Some Decidability Results for Duration Calculus under Synchronous Interpretation

Manoranjan Satpathy[1] and Dang Van Hung[2] and Paritosh K Pandya[3]

[1] Indian Institute of Technology, Guwahati - 781001, India
[2] International Institute for Software Technology, Post Box 3058, Macau
[3] Tata Institute of Fundamental Research, Bombay, India.
email(s): ms@iitg.ernet.in, dvh@iist.unu.edu, pandya@tcs.tifr.res.in

Abstract. Duration Calculus (or DC in short) presents a formal notation to specify properties of real-time systems and a calculus to formally prove such properties. Decidability is the underlying foundation to automated reasoning. But, excepting some of its simple fragments, DC has been shown to be undecidable.
DC takes the set of real numbers to represent time. The main reason of undecidability comes from the assumption that, in a real-time system, state changes can occur at any time point. But an implementation of a specification is ultimately executed on a computer, and there states change according to a system clock. Under such an assumption, it has been shown that the decidability results can be extended to cover relatively richer subsets of DC. In this paper, we extend such decidability results to still richer subsets of DC.

1 Introduction

Duration Calculus (DC) [12] is a logic for reasoning about real-time systems. It presents a formal notation to specify properties of real-time systems and a calculus to formally prove such properties. It uses real numbers to model the continuous time, and time-dependent boolean state expressions to model the behaviours of real-time systems. DC is an extension of *interval logic* (IL) [1] in the sense that it introduces duration constraints on time-dependent boolean state expressions into IL. Intuitively, duration of a state expression in a time interval tells, for what amount of time in the interval, the state expression holds. Since there is no explicit mention of time and timing of state changes, DC provides an ideal platform for reasoning about the requirements of real-time systems at an abstract level, and hence does not have to deal with implementation details. DC has been successfully used to specify and verify many real-time systems; e.g. the deadline driven scheduler [11], a gas burner [3], a railway crossing [9], an auto-pilot [6], and a mine pump controller [5]. One would always hope that such verification be checked mechanically by a model-checker. Decidability is the underlying foundation to model-checking; but, it has been shown that DC in its fullest generality is undecidable. Whatever decidability results have been shown are only for some simple subsets of DC [13].

The main reason of undecidability comes from the assumption that, in a real-time system, a state change can occur at any time point. And the density of real numbers mostly brings in the undecidabilty.

When a design reaches a certain lower level, then it satisfies some extra properties. To illustrate it, let REQ be a DC formula that specifies the requirement of a real-time system. Let $IMPL$ be another DC formula which represents an implementation of the system. What we need to show is whether:

$IMPL \longrightarrow REQ$ is valid in DC. /* $\longrightarrow$ stands for implication */

But the implementation has to be ultimately executed in a computer. In such a case, states have to change at clock points. We can then take clock ticks to be integer points and hence assume that states only change at such integer points [4]. But we will still allow the set of real numbers to represent time. This is because at specification level, the requirements are typically formulated with respect to the real-valued time without any reference to the clock period enforced by the implementation. Let us term the assumption that states do change only at integer points as the *synchronous assumption* (SA). Any interpretation of a DC formula with the synchronous assumption is termed a *synchronous interpretation*, and if formulas of DC are interpreted by *synchronous interpretation*s only, then it is termed *Synchronous Duration Calculus* (SDC). Since at implementation level synchronous assumption always holds, we can now rewrite the above formula as:

$$IMPL \wedge SA \longrightarrow REQ$$

and ask whether it is a valid DC formula. In terms of SDC, we can then ask whether

$IMPL \longrightarrow REQ$ is valid in SDC?

If SDC is decidable, then the validity of the above formula could be checked mechanically. Thus, a good decision procedure for SDC would enable us to carry out mechanical verification of many systems of interest. With this motivation, in this paper, we study the decidability of SDC.

Under synchronous interpretation, some decidability results have been shown for reasonable subsets of DC [2] (these subsets will be defined later in this section). In this paper, we extend such decidability results to richer subsets of DC.

