On Sampling Abstraction of Continuous Time Logic with Durations

Paritosh K. Pandya[1,⋆], Shankara Narayanan Krishna[2], and Kuntal Loya[2]

[1] Tata Institute of Fundamental Research, India
pandya@tifr.res.in
[2] Indian Institute of Technology, Bombay, India
{krishnas,kloya}@cse.iitb.ac.in

Abstract. Duration Calculus (DC) is a real-time logic with measurement of duration of propositions in observation intervals. It is a highly expressive logic with continuous time behaviours (also called signals) as its models. Validity checking of DC is undecidable. We propose a method for validity checking of Duration Calculus by reduction to a sampled time version of this logic called Well Sampled Interval Duration Logic ($WSIDL$). This reduction relies on representing a continuous time behaviour by a well-sampled behaviour with 1-oversampling. We provide weak and strong reductions (abstractions) of logic DC to logic $WSIDL$ which respectively preserve the validity and the counter models. By combining these reductions with previous work on deciding IDL, we have implemented a tool for validity checking of Duration Calculus. This provides a partial but practical method for validity checking of Duration Calculus. We present some preliminary experimental results to measure the success of this approach.

1 Introduction

Timed behaviours capture how the system state evolves with time. Temporal logics specify properties of such behaviours. Real-time logics deal with quantitative timing properties of timed behaviours.

Timed logics can make use of various notions of time: continuous, sampled (with precise clocks) or discrete. Continuous time, where observable propositions are boolean functions of real-valued time (also called signals), corresponds most naturally to our intuitive notion of timed behaviour. Discrete time, where the set of time points is natural numbered can be appropriate when describing clocked systems such as synchronous circuits. There are other intermediate notions such as timed words [1] which take a sampled view of timed behaviour. The behaviour is given as a sequence of states where each state has a real-valued time stamp.

Real-time logics can be interpreted over these various notions of time and their properties such as expressiveness and decidability also vary accordingly. For example, the well known Metric Temporal Logic (MTL) has been shown

⋆ This work was partially supported by General Motors India Science Lab sponsored project "Advanced Research on Formal Analysis of Hybrid Systems".

O. Grumberg and M. Huth (Eds.): TACAS 2007, LNCS 4424, pp. 246–260, 2007.

to be undecidable for continuous time where as it is decidable for sampled time (for finite behaviours) [11]. Unfortunately, using notions such as sampled time can also lead to counter intuitive behaviour. For example, the Duration Calculus formula $\ell = 3 \wedge \lceil\!\lceil P \rceil\!\rceil$ states that P holds invariantly for 3 time units. (This can be written in MTL as $\Box_{\leq 3} P$.) The DC formula $(\ell = 1 \wedge \lceil\!\lceil P \rceil\!\rceil)^\frown(\ell = 2 \wedge \lceil\!\lceil P \rceil\!\rceil)$ states that P holds invariantly for 1 time unit and this followed by P holding invariantly for 2 more time units. (This can be written in MTL as $\Box_{\leq 1}(P \wedge (\Box_{\leq 2} P))$.) Although intuitively the two properties are the same, unfortunately the two formulae are not equivalent in sampled view of time as intermediate sampling point at time 1 may not be available. With this in mind, Rabinovich and Hirschfeld [8] have argued that continuous time logics should be preferred for real-time requirements. On the other hand, sampled time logics are closer to automata theoretic models and they may have better decidability properties.

In this paper, we consider the abstraction of continuous time properties by sampled time properties while preserving validity or counter-examples. Further abstraction of sampled time properties by discrete time properties has already been considered in literature using notions such as digitization [7,3,10].

We cast our work in context of Duration Calculus [16] which was one of the early real-time logics in computer science to make use of continuous time (or signals). It is an interval temporal logic incorporating the measurement of accumulated duration for which a proposition holds in a time interval. Duration Calculus constitutes a convenient and highly expressive notation for real-time requirements. But this has also made its validity undecidable in general and hard to check in practice. Availability of effective automatic validity and model checking tools for the continuous time Duration Calculus has been a long standing quest. We provide a partial solution to this problem.

There have been many past attempts at deciding Duration Calculus (DC). A discrete time version of DC called DDC (and its extension with state quantification called $QDDC$) were shown to be decidable using a finite automata theoretic decision procedure [12]. A validity and model checking tool called DCVALID has been built for this logic [12,13]. Pandya proposed a sampled time version of DC, called Interval Duration Logic (IDL) [14]. It was argued that this logic, although undecidable in general, is more amenable to automatic validity checking. Amongst the (partial) approaches which are available for validity checking of IDL are bounded validity checking using SMT solvers [15] and abstraction to discrete duration calculus using digitization [3,15]. Both approaches seem effective on many examples of interest. For continuous time Duration Calculus, various decidable subsets have been considered [4,2,17]. But these have not found way into credible tools.

In this paper, we propose a generic version of Duration Calculus $GDC[M]$ whose behaviours are continuous time (signals) but the behaviour is parametrized by a set of admissible time intervals M. By appropriately choosing M, we show that we can define as $GDC[M]$ most variants of DC including DC, IDL, DDC as well as a version of continuous DC without point intervals called $PLDC$, and a special case of IDL called Well Sampled IDL with

1-Oversampling ($WSIDL$). The behaviours of $WSIDL$ are obtained by sampling continuous time behaviours at all change points, all integer valued points and they are oversampled by adding one more point between two consecutive aforementioned points. Logic $WSIDL$ will play a special role in our work here.

As our main result, we show that we can give reductions (abstractions) α^+ and α^- from $PLDC$ to $WSIDL$ which respectively preserve validity and counterexamples. Moreover, we show that logics $PLDC$ and DC have the same expressive power and that there are effective translations between them. Thus, we can analyze continuous time DC properties by reduction to the sampled time logic $WSIDL$. The digitization and bounded validity checking approaches to deciding original IDL easily extend to its variant $WSIDL$. Using these, we have constructed a tool which reduces continuous DC formulae to DDC formulae preserving validity/counter examples. The discrete time validity checking tool DCVALID [12,13] can analyze the resulting formulae. This provides a partial but practical approach for automatically checking the validity of continuous time Duration Calculus formulae. To our knowledge this constitutes amongst the first tools for validity checking a continuous time real-time logic. We give some preliminary experimental results to evaluate the effectiveness of our approach. The results indicate that interesting examples from the Duration Calculus literature can be automatically verified.

