Specifying Various Time Models with Temporal Propositional Variables in Duration Calculus

Dang Van Hung

The United Nations University
International Institute for Software Technology
P.O.Box 3058, Macau
dvh@iist.unu.edu

Abstract. Many extensions of Duration Calculus (DC) have been proposed for handling different aspects of real-time systems. For each extension several different semantics are defined for different time structures which are suitable for different applications and achieve low complexity for the decidability of some properties. Hence, different proof systems have to be developed for reasoning in different calculi. We demonstrate that with temporal propositional letters, many useful time structures and operators can be completely described in the original DC with continuous time. Hence, we can use the proof system for original DC and the specification of the specific time structure to reason in that time structure without the need of introducing a new calculus.

1 Introduction

Since it was introduced in 1992 by Zhou, Hoare and Ravn [3], Duration Calculus (DC) has attracted a great deal of attentions. Details on DC are presented in Zhou and Hansen's monograph [2]. Many extensions of Duration Calculus have been proposed for handling different aspects of real-time systems [15, 5]. For each extension several different semantics are defined for different time structures which are suitable for different applications and have low complexity for the decidability of some properties [13, 9]. Among the different time structures, the abstract time domain makes the calculus complete when it is abstracted away from dedicated properties of real numbers or natural numbers [6], and the discrete time domain makes the calculus decidable, able to carry out model checking, and suitable for the specification of digital systems [14, 13, 8]. When proposing a new time domain, one has to give semantics and develop a proof system for the calculus within the given semantics. This sometimes makes the calculus inconvenient to use. We found that in most cases, the proof system for the calculus on special domains is the original one with some small modification. Therefore, it would be convenient if we can use the original calculus without any modification for different applications which can still capture special aspects of the application domains. From our earlier works on specification and reasoning about real-time systems with Duration Calculus [10, 11], we found that it is very convenient to use temporal propositional letters with specific meaning to specify

the properties of time intervals. In this paper, we demonstrate that we can use this technique to model different time structures and time models. Then, reasoning in these time structures can be carried out using the original proof system of DC and the specification of the structures in DC with continuous semantics. We show that our technique works well for some case studies.

The paper is organised as follows. In the next section, we give a summary of Duration Calculus. Our main technique for modelling time structures is described in Section 3. Sections 4 demonstrates how our technique can be used for describing data sampling and projection. In Section 5, we show how the technique is used in modeling and reasoning in the development of real-time systems via the well-known case study "Biphase Mark Protocol".

2 Summary of Duration Calculus

We give a brief summary of Duration Calculus in this section. Readers are referred to [2] for more details on DC. The version of Duration Calculus we present here has several additional operators that are useful for the specification purposes.

Time in DC is the set $\mathbb{R}^+$ of the non-negative real numbers. For $t, t' \in \mathbb{R}^+ (t \leq t')$, $[t, t']$ denotes the time interval from t to t'. Let $intv$ denote the set of all time intervals.

2.1 Syntax of Duration Calculus

Assume that $V = \{P, Q, \ldots\}$ is a countable set of state variables, and $T = \{X, R, \ldots\}$ is a set of temporal propositional letters. Let f^n and A^n $(n \geq 0)$ denote n-ary function and n-ary predicate names respectively. The syntax classes *state expressions*, *terms*, and *formulas* will be then defined as follows.

State expressions: The set of state expressions is generated by the grammar

$$S \;\widehat{=}\; 0 \mid 1 \mid P \mid \neg S \mid S \wedge S\,,$$

where P stands for state names in the set V.

Terms: The set of terms is generated by

$$r \;\widehat{=}\; \textstyle\int S \mid \ell \mid f^n(r, \ldots, r)\,,$$

where S stands for state expressions.

Formulas: the set of formulas is generated by the grammar:

$$\begin{aligned} D \;\widehat{=}\; & A^n(r, \ldots, r) \mid X \mid D^\frown D \mid \neg D \mid D \vee D \mid \\ & D^* \mid \lceil P \rceil^0 \mid \mid \lceil P \rceil \mid \lceil\lceil P \rceil \end{aligned}$$

where A stands for atomic formulas, and X for temporal propositional letters.

2.2 Semantics of Duration Calculus

Assume that each n-ary function name f^n is associated with a total function from $\mathbb{R}^n$ to $\mathbb{R}$ which is denoted by f^n also, and each n-ary predicate name A^n is associated with a total function from $\mathbb{R}^n$ to $\{tt, ff\}$ which is also denoted by A^n. In this paper, for simplicity we interpret the functions f as operators on reals, e.g. $+, *$, and the relations A as comparative operators between reals, e.g. $<, \leq, =, >, \geq$.

An interpretation $\mathcal{I}$ is a function $\mathcal{I} \in (V \to (\mathbb{R}^+ \to \{0,1\})) \cup (T \to (intv \to \{0,1\}))$, for which each $\mathcal{I}(P)$, $P \in V$ has at most finitely many discontinuity points in any interval $[a,b]$. We shall use the abbreviation $P_{\mathcal{I}} \ \widehat{=} \ \mathcal{I}(P)$ for $P \in V$ and $X_{\mathcal{I}} \ \widehat{=} \ \mathcal{I}(X)$ for $X \in T$. The semantics of state expression, terms, and formulas in an interpretation $\mathcal{I}$ are then defined as follows.

Semantics of state expressions: The semantics of a state expression P in an interpretation $\mathcal{I}$ is a function $\mathcal{I}_P \in Time \to \{0,1\}$ defined inductively on the structure of state expressions by:

$$\begin{array}{ll} \mathcal{I}_0(t) & \widehat{=} 0, \\ \mathcal{I}_1(t) & \widehat{=} 1, \\ \mathcal{I}_P(t) & \widehat{=} P_{\mathcal{I}}(t), \\ \mathcal{I}_{(\neg S)}(t) & \widehat{=} 1 - \mathcal{I}_S(t), \text{ and} \\ \mathcal{I}_{(S \vee Q)}(t) & \widehat{=} \begin{cases} 0 \text{ if } \mathcal{I}_S(t) = 0 \text{ and } \mathcal{I}_Q(t) = 0, \\ 1 \text{ otherwise.} \end{cases} \end{array}$$

Semantics of terms: The semantics of a term r in an interpretation $\mathcal{I}$ is a function $\mathcal{I}_r \in intv \to \mathbb{R}$ defined inductively on the structure of terms by:

$$\begin{array}{ll} \mathcal{I}_{\int P}([a,b]) & \widehat{=} \int_a^b \mathcal{I}_P(t)dt, \\ \mathcal{I}_{\ell}([a,b]) & \widehat{=} b - a, \text{ and} \\ \mathcal{I}_{f^n(r_1,\ldots,r_n)}([a,b]) & \widehat{=} f^n(\mathcal{I}_{r_1}([a,b]), \ldots, \mathcal{I}_{r_n}([a,b])). \end{array}$$

Semantics of formulas: The semantics of a formula D in an interpretation $\mathcal{I}$ is a function $\mathcal{I}_D \in intv \to \{tt, ff\}$ defined inductively on the structure of formulas as follows. Using the following abbreviations:

$$\begin{array}{l} \mathcal{I}, [a,b] \models D \ \widehat{=} \mathcal{I}_D([a,b]) = tt \text{ and} \\ \mathcal{I}, [a,b] \not\models D \ \widehat{=} \mathcal{I}_D([a,b]) = ff, \end{array}$$