The organisation of the paper is as follows. The remaining part of this section briefly describes the syntax and the semantics of DC, and also various subsets of DC whose decidability issues have been addressed in literature. It also defines DC subsets DC^Q and DC^N whose decidabilities, under synchronous interpretation, are proved respectively in Section 2 and Section 3. Section 4 discusses the importance of the subset DC^Q, and Section 5 concludes the paper.

1.1 The DC syntax

A state expression S is defined by the following BNF.

$$S ::= 0 \mid 1 \mid v \mid \neg S \mid S \vee S$$

[4] Later in the paper, we will generalize this to the case where clock period is any fixed rational number

where 0 and 1 are boolean constants representing $true$ and $false$, and $v \in SVar$, the set of state variables.
A term t in DC is defined by:

$$t ::= c \mid x \mid \int P$$

where, c stands for a constant (usually a real number), x is a global variable (independent of time) and P is a state expression.
A formula D is defined by:

$$D ::= true \mid t_1 \circ t_2 \mid \neg D \mid D \vee D \mid D^\frown D \mid \forall x.D,$$

where x is a global variable, and $\circ$ stands for any binary relational operator like $=, <, >$ etc. The connectives $\neg$ and $\vee$ here are semantically different from those used in state expressions.

1.2 The Semantics of DC

We will take $\mathcal{R}^+$, the set of non-negative reals, as the time domain and shall refer to it by $Time$. A time interval $[b, e]$ with $b, e \in Time$ is defined by $\{t \mid b \leq t \leq e\}$. $Intv$ denotes the set of time intervals (or simply the intervals). The set $\{tt, ff\}$ is denoted by $Bool$, where tt is identified with $true$ and ff is identified with $false$. Q^+ denotes the set of non-negative rational numbers.

An interpretation Θ associates a total function $\Theta_v : (Time \Rightarrow Bool)$ with each state variable v. It is assumed that such functions have only finite number of discontinuity points in any finite interval. An interpretation Θ can be extended to $\Theta[\![S]\!] : Time \Rightarrow Bool$, for any state S in the obvious way.

A valuation is a pair $(\mathcal{V}, [b, e])$ where the value assignment $\mathcal{V}$ assigns $\mathcal{V}(x) \in Time$ to each global variable x, and $[b, e] \in Intv$. Two valuations $\mathcal{V}_1$ and $\mathcal{V}_2$ are said to be x-equivalent if $\forall y \in Var$, the set of global variables, and $y \neq x$; $\mathcal{V}_1(y) = \mathcal{V}_2(y)$. Val denotes the set of valuations. The semantics of a term t in an interpretation Θ is a function $\Theta[\![t]\!] : Val \to \mathcal{R}$ defined by:

$\Theta[\![c]\!](\mathcal{V}, [b, e]) = c$
$\Theta[\![x]\!](\mathcal{V}, [b, e]) = \mathcal{V}(x)$
$\Theta[\![\int S]\!](\mathcal{V}, [b, e]) = \int_b^e \Theta[\![S]\!](t)dt$

In the last definition involving integration, $\Theta[\![S]\!](t)$ evaluates to a boolean value. For the purpose of integration, $true$ is identified with 1 and $false$ with 0. Semantics of a formula D in an interpretation Θ is a function $\Theta[\![D]\!] : Val \to \{tt, ff\}$ defined by:

$\Theta[\![true]\!](\mathcal{V}, [b, e]) = tt$	
$\Theta[\![t_1 \circ t_2]\!](\mathcal{V}, [b, e]) = tt$	iff $\Theta[\![t_1]\!](\mathcal{V}, [b, e]) \circ \Theta[\![t_2]\!](\mathcal{V}, [b, e]) = tt$
$\Theta[\![\neg D]\!](\mathcal{V}, [b, e]) = tt$	iff $\Theta[\![D]\!](\mathcal{V}, [b, e]) = ff$
$\Theta[\![D_1 \vee D_2]\!](\mathcal{V}, [b, e]) = tt$	iff $\Theta[\![D_1]\!](\mathcal{V}, [b, e]) = tt$ or $\Theta[\![D_2]\!](\mathcal{V}, [b, e]) = tt$
$\Theta[\![D_1^\frown D_2]\!](\mathcal{V}, [b, e]) = tt$	iff $\Theta[\![D_1]\!](\mathcal{V}, [b, m]) = tt$ and $\Theta[\![D_2]\!](\mathcal{V}, [m, e]) = tt$ for some $m \in Time$ and $b \leq m \leq e$.
$\Theta[\![\forall x.D]\!](\mathcal{V}, [b, e]) = tt$	iff $\Theta[\![D]\!](\mathcal{V}', [b, e]) = tt$ for all valuations $\mathcal{V}'$ x-equivalent with $\mathcal{V}$.