The rest of the paper is organized as follows. Section 2 introduces the logic $GDC[M]$ and various Duration Calculi as its instances. The reductions from pointless DC ($PLDC$) to $WSIDL$ is given in Section 3. Section 4 establishes the equivalence of full DC and pointless DC. Section 5 gives a brief overview of past work on reducing (Well-sampled) IDL to Discrete Duration Calculus. Combining all these steps, a partial method for validity checking continuous time DC is formulated in Section 6. Section 7 describes the experimental results obtained by applying the proposed method to some problems of interest.

2 A Variety of Duration Calculi

Duration Calculus is a real-time logic which was originally defined for continuous time finitely variable behaviours [16]. Variants of DC having other forms of time (sampled time, discrete time etc) have also been investigated [14,13].

In this section, we formulate a generic Duration Calculus, GDC, whose behaviours are parametrized by a set of admissible observation intervals I. This allows us to give a uniform treatment of a variety of Duration Calculi which can all be obtained by suitably choosing I.

Let $(\Re^0, <)$ be the set of non-negative real-numbers with usual order. Let $Pvar$ be the set of observable propositions. A behaviour $\theta \in Pvar \to \Re^0 \to \{0, 1\}$. A behaviour θ is *finitely variable* if any proposition changes value only finitely often within any finite time interval. A finitely variable behaviour is called *right continuous* if the value of a proposition P at any time point is same as the value in its small right neighborhood. We omit this obvious definition. We shall restrict

ourselves to finitely variable and right continuous behaviours, and denote the set of all such behaviours by BEH.

Duration calculus is an interval temporal logic with measurements over time intervals. Let $RINTV = \{[b,e] \mid b,e \in \Re^0,\ b \leq e\}$ the set of all intervals over reals. Note that these include point intervals of the form $[b,b]$. The measurement terms mt of GDC have the form $\int P$ or ℓ. The measurement term ℓ denotes the time length of an interval $[b,e]$. The measurement term $\int P$ denotes the accumulated duration for which P is true in θ in an interval $[b,e]$. Formally, the value of measurement term mt is defined as follows: $Eval(\ell)(\theta,[b,e]) = e-b$ and $Eval(\int P)(\theta,[b,e]) = \int_b^e \theta(P)dt$.

Syntax of GDC. Let P range over $Prop$, c over natural numbers, op over comparison operators $\{\leq,<,=,>,\geq,>\}$ and mt over measurement terms. Let D range over GDC formulae with $\top$ denoting the formula "true". The abstract syntax of GDC is given by:

$$\top \mid \lceil\lceil P \rceil \mid \lceil\ \rceil \mid \lceil P \rceil^0 \mid mt\ op\ c \mid D_1 \frown D_2 \mid D_1 \wedge D_2 \mid \neg D_1$$

Semantics For a given behaviour θ, the semantics of formulae is parameterized by a set I of admissible intervals, where $I \subseteq RINTV$. Let the pair (I,θ) be called a segmented behaviour or *s-behaviour*. Let M be a specified set of s-behaviours. We parametrize the semantics of logic GDC by M and denote this by $GDC[M]$. A triple $I,\theta,[b,e]$ where $(I,\theta) \in M$ and $[b,e] \in I$ is called an *M-model*.

For $D \in GDC[M]$ and M-model $I,\theta,[b,e]$ let $I,\theta,[b,e] \models D$ denote that formula D evaluates to true in model $I,\theta,[b,e]$. Omitting the usual boolean cases, this is inductively defined below. For a proposition P and a time point $t \in \Re^0$, let $\theta,t \models P$ denote that the proposition P has value 1 at time point t in behaviour θ. We omit this straightforward definition.

$$\begin{array}{l}
I,\theta,[b,e] \models \lceil P \rceil^0 \text{ iff } b = e \text{ and } \theta,b \models P \\
I,\theta,[b,e] \models \lceil\ \rceil \text{ iff } b = e \\
I,\theta,[b,e] \models \lceil\lceil P \rceil \text{ iff } b < e \text{ and for all } t:\ b \leq t < e.\ \theta,t \models P \\
I,\theta,[b,e] \models mt\ op\ c \text{ iff } Eval(mt)(\theta,[b,e])\ op\ c \\
I,\theta,[b,e] \models D_1 \frown D_2 \text{ iff for some } z : b \leq z \leq e.\ [b,z] \in I \text{ and } [z,e] \in I \\
\qquad\qquad \text{and } I,\theta,[b,z] \models D_1 \text{ and } I,\theta,[z,e] \models D_2
\end{array}$$

Note that in the definition of $\frown$, an interval $[b,e] \in I$ must be chopped into *admissible* sub-intervals $[b,z],[z,e] \in I$.

Derived operators

- $\Diamond D \stackrel{\text{def}}{=} true \frown D \frown true$ holds provided D holds for some admissible subinterval.
- $\Box D \stackrel{\text{def}}{=} \neg\Diamond\neg D$ holds provided D holds for all admissible subintervals.
- Let $ext \stackrel{\text{def}}{=} \neg\lceil\ \rceil$. Define $Unit \stackrel{\text{def}}{=} ext \wedge \neg(ext \frown ext)$. Formula $Unit$ holds for admissible extended intervals which cannot be chopped further into smaller admissible intervals. Let $\bot \stackrel{\text{def}}{=} \neg\top$.

Prefix Validity A prefix model of $D \in GDC[M]$ is an M-model of the form $I, \theta, [0, r]$ such that $I, \theta, [0, r] \models D$. Thus, in prefix models the interval begins at initial time point 0. Also, let $(I, \theta) \models D$ iff for all $[0, r] \in I$, $(I, \theta, [0, r]) \models D$. Finally, $D \in GDC[M]$ is prefix-valid denoted $\models D$ iff $I, \theta, [0, r] \models D$ for all prefix M-models $I, \theta, [0, r]$.

2.1 Duration Calculi

A variety of duration calculi available in the literature can be defined as special cases of $GDC[M]$ by appropriately choosing the set of s-behaviors M, and by syntactically restricting the constructs available in the logic.

Continuous Time Duration Calculus (DC). This is the original Duration Calculus investigated by Zhou, Hoare and Hansen [16]. Duration calculus DC can be defined as $GDC[M_{dc}]$ where $M_{dc} = \{RINTV\} \times BEH$, i.e. in each DC model $(I, \theta, [b, e])$ the set of admissible intervals I is fixed to $RINTV$, the set of all intervals. Because of this, we shall abbreviate $RINTV, \theta, [b, e] \models D$ by $\theta, [b, e] \models_{dc} D$.