$\mathcal{I}_D$ is defined by:

$$\begin{array}{ll} \mathcal{I}, [a,b] \models A^n(r_1, \ldots, r_n) & \text{iff } A^n(\mathcal{I}_{r_1}([a,b]), \ldots, \mathcal{I}_{r_n}([a,b])) = tt, \\ \mathcal{I}, [a,b] \models X & \text{iff } X_{\mathcal{I}} = 1, \\ \mathcal{I}, [a,b] \models (\neg D) & \text{iff } \mathcal{I}, [a,b] \not\models D \\ \mathcal{I}, [a,b] \models (D_1 \vee D_2) & \text{iff } \mathcal{I}, [a,b] \models D_1 \text{ or } \mathcal{I}, [a,b] \models D_2 \\ \mathcal{I}, [a,b] \models (D_1 ^\frown D_2) & \text{iff } \mathcal{I}, [a,m] \models D_1 \text{ and } \mathcal{I}, [m,b] \models D_2 \text{ for some } m \in [a,b] \\ \mathcal{I}, [a,b] \models D^* & \text{iff either } a = b \text{ or } \mathcal{I}, [m_i, m_{i+1}] \models D \text{ for some } n \in \mathcal{N} \text{ and} \\ & a = m_0 < m_1 < \ldots < m_n = b \\ \mathcal{I}, [a,b] \models \lceil P \rceil^0 & \text{iff } a = b \text{ and } P_{\mathcal{I}}(a) = 1 \\ \mathcal{I}, [a,b] \models \lceil P \rceil & \text{iff } a < b \text{ and } P_{\mathcal{I}}(t) = 1 \text{ for all } a < t < b \\ \mathcal{I}, [a,b] \models \lceil\lceil P \rceil & \text{iff } a < b \text{ and } P_{\mathcal{I}}(t) = 1 \text{ for all } a \leq t < b \end{array}$$

A relatively complete proof system for Duration Calculus with no operator * and ^{0} has been given in [2], and the complete proof system for Duration Calculus with Iteration on the abstract time domain is given in [6].

3 Specifying Substructure of Time with Temporal Propositional Letters

In this section we consider how Duration Calculus can specify some classes of time models with temporal propositional letters. We show that different classes of Duration Calculus time models can be expressed by sub-languages of Duration Calculus. Hence, there is no need to have different definitions of Duration Calculus for different classes of time models. Using the original DC with the sub-language of DC for a class of models as the assumption we can reason about the validity in the model class.

3.1 Discrete Duration Calculus Models

Discrete models of Duration Calculus use the set of natural numbers $\mathbb{N}$, which is a subset of $\mathbb{R}^+$, for time (we assume that $0 \in \mathbb{N}$). We can embed the discrete time models into continuous time models by considering a state variable in discrete DC models as a state in continuous models that can change its value only at the integer points. For that purpose, we introduce several fresh temporal propositional letters and state variables with specific meaning. Let int be a temporal propositional letter with the meaning that int is interpreted as 1 for an interval if and only if the interval is from an integer to an integer, i.e. for any interpretation $\mathcal{I}$, $int_{\mathcal{I}}([a,b]) = 1$ iff $a, b \in \mathbb{N}$. Let C be a state variable that changes its value at each natural number which represents a tick of the real-time clock, i.e. $C_{\mathcal{I}}(t) = 1$ iff $\lfloor t \rfloor$ is odd. The axioms to characterise the properties of the temporal propositional letter int can be given as follows. First, the integer intervals have integral endpoints, and remain integer intervals when extended by 1 time unit:

$$\begin{aligned} int \Rightarrow ((int \wedge \ell = 0)^\frown(int \wedge \ell = 1)^*) \wedge \\ ((int \wedge \ell = 1)^*{}^\frown(int \wedge \ell = 0)) \end{aligned} \tag{1}$$

$$int^\frown(\ell = 1) \Rightarrow int \tag{2}$$

Second, $int \wedge \ell = 1$ should be a unique partition of the greatest integral subinterval of any interval with length 2 or more, i.e.

$$\begin{aligned} \ell \geq 2 \Rightarrow \ell < 1^\frown((int \wedge \ell = 1)^* \wedge \\ \neg(true^\frown(int \wedge \ell = 1)^\frown\neg(int \wedge \ell = 1)^*) \wedge \\ \neg(\neg(int \wedge \ell = 1)^*{}^\frown(int \wedge \ell = 1)^\frown true))^\frown \\ \ell < 1 \end{aligned} \tag{3}$$

Similarly to Lemma 3.2 in [4] we can show that the axiom 3 is equivalent to the fact that any interval $[b, e]$ that have the length 2 or longer has the unique set of time points $b \leq \tau_0 < \tau_1 < \ldots < \tau_m \leq e$ such that $\mathcal{I}, [\tau_i, \tau_{i+1}] \models int \wedge \ell = 1$, $\tau_0 - b < 1$ and $e - \tau_m < 1$, and $[\tau_i, \tau_{i+1}]$ are the only subintervals of $[b, e]$ that that satisfy $(int \wedge \ell = 1)$.

Let $\mathcal{ID}$ denote the set of these three axioms 1, 2 and 3. $\mathcal{ID}$ specifies all the properties of integer intervals except that their endpoints are integer.

Proposition 1.

1. *Let $\mathcal{I}$ be an interpretation satisfying that $int_{\mathcal{I}}([b, e]) = true$ iff $[b, e]$ is an integer interval. Then $\mathcal{I}, [b, e] \models D$ for any integer interval $[b.e]$, and for any formula $D \in \mathcal{ID}$.*
2. *Let $\mathcal{I}$ be an interpretation satisfying that $\mathcal{I}, [b, e] \models D$ for any interval $[b.e]$, and for any formula $D \in \mathcal{ID}$. Then, $int_{\mathcal{I}}([0, 0]) = true$ implies that for $int_{\mathcal{I}}([b, e]) = true$ iff $[b, e]$ is an integer interval.*

Proof. The item 1 is obvious, and we only give a proof of Item 2 here. Let us consider an interval $[0, n]$ with $n > 100$. From the fact that $\mathcal{I}, [0, n] \models D$ where D is the formula 3, we have that there are points $0 \leq \tau_0 < \tau_1 < \ldots < \tau_m \leq e$ such that $\mathcal{I}, [\tau_i, \tau_{i+i}] \models int \wedge \ell = 1$, $\tau_0 < 1$ and $n - \tau_m < 1$, and

$$\begin{aligned}\mathcal{I}, [\tau_0, \tau_m] \models (\neg(true^\frown(int \wedge \ell = 1)^\frown\neg(int \wedge \ell = 1)^*)\wedge \\ \neg(\neg(int \wedge \ell = 1)^*{}^\frown(int \wedge \ell = 1)^\frown true))\end{aligned}$$