A formula D holds (or, is *true*) in interpretation Θ (written as $\Theta \models D$) iff $\Theta[\![D]\!](\mathcal{V},[b,e]) = tt$ for every valuation $(\mathcal{V},[b,e])$. D is *valid* (written as $\models D$) iff $\Theta \models D$ for every interpretation Θ. A formula D is $satisfiable$ iff there exists an interpretation Θ and a valuation $(\mathcal{V},[b,e])$ such that $\Theta[\![D]\!](\mathcal{V},[b,e]) = tt$. It is trivial to show that D is satisfiable iff $\neg D$ is not valid. When we consider a propositional formula, i.e. a formula without quantifiers and global variables, then we drop the *value assignment* component of a valuation. By $\Theta,[b,e] \models D$, we then mean that formula D holds in interval $[b,e]$ under interpretation Θ.

The following abbreviations for certain duration formulas are heavily used.

$\ell \stackrel{\text{def}}{=} \int 1$	denotes "length of an interval".
$\lceil\!\rceil \stackrel{\text{def}}{=} \ell = 0$	denotes "a point interval".
$\lceil P \rceil \stackrel{\text{def}}{=} (\int 1 > 0) \wedge (\neg \int \neg P > 0)$	denotes "P holds almost everywhere in a non-point interval"
$\Diamond D \stackrel{\text{def}}{=} true^\frown D^\frown true$	denotes "D holds in some sub-interval"
$\Box D \stackrel{\text{def}}{=} \neg\Diamond(\neg D)$	denotes "D holds in every subinterval"

1.3 Subsets of DC

Various subsets of DC are defined by restricting formulas. In subset $\{\lceil P \rceil\}$, its formulas are generated from primitive formulas of the form $\lceil P \rceil$ and by using connectives $\neg, \vee$ and $^\frown$. Similarly, the subsets $\{\lceil P \rceil, \ell = k\}$ and $\{\int P = \int Q\}$ are defined. Formulas in subset $\{\int P < k, \int P > K\}$ include primitive formulas of the form $\int P < k$, $\int P > k$ and the formulas obtained from them by using operators $\neg, \vee$ and $^\frown$. Formulas in subset $\{\forall x, \lceil P \rceil, l = x\}$ are built from primitive formulas $\lceil P \rceil$ and $\ell = x$, using connectives $\neg, \vee, ^\frown$ and $\forall x$.
In this paper, we will concentrate on the subset of DC whose formulas are described by:

$$D ::= \int P < k \mid \int P > k \mid \neg D \mid D \vee D \mid D^\frown D$$

When $k \in \mathcal{N}$, we will address the above set of formulas as DC^N, and when $k \in Q^+$, we will address the same subset as DC^Q. This subset properly includes $\{\lceil P \rceil, \ell = k\}$, since

$$\lceil P \rceil \equiv (\int true > 0) \wedge (\neg \int \neg P > 0), \text{ and}$$
$$\ell = k \equiv \neg(\int true < k \ \vee \ \int true > k).$$

1.4 Related work

The following table tabulates the decidability results of the original DC under discrete and dense time DC [13]. Dense time DC is the DC we have described earlier. Under discrete time interpretation, states change only at integer points, and the set of intervals is defined by $\{[b,e] \mid (b,e \in \mathcal{N})\ and\ (b \leq e)\}$.

As mentioned earlier, when states change only at integer points, the interpretation is termed synchronous. Under synchronous interpretation, Fränzle [2] has shown the decidability of the DC subset $\{\int P < k, \int P > k\}$ $(k \in \mathcal{N})$.