Moreover, in the original DC, the atomic formulae of the form $\lceil P \rceil^0$ are disallowed although a more restricted atomic formula $\lceil\ \rceil$ which holds for *all* point intervals is allowed. Thus, syntactically $DC \subset GDC$. It is given by the abstract syntax: $\lceil\lceil P \rceil\ |\ \lceil\ \rceil\ |\ mt\ op\ c\ |\ D_1 \frown D_2\ |\ D_1 \wedge D_2\ |\ \neg D_1$.

Example 1. [Gas burner] Consider the following safety conditions for a gas burner (see [16]) in DC. Let $Des1 \stackrel{\text{def}}{=} \Box(\lceil\lceil Leak \rceil \Rightarrow \ell \leq maxleak)$ and $Des2 \stackrel{\text{def}}{=} \Box(\lceil\lceil Leak \rceil \frown \lceil\lceil \neg Leak \rceil \frown \lceil\lceil Leak \rceil \Rightarrow \ell > minsep)$. The desired requirement is $Concl \stackrel{\text{def}}{=} \Box(\ell \leq winlen \Rightarrow \int Leak \leq leakbound)$. Then, the validity of the formula $G(maxleak, minsep, winlen, leakbound) \stackrel{\text{def}}{=} Des1 \wedge Des2 \Rightarrow Concl$ establishes that the requirement follows from the two safety conditions. □

Pointless Duration Calculus (PLDC). This is a variant of DC without point intervals. Let $EXTINTV = \{[b, e] \in RINTV\ |\ b < e\}$ be the set of extended intervals. Then, $PLDC = GDC[M_{pl}]$ with $M_{pl} = \{EXTINTV\} \times BEH$, i.e. in each $PLDC$ model $(I, \theta, [b, e])$ the set of admissible intervals I is fixed as $EXTINTV$, the set of all non-point intervals. We abbreviate $EXTINTV, \theta, [b, e] \models D$ by $\theta, [b, e] \models_{pl} D$. Syntactically $PLDC \subset GDC[M_{pl}]$ given by the abstract syntax

$$\lceil\lceil P \rceil\ |\ mt\ op\ c\ |\ D_1 \frown D_2\ |\ D_1 \wedge D_2\ |\ \neg D_1.$$

Example 2. Recall that $\Diamond D \stackrel{\text{def}}{=} \top \frown D \frown \top$. In $PLDC$ formula $\Diamond D$ holds for an interval $[b, e]$ if some proper subinterval $[b', e']$ with $b < b' < e' < e$ satisfies D. However, in DC, the formula $\Diamond D$ holds for an interval $[b, e]$ if some subinterval $[b', e']$ with $b \leq b' \leq e' \leq e$ satisfies D. □

Interval Duration Logic (IDL). This logic was proposed by Pandya [14] as a variant of DC with sampled time. It was argued that IDL is more amenable to validity checking. While validity of IDL is also undecidable in general, several effective techniques and tools have been developed as partial methods for validity checking of IDL. These include Bounded Model Checking [15] as well as reduction to the decidable Discrete-time Duration Calculus using digitization [3,15].

Given a behaviour θ, let $C(\theta)$ be the set of time points where the behaviour changes state (including the initial point 0). Let S_θ be such that $C(\theta) \subseteq S_\theta$ where S is a countably infinite set of sampling points which is time-divergent. Such an S_θ gives a set of sampling points such that the behaviour is over-sampled.

Let $INTV(S_\theta) = \{[b, e] \mid b, e \in S_\theta, \ \ b \leq e\}$ be the set of intervals spanning sampling points. Define $M_{idl} = \{(INTV(S_\theta), \theta) \mid \theta \in BEH\}$. Then, we can define $IDL = GDC[M_{idl}]$. The syntax of IDL is same as the syntax of GDC. We will abbreviate $INTV(S_\theta), \theta, [b, e] \models D$ by $S_\theta, \theta, [b, e] \models_{id} D$.

It should be noted that the original IDL [14] was formulated using finite timed-state sequences as models. Here, we reformulate this as continuous behaviour with admissible intervals spanning the sampling points. It can be shown that the two formulations are equivalent.

Well Sampled Interval Duration Logic (WSIDL). This is a special case of IDL where continuous time behaviour is sampled at every change point and at every integer valued point. Moreover the behaviour is also 1-oversampled by including the midpoint between every consecutive pair of above sampling points.

Formally, define $\mathcal{C}(\theta)$ as a set of time points where the behaviour changes state in θ and let $\aleph$ be the set of non-negative integer valued points. Now define $\mathcal{S}'(\theta) = \aleph \cup \mathcal{C}(\theta)$. Also, let Mid contain the midpoints of all consecutive pairs of points in $\mathcal{S}'(\theta)$. Define $WS(\theta) = \mathcal{S}' \cup Mid$. The set $WS(\theta)$ is called the set of *well-sampling points with 1-oversampling*. Here, 1-oversampling refers to the fact that we add one additional point between every pair of consecutive elements of $S'(\theta)$.

Define $WSIntv(\theta) = INTV(WS(\theta))$, the set of intervals spanning elements of $WS(\theta)$. Let, $M_{wsidl} = \{ \ (WSIntv(\theta), \theta) \ \}$. Define $WSIDL = GDC[M_{wsidl}]$. The syntax of $WSIDL$ is same as the syntax of GDC. Note that the set $WS(\theta)$ is uniquely defined by θ. Hence, in a $WSIDL$ model $I, \theta, [b, e]$, the set of intervals I is uniquely determined by θ as $I = WSIntv(\theta)$. Because of this, we shall abbreviate $WSIntv(\theta), \theta, [b, e] \models D$ by $\theta, [b, e] \models_{ws} D$.

Discrete Duration Calculus (DDC). This is a special case of IDL where the formulae are interpreted only over the behaviours where $C(\theta) \subseteq \aleph$, i.e. the behaviours where state changes occur only at integer valued points. Moreover, the set of sampling points are precisely the set of non-negative integers, i.e. $S(\theta) = \aleph$. Let M_{dd} be the subset of s-behaviours of M_{idl} satisfying the above condition. Then, $DDC = GDC[M_{dd}]$. We abbreviate $INTV(S(\theta)), \theta, [b, e] \models D$ by $\theta, [b, e] \models_{dd} D$.

Consider a DC behaviour θ over an interval [1,5] as follows: The points marked Mid are the newly added points which lie in between either 2 change points or a change point and an integer point. The change points are marked with C, and integer points with I.