If $\tau_0 > 0$, from the axiom 2, it follows that $\mathcal{I}, [0, k] \models int$ for all $k \in \mathbb{N}$ and $k \leq n$ and $k < \tau_k < k + 1$. Applying the axiom 1 for the interval $[0, k]$ implies that $\mathcal{I}, [k, k + 1] \models int \wedge \ell = 1$. Consequently, $\mathcal{I}, [m - 1, \tau_m] \models \neg(int \wedge \ell = 1)^*$ and $\mathcal{I}, [m - 2, m_1] \models (int \wedge \ell = 1)$. This is a contradiction to $\mathcal{I}, [\tau_0, \tau_m] \models \neg(true^\frown(int \wedge \ell = 1)^\frown\neg(int \wedge \ell = 1)^*)$. □

Note that Item 2 of Proposition 1 can be generalised as

> Let $\mathcal{I}$ be an interpretation satisfying that $\mathcal{I}, [b, e] \models D$ for any interval $[b.e]$, and for any formula $D \in \mathcal{ID}$. Let $h \in \mathcal{R}^+$, $h < 1$. Then, $int_{\mathcal{I}}([h, h]) = true$ implies that for $int_{\mathcal{I}}([b, e]) = true$ iff $[b, e]$ is of the form $[h + n, h + m]$, $m, n \in \mathbb{N}$ and $n \leq m$.

So, $\mathcal{ID}$ is a set of formulas specifying the set of intervals of a discrete time obtained by shifting $\mathbb{N}$ by h time units $(h < 1)$.

The state variable C can also express if an interval is an integer interval. Namely, we have

$$\begin{aligned}&(\lceil C\rceil \vee \lceil\neg C\rceil) \wedge \ell = 1 \Rightarrow int \\ &int \wedge \ell = 1 \Rightarrow (\lceil C\rceil \vee \lceil\neg C\rceil) \\ &\lceil C\rceil^\frown\lceil\neg C\rceil \Rightarrow true^\frown int^\frown true \\ &\lceil\neg C\rceil^\frown\lceil C\rceil \Rightarrow true^\frown int^\frown true \\ &(int \wedge \ell = 1)^\frown(int \wedge \ell = 1) \Rightarrow \\ &\quad((\lceil C\rceil \wedge \ell = 1)^\frown(\lceil\neg C\rceil \wedge \ell = 1))\vee \\ &\quad((\lceil\neg C\rceil \wedge \ell = 1)^\frown(\lceil C\rceil \wedge \ell = 1))\end{aligned}$$

Let $\mathcal{CC}$ denote the set of these formulas. $\mathcal{CC}$ specifies all the properties of the special clock state variable C. Any interval satisfying $int \wedge \ell > 0$ can be expressed precisely via a DC formula with state variable C (without int). Perhaps $int \wedge \ell = 0$ is the only formula that cannot be expressed by a formula via state variable C without int. $\mathcal{CC}$ can also be used as a means to define the variable C via int and vice-versa. If we use $\mathcal{CC}$ to define int, the axioms for C simply are:

$$\lceil C \rceil \vee \lceil \neg C \rceil \Rightarrow \ell \leq 1 \tag{4}$$

$$((\lceil C \rceil ^\frown \lceil \neg C \rceil ^\frown \lceil C \rceil) \vee (\lceil \neg C \rceil ^\frown \lceil C \rceil ^\frown \lceil \neg C \rceil)) \Rightarrow \ell \geq 1 \tag{5}$$

The relationship between these axioms and the axioms for int presented earlier is formulated as:

Proposition 2. *Let interpretation $\mathcal{I}$ be such that the formulas in $\mathcal{CC}$ and axioms (1) and (2) are satisfied by all intervals.*

1. *If the axioms (4) and (5) are satisfied by all intervals, the axiom (3) is satisfied by all intervals.*
2. *If the axiom (3) is satisfied by all intervals then the axioms (4) and (5) are satisfied by all intervals, too.*

Proof.

Proof of Item 1. The axioms (4) and (5) implies that the formula

$$(\lceil \neg P \rceil ^\frown \lceil P \rceil ^\frown \lceil \neg P \rceil) \Rightarrow (\lceil \neg P \rceil ^\frown (\lceil P \rceil \wedge \ell = 1) ^\frown \lceil \neg P \rceil))$$

is satisfied for any interval when P is either C or $\neg C$. For any interval $[b, e]$, if $e - b \geq 2$ then there are $b = \tau_0 < \ldots < \tau_n = e$ such that $[\tau_i, \tau_{i+1}]$ satisfies $\lceil C \rceil \vee \lceil \neg C \rceil$, and τ_i, $0 < i < n$ are the points the state C changes its value. Therefore, from (3), (4) and $\mathcal{CC}$ the formula $int \wedge \ell = 1$ is satisfied by $[\tau_i, \tau_{i+1}]$ when $0 < i < n - 1$, and $\tau_1 - \tau_0 < 1$ and $\tau_n - \tau_{n-1} < 1$. Furthermore, from $int \wedge \ell = 1 \Rightarrow (\lceil C \rceil \vee \lceil \neg C \rceil)$ it follows that $[\tau_i, \tau_{i+1}]$, $0 < i < n - 1$ are the only intervals satisfying $int \wedge \ell = 1$. Hence, (3) is satisfied by $[b, e]$.

Proof of Item 2. Let $h > 0$ be the first time point that state C changes its value. From the axioms (1), (2) and (3) it follows that $h \leq 1$ and $int \wedge \ell = 1$ is satisfied by and only by the intervals of the form $[n + h, n + 1 + h]$, $n \in \mathbb{N}$. Hence, if $\mathcal{CC}$ is satisfied by all intervals, the axioms (4) and (5) are also satisfied by all intervals. □

So, with the assumption that 0 is an integer point, the axioms (4) and (5) are equivalent to the axiom (3).

Let *step* be a temporal propositional letter that represents two consecutive state changes of the system under consideration. When there are several state changes at a time point t, *step* evaluates to 1 over interval $[t, t]$. When two consecutive state changes are at t and t' such that $t \neq t'$, *step* is true for the

interval $[t, t']$, and for any state variable P, either $\lceil P \rceil$ or $\lceil \neg P \rceil$ holds for the interval $[t, t']$. This is represented by:

$$step \wedge \ell > 0 \Rightarrow (\lceil P \rceil \vee \lceil \neg P \rceil) \text{ for any state variable } P$$
$$step \wedge \ell > 0 \Rightarrow \neg((step \wedge \ell > 0)^\frown(step \wedge \ell > 0))$$

Let $\mathcal{SC}$ denote this class of formulas.

Now consider two kinds of Duration Calculus semantics which are different from the original one defined earlier for continuous time, and called discrete semantics and discrete step time semantics.

Discrete Duration Calculus semantics are defined in the same way as for continuous time semantics except that all intervals are integer intervals. So, a, b, m and m_i in the definition should be integers instead of reals, and an interpretation $\mathcal{I}$ should assign to each state variable P a function from $\mathbb{N}$ to $\{0, 1\}$, and then expanded to a function from $\mathcal{R}^+$ to $\{0, 1\}$ by letting $\mathcal{I}_P(t) = \mathcal{I}_P(\lfloor t \rfloor)$ which is right continuous, and could be discontinuous only at integer time points. Let us use $\models_{DDC}$ to denote the modelling relation in these semantics.