DC subset	$\lceil P \rceil$	$\lceil P \rceil, \ell = k$	$\int P = \int Q$	$\lceil P \rceil, \forall x, \ell = x$
Discrete time	Decidable	Decidable	Undecidable	Undecidable
Dense time	Decidable	Undecidable	Undecidable	Undecidable

1.5 Our work

Under synchronous interpretation, decidability of the subset $\{\int P < k, \int P > k\}$ ($k \in \mathcal{N}$) is the best decidability result shown so far. In this paper, we make an extension to this subset. we remove the restriction that $k \in \mathcal{N}$, and show that, even when $k \in Q^+$, the resulting subset, i.e. DC^Q, is also decidable under synchronous interpretation. Then, we give an alternative proof of Fränzle's result that is simpler than the proof in [2].

2 Decidability of DC^Q

For showing the decidability of DC^Q, we will need some results from Number Theory. In the following discussion, for integers $a \geq 0$ and $b > 0$, $gcd(a, b)$ denotes the *greatest common divisor* of a and b. And for two positive integers a and b, $lcm(a, b)$ denoted the *least common multiple* of a and b. For two positive rational numbers p_1/q_1 and p_2/q_2, we define their *greatest common divisor* $gcd(p_1/q_1, p_2/q_2)$ as follows. Let $p_1/q_1 = p_1'/lcm(q_1, q_2)$ and $p_2/q_2 = p_2'/lcm(q_1, q_2)$. Then,

$$\begin{aligned} gcd(p_1/q_1) &\stackrel{\text{def}}{=} p_1/q_1. \\ gcd(p_1/q_1, p_2/q_2) &\stackrel{\text{def}}{=} gcd(p_1', p_2')/lcm(q_1, q_2), \\ gcd(p_1/q_1, \ldots, p_m/q_m) &\stackrel{\text{def}}{=} gcd(gcd(p_1/q_1, \ldots, p_{m-1}/q_{m-1}), p_m/q_m) \\ &\quad \text{(for } m > 2\text{)}. \end{aligned}$$

For example,

$gcd(15/22, 5/8) = gcd(60, 55)/88 = 5/88$
$gcd(15/22, 5/8, 5/11) = gcd(5/88, 5/11) = 5/88$

For a formula $D \in DC^Q$, let $r_1, r_2 \ldots, r_m$ be the set of rational numbers occurring in D. Let $p_1/q_1, p_2/q_2 \ldots, p_m/q_m$ be respectively their fractional parts; i.e. $\forall i,\ p_i/q_i < 1$. Without any confusion, we will term $gcd(p_1/q_1, \ldots, p_m/q_m, 1)$ as $gcd(D)$.

As an illustration, consider the DC formula: $D = (\int P > 15/22)^\frown(\int P < 5/8)^\frown(l = 5/11)$
Then $gcd(D) = gcd(15/22, 5/8, 5/11, 1) = 1/88$

Definition 1. *Let g be any positive rational and Θ is any interpretation (not necessarily synchronous). Then define a new interpretation Θ_g by:*

$$\Theta_g(P)(t) = \Theta(P)(gt) \qquad \text{for all } t \in Time \text{ and for all states } P.$$

Definition 2. *Let g be any positive rational and $D \in DC^Q$. Then from D obtain a new formula $D[g] \in DC^Q$, where each k in D gets replaced by k/g. More formally:*

$$
\begin{array}{lcl}
(\int P < k)\,[g] & \stackrel{\text{def}}{=} & \int P < (k/g) \\
(\int P > k)\,[g] & \stackrel{\text{def}}{=} & \int P > (k/g) \\
(\neg D)\,[g] & \stackrel{\text{def}}{=} & \neg(D[g]) \\
(D_1 \vee D_2)\,[g] & \stackrel{\text{def}}{=} & D_1[g] \vee D_2[g] \\
(D_1 \frown D_2)\,[g] & \stackrel{\text{def}}{=} & D_1[g] \frown D_2[g]
\end{array}
$$

Theorem 1. *Let g be a positive rational, $D \in DC^Q$, Θ be an interpretation and $[b,e] \in Intv$. Then* $\Theta, [b,e] \models D$ *iff* $\Theta_g, [b/g, e/g] \models D[g]$

Proof: By induction over the structure of D. We consider here two important cases.