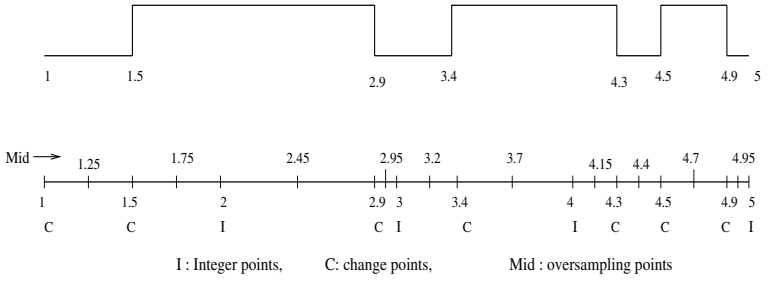

Fig. 1. A DC Behavior θ and the corresponding sampling points in $WS(\theta)$

The prefix validity of DDC (as well as its extension $QDDC$) is decidable and the logic admits a finite-automata theoretic decision procedure [12]. Based on this, a tool DCVALID has been constructed for validity and model checking of DDC formulae [12,13,9].

Example 3. Some essential features of various notions of time, can be specified by some characteristic properties.

- Let $Ax1 \stackrel{\text{def}}{=} \Box\lceil\top\rceil$. This states that every interval is extended. Clearly, $\not\models_{dc} Ax1$ but $\models_{pl} Ax1$.
- Consider the density property $Ax2 \stackrel{\text{def}}{=} \Box(ext \Rightarrow ext^\frown ext)$. It states that any non-point interval can be chopped into two non-point intervals. Then $\models_{dc} Ax2$. However, none of the sampled logics IDL, $WSIDL$, DDC satisfy this formula, e.g. $\not\models_{ws} Ax2$.
- The following property characterizes sampled time. Let $Ax3 \stackrel{\text{def}}{=} \Box(ext \Rightarrow ((Unit^\frown\top)\wedge(\top^\frown Unit))$. Then, all the sampled logics IDL, $WSIDL$, DDC have this axiom as valid. However, $\not\models_{dc} Ax3$.
- Let $Ax4 \stackrel{\text{def}}{=} \Box(Unit \Rightarrow \ell < 1)$. It states that each atomic extended interval is of length less than 1. This is characteristic of well sampled models with 1 oversampling. Thus, $\models_{ws} Ax4$ but $\not\models_{id} Ax4$.
- Discrete time logic DDC is characterized by validity of the following formula. $Ax5 \stackrel{\text{def}}{=} \Box(\ell = 1 \Leftrightarrow Unit)$. It states that every atomic extended interval is of unit length. Then, $\models_{dd} Ax6$ but $\not\models_{ws} Ax6$. □

3 PLDC to WSIDL

In this section, we investigate validity/counterexample preserving reduction (abstraction) from the pointless fragment of DC, i.e. $PLDC$ to the sampled time logic with 1-oversampling, $WSIDL$. This involves reduction of both the models and the formulae.

Sampling Approximation of *DC* Models

Consider a $PLDC$ model $(EXTINTV, \theta, [b, e])$. The s-behaviour $(EXTINTV, \theta)$ can be represented by a $WSIDL$ s-behaviour $(WSIntv(\theta), \theta)$ as explained earlier (see Figure 1).

Definition 1 (1-Sampling). *Given $PLDC$ model θ define a map $f : \Re^0 \rightarrow WS(\theta)$ as follows. Let $f(b) = \left\{ \begin{matrix} b & \text{if } b \in \mathcal{S}', \\ b_m & \text{otherwise.} \end{matrix} \right\}$ where b_m is the midpoint of the smallest number larger than b in $\mathcal{S}'$ and the largest number smaller than b in $\mathcal{S}'$.*

Then, f approximates every time point in θ to a sampling point in $WS(\theta)$. In Figure 1, $f(2.3) = f(2.88) = 2.45$.

Proposition 1. *We list some elementary properties of the onto map f.*

- *f is weakly monotonic, i.e. $b \leq e \Rightarrow f(b) \leq f(e)$. However, it is not strictly monotonic, i.e. $b < e$ does not ensure that $f(b) < f(e)$.*
- *$-0.5 < f(b) - b < 0.5$. This holds since $f(b)$ is either b or the midpoint of two points (on either side of b) at maximum distance 1,*
- *For any θ and any time point b, the state remains constant in the closed intervals $[f(b), b]$ and $[b, f(b)]$.* □

Now we consider a $PLDC$ interval $[b, e]$. This is mapped to its sampling approximation $[f(b), f(e)]$. The above proposition shows that an extended interval $[b, e]$ can be mapped into a point interval $[f(b), f(e)]$ with $f(b) = f(e)$.

Proposition 2. *The effect of sampling on measurements is as follows.*

- $-1 < [eval(\ell)(\theta, [b, e]) \; - \; eval(\ell)(\theta, [f(b), f(e)])] < 1$. *Also,*
- $-1 < [eval(\int P)(\theta, [b, e]) \; - \; eval(\int P)(\theta, [f(b), f(e)])] < 1$

Proof. We prove the first part. The proof of the second part is analogous. Let $l = e - b$ and $l' = f(e) - f(b)$. Let $\triangle e = \mid f(e) - e \mid$. If $e \in \mathcal{S}'$, then $\triangle e = 0$ else $0 \leq \triangle e < 0.5$. Similarly we have $0 \leq \triangle b < 0.5$. Thus length for the IDL interval $[f(b), f(e)]$ will be
$l' = f(e) - f(b) \Rightarrow l' = e \pm \triangle e - (b \pm \triangle b) \Rightarrow l' = l \pm \triangle e \pm \triangle b$
Since $\mid \triangle e + \triangle b \mid < 1, \mid l' - l \mid < 1 \Rightarrow$ There will be less than ± 1 error in the length. □

Approximating PLDC Formulae in WSIDL

We define a strong transformation $\alpha^+ : PLDC \rightarrow WSIDL$ and a weak transformation $\alpha^- : PLDC \rightarrow WSIDL$ as follows. Both these transformations can be computed in linear time.