Similarly, discrete step time Duration Calculus semantics are defined by restricting the set of intervals to that of intervals between state change time points. So, a, b, m and m_i in the definition should be time points where states change, and an interpretation $\mathcal{I}$ should assign to each state variable P a function from $\mathcal{S}$ to $\{0, 1\}$, where $\mathcal{S}$ is a countable subset of $\mathcal{R}^+$ intended to be the set of time points for state changes that includes the set $\mathbb{N}$. $\mathcal{I}_P$ is then expanded to a function from $\mathcal{R}^+$ to $\{0, 1\}$ by letting $\mathcal{I}_P(t) = \mathcal{I}_P(t_s)$, where $t \in \mathcal{R}^+$ and $t_s = \max\{t' \in \mathcal{S} \mid t' \leq t\}$. Then $\mathcal{I}_P(t)$ is also right continuous, and could be discontinuous only at a point in $\mathcal{S}$. Let us use $\models_{SDC}$ to denote the modelling relation in this semantics.

To express that states are interpreted as right continuous functions, we can use formula called $\mathcal{RC}$

$$\lceil P \rceil \Rightarrow \lceil \lceil P \rceil \text{ for any state variable } P$$

In [14], Paritosh also proposed a semantics using only the intervals of the form $[0, t]$. We can also specify this interval model with a temporal propositional letter Pre. Pre is interpreted as true only for the interval of the form $[0, t]$. Pre is specified by the set of formulas $Pref$ defined as

$$Pre^\frown true \Rightarrow Pre$$
$$\neg(\ell > 0^\frown Pre)$$
$$Pre \wedge D \Rightarrow (Pre \wedge \ell = 0)^\frown D$$
$$Pre \wedge (D1^\frown D2) \Rightarrow (Pre \wedge D1)^\frown D2$$

Proposition 3. *Let $\mathcal{I}$ be an interpretation that validates the set of formulas $Pref$ and $\mathcal{I}, [0, 0] \models Pre$. Then, $\mathcal{I}, \mathcal{V}, [a, b] \models Pre$ iff $a = 0$.*

Proof. Straightforward □

Then, a formula D is valid in the prefix time interval model if and only if $Pre \Rightarrow D$ is a valid formula in the original model of time interval.

So far, we have introduced special temporal propositional letters *int*, *step* and *Pre* together with DC formulas specifying their special features. We are going to show that with these propositional letters we can provide a complete description of many useful time models.

Integer Time Model To specify that a state can only change at an integer time point, we can use the formula $\mathcal{IS}$:

$$step \Rightarrow int$$

Let $\mathcal{DL}$ be the union of $\mathcal{SC}$, $\mathcal{IS}$, $\mathcal{ID}$, $\mathcal{RC}$. $\mathcal{DL}$ forms a relative complete specification for the discrete time structure. Let φ be a formula which does not have any occurrence of temporal variables *int* ans *step*. Let $intemb(\varphi)$ be a formula that obtained from φ by replacing each proper subformula ψ of φ by $\psi \wedge int$. For example $intemb(\phi^\frown\neg\psi) = (\phi \wedge int)^\frown(int \wedge \neg(\psi \wedge int)$.

Theorem 1. *Let φ be a DC formula with no occurrence of temporal proposition letters. Then, $\mathcal{DL} \vdash int \Rightarrow intemb(\varphi)$ exactly when $\models_{DDC} \varphi$.*

Proof. Any discrete time model $\mathcal{I}, [a, b]$ can be extended to a model that satisfies the formulas in $\mathcal{DL}$ in the obvious way, namely with the interpretation for *int* and *step* with the intended meanings for them. By induction on the structure of the formula φ, it is easy to prove that $\mathcal{I}, [a, b] \models_{DDC} \varphi$ if and only $\mathcal{I}, [a, b] \models intemb(\varphi)$.

Then, the "only if" part follows directly from the soundness of the proof of the DC system that $intemb(\varphi)$ is satisfied by any integer model that satisfies $\mathcal{DL}$.

The "if" part is proved as follows. From the above observation, if $\models_{DDC} \varphi$ then $int \Rightarrow intemb(\varphi)$ is a valid formula in DC with the assumption $\mathcal{DL}$. Consequently, from the relative completeness of DC, $intemb(\varphi)$ is provable in DC with the assumption $\mathcal{DL}$. □

Discrete Step Time Model As it was said earlier, a discrete step time model consists of all time points at which there is a the state change. Since we have assumed that the special state variable C for the clock ticks is present in our system that changes its value at every integer point, this model of time should also include the set of natural numbers. This is the reason that we include $\mathbb{N}$ as a subset of $\mathcal{S}$. This time model was defined and used by Pandya et al in [14]. To represent a time point in this model, we introduce a temporal propositional letter *pt*, *pt* holds for an interval $[t, t']$ iff $t = t'$ and t is a time point at which there is a state change. *pt* should satisfy:

$$\begin{aligned} &pt \Rightarrow \ell = 0 \\ &step \Rightarrow pt^\frown true^\frown pt \\ &int \Rightarrow pt^\frown true^\frown pt \\ &int \Rightarrow pt^\frown step^* \end{aligned}$$

Let $\mathcal{DP}$ denote this set of formulas. The last formula in this set expresses our assumption that no Zeno computation is allowed, i.e. in any time interval, there are only a finite number of state changes. Let us define a DC formula dis as

$$dis \mathrel{\widehat{=}} (pt^\frown true^\frown pt)$$

dis represents an interval between two discrete points. When considering the Discrete Step Time Models, the chop point should satisfy pt.

The sublanguage $\mathcal{DSL}$, which is the union of $\mathcal{SC}$, $\mathcal{ID}$, $\mathcal{CC}$, $\mathcal{DP}$ $\mathcal{DC}$ and $\mathcal{RC}$, forms a relatively complete specification for the discrete time structure.

Let $disemb(\varphi)$ be a formula that is obtained from φ by replacing each proper subformula ψ of φ by $\psi \wedge dis$. For example $disemb(\phi^\frown\neg\psi) = (\phi \wedge dis)^\frown(dis \wedge \neg(\psi \wedge dis)$.

Theorem 2. *Let φ be a DC formula with no occurrence of temporal proposition letters. Then, $\mathcal{DSL} \vdash dis \Rightarrow disemb(\varphi)$ exactly $\models_{SDC} \varphi$.*

Proof. The proof works in exactly the same way as the proof of Theorem 1.

Any discrete step time model $\mathcal{I}, [a, b]$ can be extended to a model that satisfies formulas in $\mathcal{DL}$ in the obvious way, namely with the interpretation for int and $step$ with the intended meanings for them. By induction on the structure of the formula φ, it is easy to prove that $\mathcal{I}, [a, b] \models_{SDC} \varphi$ if and only $\mathcal{I}, [a, b] \models intemb(\varphi)$.

Then, the "only if" part follows directly from the soundness of the proof of the DC system that $intemb(\varphi)$ is satisfied by any discrete step time model that satisfies $\mathcal{DL}$.