Case 1: D is of the form $\int P < k$ (k is a positive rational).
To show that: $\Theta, [b,e] \models \int P < k$ iff $\Theta_g, [b/g, e/g] \models \int P < k/g$
Define for $t \in Time$, $t = gt'$. Then $dt = gdt'$.
Now $\Theta, [b,e] \models \int P < k$
iff $\int_b^e \Theta(P)(t)dt < k$
iff $\int_{b/g}^{e/g} \Theta(P)(gt')gdt' < k$
iff $\int_{b/g}^{e/g} \Theta_g(P)(t')dt' < k/g$
iff $\Theta_g, [b/g, e/g] \models \int P < k/g$.

Case 2: D is of the form $D_1 \frown D_2$
To show that $\Theta, [b,e] \models D_1 \frown D_2$ iff $\Theta_g, [b/g, e/g] \models (D_1 \frown D_2)[g]$
Assume $\Theta, [b,e] \models D_1 \frown D_2$
i.e. $\exists m : b \leq m \leq e$ such that $\Theta, [b,m] \models D_1$ and $\Theta, [m,e] \models D_2$
Using hypothesis, $\Theta_g, [b/g, m/g] \models D_1[g]$ *and* $\Theta_g, [m/g, e/g] \models D_2[g]$
i.e. $\Theta_g, [b/g, e/g] \models D_1[g] \frown D_2[g]$
Proof of the converse is similar. □

Lemma 1. *For any positive rational r and for any interpretation Θ under which states change only at points in $\{ir \mid i \in \mathcal{N}\}$, Θ_r is synchronous (states change only at integer points).*

Proof: Under interpretation Θ, if states change at ticks of a clock with period r, then Θ_r just scales the clock period by $1/r$. □
In figure 1(a), under interpretation Θ, states change only at points $\{i*(5/7) \mid i \in \mathcal{N}\}$.
Figure 1(b) shows Θ_r, where states change only at integer points.

Theorem 2. *Let $D \in DC^Q$, Θ be a synchronous interpretation, $[b,e] \in Intv$. Then*

$\Theta, [b,e] \models D$ *iff* $\Theta_{gcd(D)}, [b/gcd(D), e/gcd(D)] \models D[gcd(D)]$,

where $D[gcd(D)] \in DC^N$ and $\Theta_{gcd(D)}$ is synchronous.

Proof: For any constant k occurring in D as length, from the definition on $gcd(D)$, it is obvious that $k/gcd(D)$ is an integer. So $D[gcd(D)] \in DC^N$. The rest of the theorem is a direct consequence of Theorem 1 and Lemma 1. □

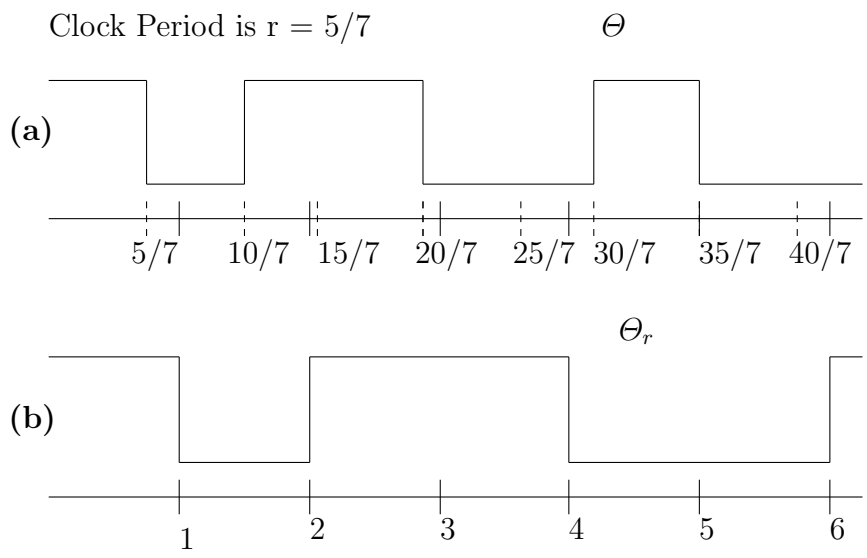

Fig. 1. (a) Θ has clock period of 5/7 (b) Θ_r is synchronous

Definition 3. *Let r be a positive rational, Θ be an interpretation. If states change only at points in $\{ir \mid i \in \mathcal{N}\}$ under Θ, then the interpretation Θ is termed r–synchronous.*