$PLDC$ formula D	Weak IDL formula $\alpha^-(D)$	Strong IDL formula $\alpha^+(D)$
$\lceil\lceil P\rceil$	$\lceil\lceil P\rceil \vee \lceil P\rceil^0$	$\lceil\lceil P\rceil \vee \lceil P\rceil^0$
$l = k$	$k-1 < l < k+1$	$k-1 \geq l \ \wedge l \geq k+1$, i.e. $\bot$
$l < k$	$l < k+1$	$l \leq k-1$
$l \leq k$	$l < k+1$	$l \leq k-1$
$l > k$	$l > k-1$	$l \geq k+1$
$l \geq k$	$l > k-1$	$l \geq k+1$
$\int P = k$	$k-1 < \int P < k+1$	$k-1 \geq \int P \wedge \int P \geq k+1$, i.e. $\bot$
$\int P < k$	$\int P < k+1$	$\int P \leq k-1$
$\int P \leq k$	$\int P < k+1$	$\int P \leq k-1$
$\int P > k$	$\int P > k-1$	$\int P \geq k+1$
$\int P \geq k$	$\int P > k-1$	$\int P \geq k+1$
$D_1 \wedge D_2$	$\alpha^-(D_1) \wedge \alpha^-(D_2)$	$\alpha^+(D_1) \wedge \alpha^+(D_2)$
$D_1 \frown D_2$	$\alpha^-(D_1) \frown \alpha^-(D_2)$	$\alpha^+(D_1) \frown \alpha^+(D_2)$
$\neg D_1$	$\neg\alpha^+(D_1)$	$\neg\alpha^-(D_1)$

One noteworthy aspect of above abstraction is that a $PLDC$ formula $mt = k$ can only be strongly approximated (using α^+) by $WSIDL$ formula $\bot$. Unfortunately, sampling does not preserve exact measurements.

Theorem 1. *For any $PLDC$ formula D and interval $[b, e] \in EXTINTV$, we have*

1. $\theta, [b, e] \models_{pl} D \quad \Leftarrow \quad \theta, [f(b), f(e)] \models_{ws} \alpha^+(D)$
2. $\theta, [b, e] \models_{pl} D \quad \Rightarrow \quad \theta, [f(b), f(e)] \models_{ws} \alpha^-(D)$

Proof. The proof is by induction on the structure of the formula D. We give some of the cases. The complete proof may be found in the full paper.

1. Let $D \ = \ \ell \ op \ k$.
   We first prove part (2), i.e. $\theta, [b, e] \models_{pl} l \ op \ k \Rightarrow \theta, [f(b), f(e)] \models_{ws} \alpha^-(l \ op \ k)$. Let $l = e-b$ and $l' = f(e)-f(b)$. From Proposition 2, we know that $\mid l-l' \mid < 1$ which implies $l' - 1 < l < l' + 1$. Then,
   $l = k \ \Rightarrow \ k-1 < l' < k+1$, $l < k \ \Rightarrow \ l' < k+1$,
   $l \leq k \ \Rightarrow \ l' < k+1$, $l > k \ \Rightarrow \ l' > k-1$
   $l \geq k \ \Rightarrow \ l' > k-1$. In each case RHS is $\alpha^-(LHS)$.
   We now prove part (1), i.e. $\theta, [b, e] \models_{pl} l \ op \ k \Leftarrow \theta, [f(b), f(e)] \models_{ws} \alpha^+(l \ op \ k)$. Let $l = e - b$ and $l' = f(e) - f(b)$. From Proposition 2, we know that $\mid l - l' \mid < 1$ which implies $l - 1 < l' < l + 1$. Then, $l < k \Leftarrow l' \leq k-1$ and $l > k \Leftarrow l' \geq k+1$.
   We have already proved that $(l > k \ \Rightarrow \ l' > k-1) \Leftrightarrow (\neg(l > k) \Leftarrow \neg(l' > k-1)) \Leftrightarrow (l \leq k \Leftarrow l' \leq k-1)$. Similarly we have proved that $(l < k \Rightarrow l' < k+1) \Leftrightarrow (\neg(l < k) \Leftarrow \neg(l' < k+1)) \Leftrightarrow (l \geq k \Leftarrow l' \geq k+1)$, $l = k \Leftarrow l \leq k \wedge l \geq k \Leftarrow l' \leq k-1 \wedge l' \geq k+1$. In each case RHS is $\alpha^+(LHS)$.

2. Let $D = \neg D_1$. We prove only the part (1).
   $\theta, [f(b), f(e)] \models_{ws} \alpha^+(\neg D_1) \iff$ {Defn. α^+, Semantics}
   $\theta, [f(b), f(e)] \not\models_{ws} \alpha^-(D_1) \implies$ {Induction Hyp.}
   $\theta, [b, e] \not\models_{pl} D_1 \iff$ {Semantics}
   $\theta, [b, e] \models_{pl} \neg D_1$
3. Let $D = D_1 \frown D_2$. We prove only part (1).
   $\theta, [f(b), f(e)] \models_{ws} \alpha^+(D_1 \frown D_2) \iff$ {Defn. α^+}
   $\theta, [f(b), f(e)] \models_{ws} \alpha^+(D_1) \frown \alpha^+(D_2) \iff$ {Semantics.}
   $\exists m \in WS(\theta)$ s.t. $f(b) \leq m \leq f(e)$ and
   $\theta, [f(b), m] \models_{ws} \alpha^+(D_1)$ and $\theta, [m, f(e)] \models_{ws} \alpha^+(D_2) \iff$
   {f is Onto and monotonic}
   $\exists m' \in \Re^0$ s.t. $b \leq m' \leq e$ and $f(m') = m$ and
   $\theta, [f(b), f(m')] \models_{ws} \alpha^+(D_1)$ and $\theta, [f(m'), f(e)] \models_{ws} \alpha^+(D_2) \implies$
   {Induction Hyp.}
   $\exists m' \in \Re^0$ s.t. $b \leq m' \leq e$ and $\theta, [b, m'] \models_{pl} D_1$ and $\theta, [m', e] \models_{pl} D_2 \iff$
   {Semantics of PLDC}
   $\exists m' \in \Re^0$ s.t. $b < m' < e$ and
   $\theta, [b, m'] \models_{pl} D_1$ and $\theta, [m', e] \models_{pl} D_2 \iff$
   {Semantics of PLDC}
   $\theta, [b, e] \models_{pl} D_1 \frown D_2$.

□

Corollary 1. *For any $D \in WSIDL$,*

1. $\models_{ws} \alpha^+(D) \implies \models_{pl} D$
2. $\theta, [b, e] \not\models_{ws} \alpha^-(D) \implies \theta, [b', e'] \not\models_{pl} D$ *for all* $b' \in f^{-1}(b),\ e' \in f^{-1}(e)$.
   In particular, $b \in f^{-1}(b),\ e \in f^{-1}(e)$. *Hence, for any* $[b, e] \in EXTINTV$
   $\theta, [b, e] \not\models_{ws} \alpha^-(D) \implies \theta, [b, e] \not\models_{pl} D$. □

Optimality of 1-Oversampling We now show that as far as preserving validity/counter examples of $PLDC$ formulae is concerned, increasing the oversampling from 1 mid-point to say n intermediate points does not help in making approximations α^+ and α^- more precise. However, later in the paper we consider a *scaling* of *both model and formulae* which can improve the precision of the abstractions $\alpha^+,\ \alpha^-$.