For the "if" part, notice that if $\models_{SDC} \varphi$ then $dis \Rightarrow intemb(\varphi)$ is a valid formula in DC with the assumption $\mathcal{DL}$. Consequently, from the relative completeness of DC, $disemb(\varphi)$ is provable in DC with the assumption $\mathcal{DL}$. □

Sampling Time Models A sampling time model consists of the time points where we sample the data. Assume that the samplings are frequent enough and that any state change should be at a sampling point. To specify this time model, we can use $\mathcal{DSL}$ and an additional assumption

$$step \Rightarrow \ell = 1/h$$

where $h \in \mathbb{N}$, $h > 0$, i.e. $1/h$ is the sampling time step. Let $\mathcal{SL}_h$ be the language for the sampling time model with the sampling time step $1/h$.

4 Specifying Sampling, Periodic Task Systems and Projection to Discrete Time

4.1 Sampling

Sampling and specifying periodic task systems are immediate applications of the results presented in the previous section.

We have built a language for sampling time models based on the continuous time DC. Hence, we can use the proof system of DC to reason about validity of a formula in that time and state model. How to relate the validity of a formula D in that time and state model with the validity of a formula D' in the original DC? In our early work [9], we have considered that relation, but had to formulate the results in a natural meta language due to the use of different semantic models. With the help from the time modeling language, we can also formulate the relationship as formulas in DC.

Let P be a state variable. Let P_h be a state in the sampling time model with the sampling time step $1/h$ such that P_h is interpreted the same as P at any sampling time point, i.e. $\Box(pt \Rightarrow (\lceil P \rceil^0 \Leftrightarrow \lceil P_h \rceil^0)$ (denoted by $samp(P, P_h)$), and $\Box(step \wedge \ell > 0 \Rightarrow (\lceil P_h \rceil \vee \lceil \neg P_h \rceil))$ (denoted by $dig(P_h)$). Let $stable(P, d)$ denote the formula $\Box((\lceil \neg P \rceil^\frown \lceil P \rceil^\frown \lceil \neg P \rceil) \Rightarrow \ell \geq d)$.

Theorem 3. *Let $d > 1/h$. The following formulas are valid in DC:*

1. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $(\int P = m \Rightarrow |\int P_h - m| \leq \min\{\ell, (\ell/d + 1)1/h\}$
2. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $(\int P = m \wedge dis) \Rightarrow |\int P_h - m| \leq \min\{\ell, 1/h\ell/d\}$
3. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $\int P_h = m \Rightarrow |\int P - m| \leq (\ell/d + 1)1/h$
4. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $\int P_h < m \Rightarrow \int P < m + 1/h(\ell/d + 1)$
5. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $\int P < m \Rightarrow \int P_h < m + 1/h(\ell/d + 1)$
6. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $\int P_h > m \Rightarrow \int P > m - 1/h(\ell/d + 1)$
7. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $\int P > m \Rightarrow \int P_h > m - 1/h(\ell/d + 1)$
8. $(stable(P, d) \wedge samp(P, P_h) \wedge dig(P_h)) \Rightarrow$
   $dis \Rightarrow (\lceil P_h \rceil \Leftrightarrow \lceil P \rceil)$

Proof. This is just a reformulation of Theorem 1 in [9]. □

This theorem is useful for deriving a valid formula in the original DC from valid formulas in discrete time model. It can be used in approximate reasoning, especially in model checking: to check if a system S satisfies a DC property D, we can check a sampling system S_h of S whether it satisfies a discrete DC property D_h. D_h is found such that $S_h \models D_h$ implies $S \models D$. This technique has been used in [14].

4.2 Periodic Task System

A periodic task system T consists of n processes $\{1, \ldots, n\}$. Each process i raises its request periodically with period T_i, and for each period it requests a constant amount of processor time C_i. A specification of system T in DC has been

given in many works, see e.g [2], which assume that all the processes raise their request at time 0. We can give a complete specification of the system without this assumption using the same technique that was introduced for temporal variable *int* in the previous section. To specify periodic behaviour of process i, we also use temporal variable $dLine\_i$ as in [2] whose behavior is similar to temporal variable *int*, and specified by:

$$dLine\_i \Rightarrow ((dLine\_i \wedge \ell = 0)^\frown(dLine\_i \wedge \ell = T_i)^*) \wedge ((dLine\_i \wedge \ell = T_i)^*{}^\frown(dLine\_i \wedge \ell = 0)) \quad (6)$$

$$dLine\_i^\frown(\ell = T_i) \Rightarrow dLine_i \quad (7)$$

$$\ell \geq 2T_i \Rightarrow \ell < T_i{}^\frown((dLine\_i \wedge \ell = T_i)^* \wedge \neg(true^\frown(dLine\_i \wedge \ell = T_i)^\frown\neg(dLine\_i \wedge \ell = T_i)^*) \wedge \neg(\neg(dLine\_i \wedge \ell = T_i)^*{}^\frown(dLine\_i \wedge \ell = T_i)^\frown true))^\frown \ell < T_i \quad (8)$$

Let Run_i be a state variable saying that process i is running on the processor, i.e. $Run_i(t) = 1$ if process i is running on the processor, and $Run_i(t) = 0$ otherwise. Let $Stand_i$ be a state variable saying that the current request of process i has not been fulfilled. The behaviour of process i is fully specified by:

$$dLine\_i \wedge \ell = T_i \Rightarrow (((\int Run_i < C_i \Leftrightarrow \lceil Stand_i \rceil)^\frown true) \wedge (\int Run_i = C_i{}^\frown \ell > 0 \Rightarrow \int Run_i = C_i{}^\frown\lceil\neg Stand_i\rceil))$$

The requirement of system T is simply specified by: for all $i \leq n$,

$$dLine\_i \wedge \ell = T_i \Rightarrow \int Run_i = C_i$$

Formulas 6, 7 and 8 form a complete specification of temporal propositional variables $dLine_i$, $i \leq n$, and are useful in proving the correctness of a scheduler for system T. A priority-based scheduler $\mathcal{S}$ for system T with single processor is characterised by state variables $HiPri_{ij}$ $(i, j \leq n, i \neq j)$ which specify the dynamic priority among the processes defined by $\mathcal{S}$, and the following state formulas characterising its behaviour:

$$\wedge_{i\neq j}((Run_i \wedge Stand_j) => HiPri_{ij})$$
$$\wedge_{i\leq n}(Run_i => Stand_i)$$
$$\wedge_{i\neq j}(HiPri_{ij} \Rightarrow \neg HiPri_{ji})$$
$$\wedge_{i\neq j}\neg(Run_i \wedge Run_j)$$
$$\vee_{i\leq n} Stand_i \Rightarrow \vee_{i\leq n} Run_i$$

A deadline driven scheduler is a priority-based scheduler that considers process i to have a higher prioity than process j (i.e. the value of $HiPri_{ij}$ at the current time point is 1) iff the deadline for process i is nearer than the deadline for process j. The deadline driven scheduler can be modelled in a much more convenient way than in [2] with the additional formula specifying the behaviour of state variables $HiPri_{ij}$ $(i, j \leq n)$:

$$\wedge_{i\neq j}\lceil HiPri_{ij}\rceil^\frown\ell = T_i \Rightarrow (\neg\Diamond dLine_j)^\frown dLine_i^\frown true$$

The interesting thing here is that variables $HiPri_{ij}$ can be defined in DC, without any quantification on rigid variables, via temporal propositional variables $dLine_i$ $(i \leq n)$ which are completely specified by formulas 6, 7 and 8. With defining $HiPri_{ij}$ in this way, we don't have to assume that all the processes raise their request at time 0. Hence, reasoning about the correctness of the scheduler for the general case can be done with the proof system of DC. We believe that the general model for task scheduling presented in [1] can be clearer and simplified a lot using this technique.