Definition 4. *(i) A formula $D \in DC^Q$ is s–valid iff for all synchronous interpretation Θ, $\Theta \models D$.*
(ii) A formula $D \in DC^Q$ is rs-valid iff for all r-synchronous interpretation Θ_r, $\Theta_r \models D$

Theorem 3. *For all $D \in DC^Q$, s–validity of D is decidable.*

Proof: For a formula $D \in DC^N$ and for synchronous interpretations, validity of D is known to be decidable (proof in next section). And from Theorem 2, validity of $D \in DC^Q$ under synchronous interpretations can be reduced to validity of $D[gcd(D)] \in DC^N$ under synchronous interpretations. □

Theorem 4. *$\forall\ D \in DC^Q$, rs–validity of D is decidable.*

Proof: Let Θ be a r–synchronous interpretation. Then from Theorem 1 and Lemma 1:

$$\Theta, [b, e] \models D \text{ iff } \Theta_r, [b/r, e/r] \models D[r],$$

where Θ_r is synchronous and $D[r] \in DC^Q$. Now take $g = gcd(D[r])$. From Theorem 2,

$$\Theta_r, [b/r, e/r] \models D[r] \text{ iff } \Theta_{rg}, [b/rg, e/rg] \models D[r][g],$$

where Θ_{rg} is synchronous and $D[r][g] \in DC^N$. So rs–validity of a formula $D \in DC^Q$ could be reduced to the s-validity of formula $D[r][gcd(D)] \in DC^N$. □

3 Decidability of DC^N

In this section, we will show the decidability of the validity of a formula $D \in DC^N$ under synchronous interpretation. Fränzle [2] has shown that the DC subset $\{\int P < k, \int P > k\}$ is decidable under synchronous interpretation. In this section, we give an alternative proof of the same result but it is much simpler. To keep our proof simple, we will first prove the decidability result for a subset of DC^N (and call it DC^s) whose formulas are described by:

$D ::= \lceil\!\lceil P \rceil\!\rceil \mid \neg D \mid D \vee D \mid D^\frown D \mid \ell = k \quad$ /* $k \in \mathcal{N}$ */.

Then we will extend the decidability of DC^s to that of DC^N.

Definition 5. *Let* $[b,e]\,,\,[b',e'] \in Intv$. *We say that* $[b',e'] \approx [b,e]$ *iff*

1. $\lceil b' \rceil = \lceil b \rceil$ *and* $\lfloor b' \rfloor = \lfloor b \rfloor$
2. $\lceil e' \rceil = \lceil e \rceil$ *and* $\lfloor e' \rfloor = \lfloor e \rfloor$
3. $\lfloor e - b \rfloor = \lfloor e' - b' \rfloor$ *and*
4. $\lceil e - b \rceil = \lceil e' - b' \rceil$

It can be seen that $\approx$ is an equivalence relation. We will state the following lemma without proof. The proof is given in [8].

Lemma 2. *For arbitrary intervals* $[b,e]$ *and* $[b',e']$ *such that* $[b,e] \approx [b',e']$, *for any real number* $m \in [b,e]$ *(i.e.* $b \le m \le e$*) there exists a real number* $m' \in [b',e']$ *such that* $[b,m] \approx [b',m']$ *and* $[m,e] \approx [m',e']$ □

Lemma 3. *Let* D *be a formula in* DC^s, $[b',e']$ *and* $[b,e]$ *be intervals such that* $[b',e'] \approx [b,e]$, *and* Θ *be a synchronous interpretation. Then*

$$\Theta\,,[b,e] \models D \;\Leftrightarrow\; \Theta\,,[b',e'] \models D\,.$$

Proof: By structural induction over D. We will consider two important cases.