We consider here a case with $n-1$ oversampling points, where n is a natural number, greater than 1. In this general case, $f(b)$ is the oversampling point *closest* to b. Consider a $PLDC$ behaviour with change points at 0, 2, 2.2, 4. Thus, we will have $(0, s_0), (1, s_0),\ (2, s_1), (2.2, s_2), (3, s_2), (4, s_3)$. If we decide to have $n-1$ sample points in between, then we will have the points
$(0, s_0), (\frac{1}{n}, s_0), \ldots, ((n-1)*\frac{1}{n}, s_0), (1, s_0), (1+\frac{1}{n}, s_0), \ldots, (1+(n-1)*\frac{1}{n}, s_0), (2, s_1)$,
$(2+\frac{0.2}{n}, s_1), \ldots, (2+(n-1)*\frac{0.2}{n}, s_1), (2.2, s_2), \ldots, (2.2+(n-1)*\frac{0.8}{n}, s_2), (3, s_2), (3+\frac{1}{n}, s_2), \ldots, (3+\frac{n-1}{n}, s_2), (4, s_3)$.

Now, consider the $PLDC$ interval $[b, e] = [1 + \frac{n-0.7}{n}, 2.2 + \frac{(n-2)*0.8+0.6}{n}]$. The length of this interval in $PLDC$ is $(2.2-1) + \frac{(n-2)*0.8+0.6-(n-0.7)}{n}$, which is equal to $1 + \frac{-0.3}{n}$, which is less than 1.

The corresponding approximated $WSIDL$ interval is $[f(b), f(e)] = [1+\frac{n-1}{n}, 2.2 + \frac{(n-1)*0.8}{n}] = [1 + \frac{n-1}{n}, 2 + \frac{n-0.8}{n}]$. The length of this interval is $2 - 1 + \frac{(n-0.8)-(n-1)}{n}$ which simplifies to $1 + \frac{0.2}{n}$.

Hence, for the given interval $\theta, [b, e] \models_{pl} \ell < 1$ where as $\theta, [f(b), f(e)] \models_{ws} \ell > 1$. This shows that the closest approximation of $\ell < 1$ in logic $WSIDL$ which preserves models is $\alpha^-(\ell < 1) = \ell < 1 + 1$. This holds for all possible n-samplings with $n > 1$.

4 DC to PLDC

Theorem 1 allows us to abstract $PLDC$ formulae to $WSIDL$ formulae. We now show that DC and $PLDC$ have the same expressive power (modulo point intervals). We give a translations $\delta : DC \rightarrow PLDC$ and show that it preserves models.

While logic DC has point intervals, the following proposition shows that DC cannot say anything meaningful about the states at these points. It can be proved by induction on the structure of D.

Proposition 3. *If $\theta, [b, b] \models_{dc} D$ then for all $b' \in \Re^0$ and all $\theta' \in BEH$ we have $\theta', [b', b'] \models_{dc} D$.*

We first define whether a formula D is satisfiable by a point interval and denote this by $Pointsat(D)$.

Definition 2. *$Pointsat : DC \rightarrow \{\top, \bot\}$ is inductively defined as follows.*

$$
\begin{array}{l}
Pointsat(\top) = \top, \;\; Pointsat(\lceil\lceil P \rceil\rceil) = \bot \\
Pointsat(mt \;\; op \;\; c) = \top \;\; \textbf{iff} \;\; (0 \;\; op \;\; c) \\
Pointsat(\neg D) = \neg Pointsat(D), \\
Pointsat(D_1 \wedge D_2) = Pointsat(D_1) \wedge Pointsat(D_2) \\
Pointsat(D_1 \frown D_2) = Pointsat(D_1) \wedge Pointsat(D_2)
\end{array}
$$

For example, by clause 3 we get that $pointsat(\ell <= 3) = (0 \leq 3) = \top$.

Proposition 4. *$Pointsat(D)$ **iff** $\theta, [b, b] \models_{dc} D$ for some $\theta, [b, b]$.*

Using the above we can embed DC in $PLDC$ as follows.

Definition 3. *Let $\delta : DC \rightarrow PLDC$ be inductively defined as follows. Note that size of the output of δ can be exponential in the size of input. The computation time is proportional to the output size.*

$$
\begin{array}{l}
\delta(\lceil \; \rceil) = \bot \\
\delta(X) = X, \;\; for \;\; X \in \{\top, \; \lceil\lceil P \rceil\rceil, \; \ell \; op \; c, \; \int P \; op \; c\}, \\
\delta(\neg D) = \neg\delta(D), \delta(D_1 \wedge D_2) = \delta(D_1) \wedge \delta(D_2),
\end{array}
$$

$$
\begin{array}{rcl}
\delta(D_1 \frown D_2) & = & \delta(D_1) \frown \delta(D_2) \\
& & \vee \;\; \delta(D_1) \wedge Pointsat(D_2) \\
& & \vee \;\; \delta(D_2) \wedge Pointsat(D_1)
\end{array}
$$

Theorem 2. *For all* $\theta \in Beh$ *and* $[b,e] \in EXTINTV$*, we have*

$$\theta, [b,e] \models_{dc} D \quad \textbf{iff} \quad \theta, [b,e] \models_{pl} \delta(D)$$

Proof The proof is by induction on the structure of D. We prove only the case of chop here, the whole proof can be found in the full paper.

- $D = D_1 \frown D_2$. $\theta, [b,e] \models_{dc} D_1 \frown D_2$ iff $\exists m, b \leq m \leq e : \theta, [b,m] \models_{dc} D_1$ and $\theta, [m,e] \models_{dc} D_2$.
  Case 1: $b < m < e$.
  Then, $\theta, [b,m] \models_{dc} D_1$ and $\theta, [m,e] \models_{dc} D_2$. As $[b,m], [m,e] \in EXTINTV$, by the inductive hypothesis, $\theta, [b,m] \models_{pl} \delta(D_1)$ and $\theta, [m,e] \models_{pl} \delta(D_2)$. Thus, $\theta, [b,e] \models_{pl} \delta(D_1) \frown \delta(D_2)$. Conversely, if we assume that $\theta, [b,e] \models_{pl} \delta(D_1) \frown \delta(D_2)$, then $\exists m, b < m < e$ such that $\theta, [b,m] \models_{pl} \delta(D_1)$ and $\theta, [m,e] \models_{pl} \delta(D_2)$. By inductive hypothesis, this implies that $\theta, [b,m] \models_{dc} D_1$ and $\theta, [m,e] \models_{dc} D_2$.
  Case 2: $b < m = e$.
  Then, $\theta, [b,e] \models_{dc} D_1$ and $\theta, [e,e] \models_{dc} D_2$. Then $Pointsat(D_2)$, and by inductive hypothesis, we have $\theta, [b,e] \models_{pl} \delta(D_1)$. Conversely, if $\theta, [b,e] \models_{pl} \delta(D_1)$, then by inductive hypothesis, $\theta, [b,e] \models_{dc} D_1 \Rightarrow \theta, [b,e] \models D_1 \frown D_2$, for $Pointsat(D_2)$.
  Case 3: $b = m < e$. Here, $Pointsat(D_1)$. Similar to Case 2, we have $\theta, [b,e] \models_{dc} D_1 \frown D_2$ iff $\theta, [b,e] \models_{pl} \delta(D_2)$.
  Case 4: $b = m = e$. This case cannot arise as $[b,e] \in EXTINTV$. □