4.3 Hybrid States and Projection to Discrete Time

In [12], He Jifeng introduced the projection from continuous time to discrete time to reason about hybrid systems where we have continuous state and temporal variables. In that paper, intervals are either discrete or continuous. Discrete intervals are embedded in continuous intervals. He introduced an operator for projection $\backslash\backslash$ defined as: let F and G be a DC formula, then $F\backslash\backslash G$ is also a formula with the following semantics: $\mathcal{I}, [a, b] \models F\backslash\backslash G$ iff for for some $n \in \mathbb{N}$ and $a = m_1 \leq m_2 \leq \ldots \leq m_n = b\, \mathcal{I}, [m, m_{i+1}] \models F$ and $\mathcal{I}, < m_1, m_2, \ldots, m_n > \models G$, where $< m_1, m_2, \ldots, m_n >$ is a discrete interval. In [4], Guelev also showed that $F\backslash\backslash G$ can be expressed by a formula in the original DC with a temporal propositional letter. In the framework of this paper, $F\backslash\backslash G$ is described in DC as follows. We assume two kinds of states in the system: continuous states and discrete ones. Instead of forcing $\mathcal{SC}$ to be satisfied by any state, we enforce it to be satisfied by discrete states only. So, $F\backslash\backslash G$ is expressed by

$$((F \wedge step)^* \wedge disemb(G[P_s/P]))$$

where $G[P_s/P]$ is obtained from G by the substitution of discrete state P_s for state P such that $samp(P, P_s)$ for each state P occurred in G. Formula $((F \wedge step)^* \wedge disemb(G[P_s/P]))$ says that the reference interval is discrete interval that satisfies formula $disemb(G[P_s/P])$, and each discrete step satisfies continuous formula F. So, with using temporal propositional letters *step*, the projection $\backslash\backslash$ can be defined and reasoning in the original DC as well.

5 Modelling Communication Protocols with Digitizing in DC

In this section, we show that with discrete time structure formalised, we can model communication protocols using Duration Calculus (DC) in a very convenient way without any extension for digitising. This model has been presented in our earlier work [10, 7]. Consider a model for communication at the physical layer (see Fig. 1). A sender and a receiver are connected via a bus. Their clocks are running at different rates. We refer to the clock of the receiver as the time reference. The receiver receives signals by digitising. Since the signals sent by the sender and the signals received by the receiver are functions from the set $\mathbb{R}^+$

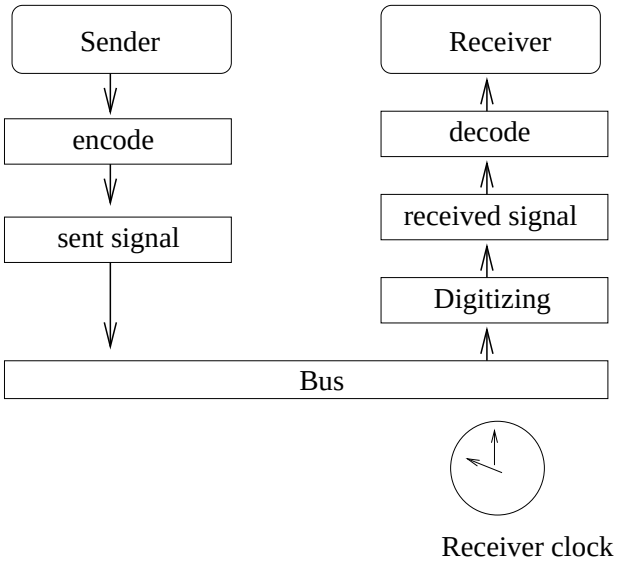

Fig. 1. Communication Protocol Model

to $\{0, 1\}$ (1 represents that the signal is *high*, and 0 represents that the signal is *low*), we can model them as state variables in DC.

The communication protocols are modelled in DC as follows. The signal sent by the sender is modelled by a state X. The signal received by the receiver by sampling the signal on the bus is modelled by a state Y in the sampling time model with the sampling time step 1. So, $step \Leftrightarrow int \wedge \ell = 1$. However, it is not the case that $samp(X, Y)$ due to the fact that it takes a significant amount of time to change the signal on the bus from high to low or vice-versa, and hence, the signal on the bus cannot be represented by a Boolean function. Without loss of generality, assume that the delay between the sender and the receiver is 0. Assume also that when the signal on the bus is neither high nor low, the receiver will choose an arbitrary value from $\{0, 1\}$ for the value of Y. The phenomenon is depicted in Fig. 2. Assume that it takes r (r is a natural number) receiver-clock cycles for the sender to change the signal on the bus from high to low or vice-versa. Then if the sender changes the signal from low to high or from high to low, the receiver's signal will be unreliable for r cycles starting from the last tick of the receiver clock and during this period it can be any value chosen nondeterministically from 0 and 1. Otherwise, the signal received by the receiver is the same as the signal sent by the sender (see Figure 2). This relationship between X and Y is formalised as

$$(\lceil X \rceil \wedge (\ell \geq r + 1)) \Rightarrow (\ell \leq r)^\frown(\lceil Y \rceil \wedge int)^\frown(\ell < 1)\,,$$
$$(\lceil \neg X \rceil \wedge (\ell \geq r + 1)) \Rightarrow (\ell \leq r)^\frown(\lceil \neg Y \rceil \wedge int)^\frown(\ell < 1)\,.$$

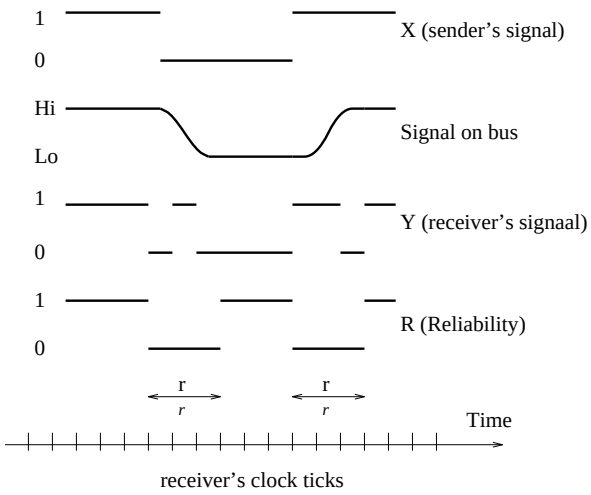

Fig. 2. Signal patterns

Since the behaviour of a state can be specified by a DC formula, a communication protocol can be modelled as consisting of a coding function f, which maps a sequence of bits to a DC formula expressing the behaviour of X, and a decoding function g, which maps a DC formula expressing the behaviour of Y to a sequence of bits. The protocol is correct iff for any sequence w of bits, if the sender puts the signal represented by $f(w)$ on the bus then by digitising the receiver must receive and receives only the signals represented by a DC formula D for which $g(D) = w$.