Case 1: D is of the form $\lceil\!\lceil P \rceil\!\rceil$:

$\Theta\,,[b,e] \models \lceil\!\lceil P \rceil\!\rceil$
$\Leftrightarrow\ b < e \ \wedge\ \forall t : b < t < e;\ \ \Theta(P)(t) = true$
$\Leftrightarrow\ \lfloor b \rfloor < \lceil e \rceil \ \wedge\ \forall t : \lfloor b \rfloor < t < \lceil e \rceil;\ \ \Theta(P)(t) = true$
/* since Θ is synchronous*/
$\Leftrightarrow\ \lfloor b' \rfloor < \lceil e' \rceil \ \wedge\ \forall t : \lfloor b' \rfloor < t < \lceil e' \rceil;\ \ \Theta(P)(t) = true$
/* $\lfloor b \rfloor = \lfloor b' \rfloor$ and $\lceil e \rceil = \lceil e' \rceil$ */
$\Leftrightarrow\ b' < e' \ \wedge\ \forall t : b' < t < e';\ \ \Theta(P)(t) = true$
$\Leftrightarrow\ \Theta\,,[b',e'] \models \lceil\!\lceil P \rceil\!\rceil$

Case 2: D is of the form $\ell = k$ and $k \in \mathcal{N}$:

Given $[b,e] \approx [b',e']$. From the definition of equivalence, and since $e - b$ is a non-negative integer, we have $e' - b' = e - b = k$. So $\Theta\,,[b,e] \models (\ell = k) \ \Leftrightarrow\ \Theta\,,[b',e'] \models (\ell = k)$. □

3.1 Decidability of DC^s

Given a formula $D \in DC^s$, we build a regular language $\mathcal{L}(D)$. Then we show that a synchronous interpretation Θ satisfies D in an interval $[b, e]$ iff there exists a corresponding word $w \in \mathcal{L}(D)$. Let S be the set of state variables in formula D. Each word w in language $\mathcal{L}(D)$ will consist of a string s made up of characters in alphabet

$\Sigma = 2^S$ /* Power set of S */

The intention is that we let $a \in \Sigma$ represent that all state variables in a hold for one time unit (from one integer point to the next).

Definition 6. *Let $s = a_1 a_2 \ldots a_n$ be a string over Σ. We say that s corresponds to a synchronous interpretation Θ in interval $[b, e]$, $b \neq e$ iff $n = \lceil e \rceil - \lfloor b \rfloor$ and for all $i = 1, \ldots, n$ $\Theta[\![\lceil P \rceil]\!](t) = true$ for all $t \in [\lfloor b \rfloor + i - 1, \lfloor b \rfloor + i]$ iff $P \in a_i$. Thus, the string corresponding to a synchronous interpretation Θ in interval $[b, e]$ is defined uniquely and will be denoted by $word(\Theta, [b, e])$. By our convention, $word(\Theta, [b, b]) = \epsilon$.*

Clearly, for each string $s = a_1 a_2 \ldots a_n$ over Σ, $n \geq 0$, we can find a synchronous interpretation Θ and an interval $[b, e]$ such that $s = word(\Theta, [b, e])$. For the following discussion, let Δx denote the fraction of the real number x, i.e. $x = \lfloor x \rfloor + \Delta x$. Further, let for a real number x, δx denote $\lceil x \rceil - x$ (thus, $x = \lfloor x \rfloor + \Delta x = \lceil x \rceil - \delta x$). Let us define:

$\Sigma_\delta = \{\delta_{01H}, \delta_{01R}, \delta_{01L}, \delta_{01}, \delta_{12}, \delta_{00}, \delta_{11}, \delta_\epsilon\}$

The elements of Σ_δ represent the following facts about an interval $[b, e]$ (any interval should be of one of the following types):