Corollary 2. *For any* $D \in DC$*,*

1. $\models_{dc} D$ **iff** $\models_{pl} \delta(D)$ *and* $Pointsat(D)$.
2. $\neg Pointsat(D)$ *then* $\theta, [b,b] \not\models_{dc} D$ *for any* $\theta, [b,b]$
3. $\theta, [b,e] \not\models_{pl} \delta(D) \quad \Rightarrow \quad \theta, [b,e] \not\models_{dc} D$*, for any* $[b,e] \in EXTINTV$.

Derived Modalities. Applying the translation δ to derived modality we get:

1. If $Pointsat(D)$ then

$$\begin{aligned} \delta(\Box D) &= \neg(\top \frown \neg\delta(D) \frown \top) \\ &\quad \wedge \ \neg(\neg\delta(D) \frown \top) \ \wedge \ \neg(\top \frown \neg\delta(D)) \ \wedge \ \neg\delta(D) \\ \delta(\Diamond D) &= true \end{aligned}$$

2. If $\neg Pointsat(D)$ then

$$\begin{aligned} \delta(\Box D) &= false \\ \delta(\Diamond D) &= (\top \frown \delta(D) \frown \top) \\ &\quad \vee \ (\delta(D) \frown \top) \ \vee \ (\top \frown \delta(D)) \ \vee \ \delta(D) \end{aligned}$$

The reverse translation of $PLDC$ into DC can be found in the full paper.

5 WSIDL to DDC

Validity of sampled time logics $WSIDL$ as well as IDL are undecidable [14] where as validity of discrete time logic DDC is decidable [12,13]. As a partial technique, Chakravorty and Pandya [3,15] have proposed strong and weak translations (abstractions) ST and WT from logic IDL to logic DDC which respectively preserve the validity and the counter examples. These reductions make use of the digitization technique [7,10]. By a small variant of this technique, we can also propose similar reductions from $WSIDL$ to DDC. We omit the details and refer the reader to the original paper [3] for details.

Theorem 3. *We can define linear time computable translations $ST : WSIDL \rightarrow DDC$ and $WT : WSIDL \rightarrow DDC$, and a linear time computable model transformation r from DDC models to $WSIDL$ models such that for any formula $D \in WSIDL$, the following holds.*

1. $\models_{ws} D \quad \Leftarrow \quad \models_{dd} ST(D)$
2. $\theta, [b,e] \not\models_{dd} WT(D) \quad \Rightarrow \quad r(\theta,[b,e]) \not\models_{ws} D.$

6 Validity Checking DC

In order to check the validity of a DC formula D first compute if $Pointsat(D) = \perp$. In this case the formula D is not valid by Corollary 2(2). Otherwise, compute the $PLDC$ formula $\delta(D)$ and proceed as follows:

1. Compute $D' = ST(\alpha^+(\delta(D))$ obtained by applying strong translations of $PLDC$ to $WSIDL$ and then strong translation of $WSIDL$ to DDC. A tool *dc2qddcstrong* has been implemented to compute D' from D.
2. Check the validity of D' using the tool DCVALID.
3. If D' is valid, then by Theorem 3, $\alpha^+(\delta(D))$ is a valid $WSIDL$ formula and by Corollary 1, $\delta(D)$ is a valid $PLDC$ formula. Finally Corollary 2 implies that D is a valid DC formula.
4. If D' is not valid (i.e. DCVALID generates a counter example) then compute $D'' = WT(\alpha^-(\delta(D))$ obtained by first applying the weak translation from $PLDC$ to $WSIDL$ and then applying the weak translation from $WSIDL$ to DDC. A tool *dc2qddcweak* implements translation from D to D''.
5. Validity check D'' using DCVALID. If a counter example $\theta, [b,e]$ is generated for D'' then by Theorem 3, we can infer that $\alpha^-(\delta(D))$ has a counter example $r(\theta,[b,e])$. Then, Corollary 1 implies that $r(\theta,[b,e])$ is a counter example of $\delta(D)$. Corollary 2 tells us that D also has the counter example $r(\theta,[b,e])$.
6. In case D'' is found to be valid (and earlier D' was found invalid), the method fails to conclude anything about the validity of D. However, using the well-known result on the linearity of behaviours [5], we can attempt to infer the validity of D by checking the validity of D_k obtained by suitably scaling up the constants in D by an integer $k > 1$. Theorem 4 below states that the validity of D is preserved by such transformation. The above steps must be iterated for D_k with different values of k.

Theorem 4. *Let* $\theta, [b, e] \models_{dc} D$ *and let* $k \in \Re_{>0}$. *Then* $\theta', [b \cdot k, e \cdot k] \models_{dc} D_k$ *where* D_k *is the DC formula obtained from* D *by replacing each occurrence of* $\int P$ *op* c *by* $\int P$ *op* $c \cdot k$ *and* l *op* c *by* l *op* $c \cdot k$ *and* θ' *is a behaviour satisfying* $\theta'(t) = \theta(\frac{t}{k})$ *for all* $t \in \Re^0$. □

7 Experimental Results

We first illustrate the *DC* validity checking method of the previous section by a simple example.