5.1 Biphase Mark Protocols

In the Biphase Mark Protocols (BMP) the sender encodes a bit as a cell consisting of a mark subcell of length b and a code subcell of length a. The sender keeps the signal stable in each subcell (hence either $\lceil X \rceil$ or $\lceil \neg X \rceil$ holds for the interval representing a subcell). For a cell, if the signal in the mark subcell is the same as the signal in the code subcell, the information carried by the cell is 0; otherwise, the information carried by the cell is 1. There is a phase reverse between two consecutive cells. This means that, for a cell, the signal of the mark subcell of the following cell is held as the negation of the signal of the code subcell of the cell. The receiver, on detecting a state change (of Y), knows that it is the beginning of a cell, and skips d cycles (called the *sampling distance*) and samples the signal. If the sampled signal is the same as the signal at the beginning of the cell, it decodes the cell as 0; otherwise it decodes the cell as 1.

At the beginning of the transmission, the signal is low for a cycles (this means, $\lceil \neg X \rceil$ holds for the interval of length a starting from the beginning). When the sender finishes sending, it keeps the signal stable for cc time units which is longer

than the code subcell. We use HLS, LHS to denote the formulas representing intervals consisting of the code subcell of a cell and the mark subcell of the next one for the sender, and use $HLR^\frown(\ell = d)$, $LHR^\frown(\ell = d)$ to denote the formulas representing the intervals between the two consecutive sampling points (from the time the receiver samples the signal of a code subcell to the next one. Formally,

$$\begin{aligned}
HLS &\mathrel{\widehat{=}} (\lceil X \rceil \wedge \ell = a)^\frown(\lceil \neg X \rceil \wedge \ell = b)\,,\\
LHS &\mathrel{\widehat{=}} (\lceil \neg X \rceil \wedge \ell = a)^\frown(\lceil X \rceil \wedge \ell = b)\,,\\
HLR &\mathrel{\widehat{=}} (\lceil Y \rceil \wedge int \wedge 1 \leq \ell \leq \rho)^\frown(\lceil \neg Y \rceil \wedge \ell = 1)\,,\\
LHR &\mathrel{\widehat{=}} (\lceil \neg Y \rceil \wedge int \wedge 1 \leq \ell \leq \rho)^\frown(\lceil Y \rceil \wedge \ell = 1)\,.
\end{aligned}$$

Now, we are ready to formalise the BMP in DC. What we have to do is write down the encoding function f and the decoding function g. From the informal description of the protocol, we can define f inductively as follows.

1. $f(\epsilon) \mathrel{\widehat{=}} (\lceil \neg X \rceil \wedge \ell = c)$
2. If $f(w) = D^\frown(\lceil X \rceil \wedge \ell = c)$, then

$$\begin{aligned}
f(w0) &\mathrel{\widehat{=}} D^\frown HLS^\frown(\lceil \neg X \rceil \wedge \ell = c)\\
f(w1) &\mathrel{\widehat{=}} D^\frown HLS^\frown(\lceil X \rceil \wedge \ell = c)
\end{aligned}$$

3. If $f(w) = D^\frown(\lceil \neg X \rceil \wedge \ell = c)$, then

$$\begin{aligned}
f(w0) &\mathrel{\widehat{=}} D^\frown LHS^\frown(\lceil X \rceil \wedge \ell = c)\\
f(w1) &\mathrel{\widehat{=}} D^\frown LHS^\frown(\lceil \neg X \rceil \wedge \ell = c)
\end{aligned}$$

For example, $f(1) = LHS^\frown(\lceil \neg X \rceil \wedge \ell = c)$, $f(10) = LHS^\frown LHS^\frown(\lceil X \rceil \wedge \ell = c)$, and $f(101) = LHS^\frown LHS^\frown HLS^\frown(\lceil X \rceil \wedge \ell = c)$.

Because the decoding function g is a partial function, we have to describe its domain first, i.e. what kind of DC formulas on the state Y are detected (received) by the receiver. According to the behaviour of the receiver, first it skips r cycles. Then it begins to scan for an edge (HLR or LHR). When an edge is detected, it skips d cycles and repeats this procedure until it detects that the transmission has completed (Y is stable for more than ρ cycles). Thus, a DC formula D is received by the receiver iff D is of the form $A_0{}^\frown A_1{}^\frown \ldots {}^\frown A_n$, $n \geq 1$, where

- $A_0 = (1 \geq \ell \wedge \ell > 0)^\frown(int \wedge (\ell = r - 1)))$
- and either $A_n = (int \wedge \lceil Y \rceil \wedge (\ell > \rho))^\frown(\ell < 1))$,
  or $A_n = (int \wedge \lceil \neg Y \rceil \wedge (\ell > \rho))^\frown(\ell < 1))$
- and for $j = 1, \ldots, n-1$ either $A_j = LHR^\frown(\ell = d)$ or $A_j = HLR^\frown(\ell = d)$
- and if $n = 1$ then $A_n = (int \wedge \lceil \neg Y \rceil \wedge (\ell > \rho))^\frown(\ell < 1))$ and if $n > 1$ then $A_1 = LHR^\frown(\ell = d)$ (since at the beginning the signal is low).

Now, the decoding function g can be written as follows. Let D be a formula received by the receiver.

- If $D = (\ell \leq 1 \wedge \ell > 0)^\frown(int \wedge \ell = r - 1)^\frown(\lceil \neg Y \rceil \wedge \ell > \rho \wedge int)^\frown \ell < 1$ then $g(D) = \epsilon$.
- Let $g(D)$ be defined.

- If $D = D'^\frown(\lceil Y \rceil \wedge int \wedge \ell \geq \rho)^\frown \ell < 1$ then
  $g(D'^\frown HLR^\frown(\ell = d)^\frown(\lceil Y \rceil \wedge int \wedge \ell \geq \rho)^\frown \ell < 1) = g(D)1$, and
  $g(D'^\frown HLR^\frown(\ell = d)^\frown(\lceil \neg Y \rceil \wedge int \wedge \ell \geq \rho)^\frown \ell < 1) = g(D)0$.
- If $D = D'^\frown(\lceil \neg Y \rceil \wedge int \wedge \ell \geq \rho)^\frown \ell < 1$, then
  $g(D'^\frown LHR^\frown(\ell = d)^\frown(\lceil Y \rceil \wedge int \wedge \ell \geq \rho)^\frown \ell < 1) = g(D)0$, and
  $g(D'^\frown LHR^\frown(\ell = d)^\frown(\lceil \neg Y \rceil \wedge int \wedge \ell \geq \rho)^\frown \ell < 1) = g(D)1$.

