrow\pi$ holds in the last time unit. Since in an interval of one time unit, the number of such time points is unbounded, an infinite number of clocks is needed.

7 Decidability Results

We showed in the previous section that the verification problem of CDF formulas w.r.t. RF formulas, can be reduced to the emptiness problem of integration automata. The automata we consider have the 1-acceptance condition, and hence, their emptiness problem is equivalent to the complement of the reachability problem of the integration graphs. This reachability problem is undecidable in general [9, 14], nevertheless, some subclasses of integration graphs have been identified for which this problem is decidable [1, 4, 16, 6]. Using some of these results, we show that the verification problem of pure-time CDF formulas w.r.t. simple RF formulas is decidable. This implies that the verification problem of CoD control designs w.r.t. CoD requirements is decidable.

Furthermore, we show that the validity problem of RF formulas is decidable whereas their satisfiability problem is not recursively enumerable.

Let us start by considering the verification problem. First of all, observe that for every pure-time CDF formula φ, the integration graph recognizing $[\![\varphi]\!]_{rc}$ obtained by the construction of Section 6.1 is actually a timed graph. Moreover, as we prove in the full paper, for every simple RF formula φ, the construction of Section 6.2 yields a simple integration automaton which recognizes the trajectories falsifying φ.

It can also be seen that the product of a timed graph with a simple integration automaton is a simple integration automaton. Thus, the verification problem for pure-time CDF formulas, and more generally for designs described by timed graphs, w.r.t. simple RF formulas, is reducible to the reachability problem for simple integration graphs. It can be deduced from the result given in [4] that this reachability problem is decidable. Then, we obtain the following decidability result.

Theorem 12. *The verification problem for pure-time CDF formulas (more generally for timed graphs) w.r.t. simple RF formulas, is decidable.*

By Proposition 2 and Theorem 12, we obtain the following decidability result.

Corollary 13. *The verification problem for CoD control design formulas w.r.t. CoD requirement formulas is decidable.*

It is worth to note that we cannot use directly the decidability results concerning integration graphs that have been established using digitization techniques as [16, 6]. Indeed, these results concern systems without strict inequalities, and adopt a semantics which allows to observe several states at a same time. These assumptions are not true in the framework of CoD we consider in this paper. Let us now consider the validity and the satisfiability problems of RF formulas. First of all, since a formula is valid iff it is satisfied by all right-continuous trajectories, and since the set of all right-continuous trajectories can be defined by a timed graph, we obtain as a consequence of Theorem 12 the following result.

Corollary 14. *The validity problem of simple RF formulas, and hence of CoD requirement formulas, is decidable.*

Actually, we can prove that the validity problem of all RF formulas is decidable by reduction to a linear programming problem. On the other hand, we can show that the divergence problem (complement of the halting problem) of deterministic 2-counter machines is reducible to the satisfiability of RF formulas. Then, we have the following results.

Theorem 15. *The validity problem of RF formulas is decidable whereas their satisfiability problem is undecidable (Π_1^0-hard).*

8 Conclusion

We have shown that the verification problem of a CoD control design w.r.t. a CoD requirement is decidable. We have obtained this result by showing the link between this problem and an analysis problem in the framework of hybrid automata. In fact, given a CoD control design φ_1 and a CoD requirement φ_2, we can construct systematically an automaton whose language is empty iff every trajectory satisfying φ_1 satisfies also φ_2. As far as we know, this is the first completely mechanizable translation between CoD and hybrid automata. Furthermore, the positive decidability results we obtained are the first ones concerning fragments of dense time CoD which are not purely propositional. Our work has also practical interests since it allows to benefit from the advantages of the CoD based approach as well as of the hybrid automata based approach. Indeed, the CoD framework provides an expressive high-level specification language; on the other hand, the hybrid automata framework offers several algorithmic analysis techniques and verification tools [2].

Acknowledgment We thank Amir Pnueli and Joseph Sifakis for fruitful discussions.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model-Checking for Real-Time Systems. In *5th Symp. on Logic in Computer Science*. IEEE, 1990.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *TCS*, 138, 1995.
3. R. Alur, C. Courcoubetis, T. Henzinger, and P-H. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.
4. R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing Accumulated Delays in Real-time Systems. In *CAV'93*. LNCS 697, 1993.
5. R. Alur and D. Dill. A Theory of Timed Automata. *TCS*, 126, 1994.
6. A. Bouajjani, R. Echahed, and R. Robbana. Verifying Invariance Properties of Timed Systems with Duration Variables. In *FTRTFT'94*. LNCS 863, 1994.
7. A. Bouajjani, R. Echahed, and J. Sifakis. On Model Checking for Real-Time Properties with Durations. In *8th Symp. on Logic in Computer Science (LICS'93)*. IEEE, 1993.
8. A. Bouajjani and R. Robbana. Verifying ω-Regular Properties for Subclasses of Linear Hybrid Systems. Submitted for publication, 1995.
9. K. Cerans. Decidability of Bisimulation Equivalence for Parallel Timer Processes. In *Proc. Computer-Aided Verification (CAV'92)*. LNCS 663, 1992.
10. Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A Calculus of Durations. *Information Processing Letters*, 40:269–276, 1991.
11. Z. Chaochen and X. Li. A Mean -Value Duration Calculus. In *A Classical Mind, Essays in Honour of C.A.R. Hoare*. Prentice-Hall, 1994.
12. Z. Chaouchen, M.R. Hansen, and P. Sestoft. Decidability and Undecidability Results for Duration Calculus. In *STACS'93*. LNCS 665, 1993.
13. J. He, C.A.R. Hoare, M. Fränzle, M. Müller-Olm, E.R. Olderog, M. Schenke, M.R. Hansen, A.P. Ravn, and H. Rishel. Provably Correct Systems. In *FTRTFT'94*. LNCS 863, 1994.
14. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's Decidable about Hybrid Automata. In *ICALP'95??????????*, 1995.
15. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model-Checking for Real-Time Systems. In *LICS'92*. IEEE, 1992.
16. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.
17. O. Maler, Z. Manna, and A. Pnueli. A Formal Approach to Hybrid Systems. In *REX workshop on Real-Time: Theory and Practice*. LNCS 600, 1992.
18. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An Approach to the Description and Analysis of Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.
19. A. Ravn. Design of Embedded Real-time Computing Systems. Unpublished Notes, 1994.
20. A.P. Ravn, H. Rischel, and K.M. Hansen. Specifying and Verifying Requirements of Real-Time Systems. In *Transaction on Software Engeneering*. IEEE, 1993.
21. W. Thomas. Automata on Infinite Objects. In *Handbook of Theo. Comp. Sci.* Elsevier Sci. Pub., 1990.