Example 4. Let $D \stackrel{\text{def}}{=} (\lceil\ \rceil \vee (\lceil\!\lceil P \rceil\!\rceil \frown true) \vee (\lceil\!\lceil \neg P \rceil\!\rceil \frown true))$. Formula $Ax7 \stackrel{\text{def}}{=} \Box D$ is stated as an axiom of *DC* [16], i.e. $\models_{dc} Ax7$. We verify its validity using the method of previous section. We have, $Pointsat(D)$ and $\delta(D) = (\ ((\lceil\!\lceil P \rceil\!\rceil \frown \top) \vee \lceil\!\lceil P \rceil\!\rceil) \vee ((\lceil\!\lceil \neg P \rceil\!\rceil \frown \top) \vee \lceil\!\lceil \neg P \rceil\!\rceil)\ )$. Taking strong translation to $WSIDL$ we obtain that $D' = \alpha^+(\delta(D))$ is $((((\lceil P \rceil^0 \vee \lceil\!\lceil P \rceil\!\rceil) \frown \top) \vee (\lceil P \rceil^0 \vee \lceil\!\lceil P \rceil\!\rceil)) \vee ((((\lceil \neg P \rceil^0 \vee \lceil\!\lceil \neg P \rceil\!\rceil) \frown \top) \vee ((\lceil \neg P \rceil^0 \vee \lceil\!\lceil \neg P \rceil\!\rceil))))$. Then, $Ax7' = \alpha^+(\delta(Ax7))$ is obtained as $\neg(\top \frown \neg D' \frown \top) \wedge \neg(\top \frown \neg D') \wedge \neg(\neg D' \frown \top) \wedge D'$. This is a valid $WSIDL$ formula. We do not give the translation $Ax7''$ of this into DDC which can be found in the full paper. Our tool *dc2qddcstrong* was used to compute the full translation and resulting DDC formula was shown valid using the DCVALID tool taking a total time of 0.05 sec. □

A benchmark example of *DC* formula, the Gas burner problem was presented earlier in Example 1. We have checked the validity of the gas burner formula $G(maxleak, minsep, winlen, leakbound)$ for several instances of the parameters using the validity checking method of the previous section. The times taken for translating the formula into DDC as well as the validity checking time taken by the tool DCVALID are given in Table 1. The modal strength reduction technique [9] was used to optimize the performance of the DCVALID tool. The experiments were run on a 1GHz i686 PC with 1GB RAM running RedHat Linux 9.0. Both valid and invalid instances were tried. For the instance $G(2, 6, 15, 7)$ the method failed to give any result as the strong translation to DDC was invalid but the

Table 1. Results for Gas Burner

Parameters	dc2qddc Strong	DCVALID (hh:mm:ss)	Parameters	dc2qddc Strong/ Weak	DCVALID strong (hh:mm:ss)	DCVALID weak (hh:mm:ss)
Gas Burner: Valid Cases			Gas Burner: Cases with counter examples			
(4,8,30,18)	.3s	02.91s	(2,4,99,6)	.3s	1.25s	1m 22s
(20,40,120,50)	.3s	2m 28.43s	(3,3,150,36)	.3s	18m 37s	19m 31.53s
(1,4,20,12)	.3s	1.50s	(20,40,200,75)	.3s	33m 29.54s	6m 27.55s
(1,4,60,32)	.3s	14.95s	(2,4,500,15)	.3s	2h 5m 3.75s	2h 4m 8.91s
(2,4,100,53)	.3s	1m 1.62s	(5,5,350,25)	.3s	2h 13m 53s	2h 14m 12s
(2,4,300,250)	.3s	20m 39.22s	(7, 3, 175, 27)	.3s	33m 37.47s	32m 57s

weak translation gave valid formula. However, using Theorem 4, and scaling the values of constants by 2, we obtained the instance $G(4, 12, 30, 14)$, for which the strong translation resulted into a valid DDC formula, thereby confirming the validity of the original DC formula $G(2, 6, 15, 7)$.

References

1. R. Alur and D.L. Dill, Automata for modeling real-time systems, *Proc. of 17th ICALP*, LNCS 443, (1990), Springer-Verlag, pp 332-335.
2. A. Bouajjani, Y. Lakhnech and R. Robbana, From Duration Calculus to Linear Hybrid Automata, *Proc.of 7th CAV*, LNCS 939, (1995), Springer-Verlag, pp 196-210.
3. G. Chakravorty and P.K. Pandya, Digitizing Interval Duration Logic, *Proc. of 15th CAV*, LNCS 2725, (2003), Springer-Verlag, pp 167-179.
4. M. Fränzle, Model-Checking Dense-Time Duration Calculus, in *M.R. Hansen (ed.), Duration Calculus: A Logical Approach to Real-Time Systems Workshop proceedings of ESSLLI X*, 1998.
5. M. Fränzle, Take it NP-easy: Bounded Model Construction for Duration Calculus. *Proc. 7th FTRTFT*, LNCS 2469, (2002), Springer-Verlag, pp 245-264.
6. Dang Van Hung and P. H. Giang, Sampling Semantics of Duration Calculus, *Proc. 4th FTRTFT*, LNCS 1135, (1996), Springer-Verlag, pp 188-207.
7. T. A. Henzinger, Z. Manna, and A. Pnueli, What good are digital clocks?, *Proc. 19th ICALP*, LNCS 623, (1992), Springer-Verlag, pp. 545-558.
8. Y. Hirshfeld and A. Rabinovich, Logics for Real-time: Decidability and Complexity, *Fundamenta Informaticae*, 62(1), (2004), pp 1-28.
9. S. N. Krishna and P. K. Pandya, Modal Strength reduction in QDDC, *Proc. 25th FST & TCS*, LNCS 3821, (2005), Springer-Verlag, pp 444-456.
10. J. Ouaknine and J. Worrell, Revisiting Digitization, Robustness and Decidability for Timed Automata, *Proc. 18th IEEE Symposium on LICS*, (2003), pp 198-207.
11. J. Ouaknine and J. Worrell, On Decidability of Metric Temporal Logic, *Proc. 20th IEEE Symposium on LICS*, (2005), pp 188-197.
12. P.K. Pandya, Specifying and Deciding Quantified Discrete-time Duration Calculus Formulae using DCVALID: An Automata Theoretic Approach, in *Proc. RT-TOOLS'2001*, (2001).
13. P.K. Pandya, Model checking CTL*[DC], *Proc. 7th TACAS*, LNCS 2031, (2001), Springer-Verlag, pp 559-573.
14. P.K. Pandya, Interval duration logic: expressiveness and decidability, *Proc. TPTS*, ENTCS 65(6), (2002), pp 1-19.
15. B. Sharma, P.K. Pandya and S. Chakraborty, Bounded Validity Checking of Interval Duration Logic, *Proc. 11th TACAS*, LNCS 3440, (2005), Springer-Verlag, pp 301-316.
16. Zhou Chaochen, C.A.R. Hoare and A.P. Ravn, A Calculus of Durations, *Info. Proc. Letters*, 40(5), 1991.
17. Zhou Chaochen, Zhang Jingzhong, Yang Lu and Li Xiaoshan, Linear duration invariants. in *Proc. 3rd FTRTFT*, LNCS 863, (1994), Springer Verlag, pp 86-109.