For example, let D be $(\ell \leq 1 \wedge \ell > 0)^\frown(int \wedge \ell = r-1)^\frown LHR^\frown(\ell = d)^\frown LHR^\frown(\ell = d)^\frown HLR^\frown(\ell = d)^\frown(\lceil Y \rceil \wedge \ell > \rho \wedge int)^\frown(\ell < 1))$. Then,

$$\begin{aligned} g(D) &= g((\ell \leq 1 \wedge \ell > 0)^\frown(int \wedge \ell = r-1)^\frown LHR^\frown(\ell = d) \\ &\quad \frown LHR^\frown(\ell = d)^\frown(\lceil Y \rceil \wedge \ell > \rho \wedge int)^\frown(\ell < 1))\,1 \\ &= g((\ell \leq 1 \wedge \ell > 0)^\frown(int \wedge \ell = r-1)^\frown LHR^\frown(\ell = d)^\frown \\ &\quad (\lceil \neg Y \rceil \wedge \ell > \rho \wedge int)^\frown(\ell < 1))\,01 \\ &= g((\ell \leq 1 \wedge \ell > 0)^\frown(int \wedge \ell = r-1) \\ &\quad \frown(\lceil \neg Y \rceil \wedge \ell > \rho \wedge int)^\frown(\ell < 1))\,101 \\ &= \epsilon 101\,. \end{aligned}$$

5.2 Verification of BMP

As said earlier, we have to verify that for any sequence of bits w, if the sender puts on the bus the signal represented by DC formula $f(w)$, then the receiver must receive and receives only the signals represented by a DC formula D for which $g(D) = w$. We can only prove this requirement with some condition on the values of the parameter r, a, b, c, ρ and d. The requirement is formalised as:

For all sequence of bits w,

- there exists a DC formula D received by the receiver such that $f(w) \Rightarrow D$, and
- for all D receivable by the receiver, if $f(w) \Rightarrow D$ then $g(D) = w$.

Since in BMP g is a deterministic function, for any sequence of bits w there is no more than one receivable formula D for which $f(w) \Rightarrow D$. Thus we can have a stronger requirement which is formalised as:

For all sequences of bits w there exists uniquely a receivable formula D such that $f(w) \Rightarrow D$ and $g(D) = w$.

Our verification is done by proving the following two theorems.

Theorem 4. *For any receivable formulas D and D', if D is different from D' syntactically then $\models ((D \wedge D') \Rightarrow ff)$.*

This theorem says that each time at most one receivable formula D is received by the receiver.

Theorem 5. *Assume that $r \geq 1$, $b \geq r+1$, $a \geq r+1$, $c \geq \rho + a$, $d \geq b + r$, $d \leq a + b - 3 - r$, and $\rho \geq a + 1$. Then for any sequence of bits w there exists a receivable formula D for which $f(w) \Rightarrow D$ and $g(D) = w$.*

In [7] we proved these two theorems, with PVS proof checker, with the encoding of the proof system for Duration Calculus.

6 Conclusion

We have presented our approach to the specification and verification of real-time hybrid systems using Duration Calculus. By using temporal propositional letters we can specify many classes of time models that are suitable for our applications. The properties of the introduced temporal propositional letters are then specified by a class of Duration Calculus formulas. Using this class of formulas and the proof system of the original Duration Calculus we can reason about the behaviour of our real-time systems in different time domains without any further efforts for developing a new proof system. We have shown that this technique works well for reasoning about the relationship between real systems and digitised systems. This enables us to use a proof checker of Duration Calculus for different classes of applications.

By embedding discrete DC into the continuous ones, we can use the decidability of the discrete DC to decide the falsifiability of a subclass of formulas in DC. The technique demonstrated in this paper could be considered as an effort for unifying different versions of DC.

References

1. Philip Chan and Dang Van Hung. Duration Calculus Specification of Scheduling for Tasks with Shared Resources. Research Report 44, UNU-IIST, P.O.Box 3058, Macau, June 1995. Published in: Kanchana Kanchanasut and Jean-Jacques Levy (Eds.), *Algorithms, Concurrency and Knowledge*, LNCS 1023, Springer-Verlag 1995, pp. 365–380.
2. Zhou Chaochen and Micheal R. Hansen. *Duration Calculus*. Springer-Verlag, 2004.
3. Zhou Chaochen, C.A.R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1992.
4. Dimitar P. Guelev. A Complete Proof System for First Order Interval Temporal Logic with Projection. Technical Report 202, UNU-IIST, P.O.Box 3058, Macau, June 2000. A revised version of this report was published in the Journal of Logic and Computation, Volume 14, Issue 2, April 2004, pp. 215-249 by Oxford University Press.
5. Dimitar P. Guelev and Dang Van Hung. Prefix and projection onto state in duration calculus. In Oded Maler Eugene Asarin and Sergio Yovine, editors, *Electronic Notes in Theoretical Computer Science*, volume 65. Elsevier Science Publishers, 2002.
6. Dimitar P. Guelev and Dang Van Hung. On the completeness and decidability of duration calculus with iteration. *Theor. Comput. Sci.*, 337(1-3):278–304, 2005.
7. Dang Van Hung. Modelling and Verification of Biphase Mark Protocols in Duration Calculus Using PVS/DC$^-$. Research Report 103, UNU-IIST, P.O.Box 3058, Macau, April 1997. Presented at and published in the Proceedings of the *1998 International Conference on Application of Concurrency to System Design* (CSD'98), 23-26 March 1998, Aizu-wakamatsu, Fukushima, Japan, IEEE Computer Society Press, 1998, pp. 88 - 98.
8. Dang Van Hung. Real-time Systems Development with Duration Calculus: an Overview. Technical Report 255, UNU-IIST, P.O. Box 3058, Macau, June 2002. Published in the proceedings for the *UNU-IIST 10th Anniversary Colloquium* on

Formal Methods at the Crossroads, from Panacea to Foundational Support, LNCS 2757, Springer-Verlag, November 2003, pp. 81-96.

9. Dang Van Hung and Phan Hong Giang. A sampling semantics of duration calculus. In Bengt Jonsson and Joachim Parrow, editors, *Formal Techniques for Real-Time and Fault Tolerant Systems*, volume 1135 of *LNCS*, pages 188–207. Springer-Verlag, 1996.
10. Dang Van Hung and Ko Kwang Il. Verification via Digitized Model of Real-Time Systems. Research Report 54, UNU-IIST, P.O.Box 3058, Macau, February 1996. Published in the Proceedings of *Asia-Pacific Software Engineering Conference 1996* (APSEC'96), IEEE Computer Society Press, 1996, pp. 4–15.
11. Ho Van Huong and Dang Van Hung. Modelling Real-time Database Systems in Duration Calculus. Technical Report 260, UNU-IIST, P.O. Box 3058, Macau, September 2002. Presented at and published in the proceedings of the IASTED International Conference on Databases and Applications (DBA 2004), February 17 – 19, 2004, Innsbruck, Austria, M.H. Hamza (ed.), ACTA Press, pp. 37-42.
12. He Jifeng. A behavioural Model for Co-design. Technical Report 166, UNU-IIST, P.O.Box 3058, Macau, June 1999. Presented at and published in the Proceedings of the World Congress of Formal Methods, Toulouse, France, September, 1999, LNCS 1709, Springer-Verlag, 1999, pp. 1420–1439.
13. Franzle Martin. Synthesizing Controllers of Duration Calculus. In Bengt Jonsson and Joachim Parrow, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 1135 in LNCS, pages 168–187. Springer-Verlag, 1996.
14. Paritosh K. Pandya, Shankara N. Krishna, and Kuntal Loya. On Sampling Abstraction of Continuous Time Logic with Duration Calculus. Technical Report TIFR-PKP-GM-2006/1, Tata Institute of Fundamental Research, India, 2006.
15. Paritosh K. Pandya and Y Ramakrishna. A Recursive Duration Calculus. Technical Report CS-95/3, TIFR, Mumbai, 1995.