- δ_{01H} : $b, e \notin \mathcal{N}$; $\lfloor b \rfloor = \lfloor e \rfloor$; $\lceil b \rceil = \lceil e \rceil$ and $b \neq e$ (or, equivalently, $\lfloor b \rfloor = \lfloor e \rfloor$ and $\Delta b < \Delta e$)
- δ_{01R} : $b \in \mathcal{N}$ and $e \notin \mathcal{N}$ (or, equivalently, $0 = \Delta b < \Delta e$)
- δ_{01L} : $b \notin \mathcal{N}$ and $e \in \mathcal{N}$ (or, equivalently, $0 = \Delta e < \Delta b$)
- δ_{01} : $b, e \notin \mathcal{N}$ and $0 < \delta b + \Delta e < 1$ (or, equivalently, $0 < \Delta e < \Delta b$)
- δ_{12} : $b, e \notin \mathcal{N}$ and $1 < \delta b + \Delta e < 2$ (or, equivalently, $0 < \Delta b < \Delta e$)
- δ_{00} : $b, e \in \mathcal{N}$ and $b \neq e$ (or, equivalently, $0 = \Delta b = \Delta e$ and $b \neq e$).
- δ_{11} : $b, e \notin \mathcal{N}$; $b \neq e$ and $\delta b + \Delta e = 1$ (or, equivalently, $0 < \Delta b = \Delta e$)
- δ_ϵ : Point intervals.

Note that if an interval is of type δ then any other interval equivalent to it is also of type δ. Conversely, all intervals of the same type with the same floor of the beginning point and with the same floor of the ending point are equivalent.

3.2 The Language definition

For a formula D, for each $\delta \in \Sigma_\delta$ we are now going to construct a language $\mathcal{L}_\delta(D)$ such that for a synchronous interpretation Θ, for an interval $[b, e]$ of type δ, it holds that $\Theta, [b, e] \models D$ iff $word(\Theta, [b, e]) \in \mathcal{L}_\delta(D)$. Let us denote by $\mathcal{L}_\delta$ the set of all strings that can correspond to a synchronous interpretation in an interval of the type δ:

$$
\begin{aligned}
\mathcal{L}_{\delta_{00}} &= \Sigma^+ \\
\mathcal{L}_{\delta_{01H}} &= \Sigma \\
\mathcal{L}_{\delta_{01R}} &= \Sigma^*\,\Sigma \\
\mathcal{L}_{\delta_{01L}} &= \Sigma\,\Sigma^* \\
\mathcal{L}_{\delta_{\epsilon}} &= \{\epsilon\} \\
\mathcal{L}_{\delta_{01}} &= \Sigma\,\Sigma^*\,\Sigma \\
\mathcal{L}_{\delta_{11}} &= \Sigma\,\Sigma^*\,\Sigma \\
\mathcal{L}_{\delta_{12}} &= \Sigma\,\Sigma^*\,\Sigma
\end{aligned}
$$

$\mathcal{L}_\delta(D)$ are defined over the structure of D.

$$\mathcal{L}_i(\lceil\!\rceil) = \begin{cases} \{\epsilon\} \text{ if } i = \delta_\epsilon \\ \emptyset \quad \text{otherwise} \end{cases}$$

In the context of formula $\lceil P \rceil$, let $DNF(P)$ stand for the set of conjuncts in the *disjunctive normal form* of predicate P. So if $a \in DNF(P)$ is *true*, it also implies that predicate P is true. With these in the background, the definition of $\mathcal{L}(\lceil P \rceil)$ should be obvious.

$$\mathcal{L}_i(\lceil P \rceil) = \begin{cases} DNF(P)^+ & \text{if } i \in \{\delta_{00}, \delta_{01R}, \delta_{01L}\} \\ DNF(P) & \text{if } i = \delta_{01H} \\ DNF(P)\,.\,DNP(P)^*\,.\,DNF(P) & \text{if } i \in \{\delta_{01}, \delta_{12}, \delta_{11}\} \\ \emptyset & \text{if } i = \delta_\epsilon \end{cases}$$

When the formula is of the form $\ell = k$, then an interval satisfies D iff either it starts and ends with integer points with length k, or it can have fractional parts at both ends with the length of their sum equals to 1 and the length of the integral part will be $k - 1$.

$$\mathcal{L}_i(\ell = k) = \begin{cases} \Sigma^k & \text{if } i = \delta_{00} \\ \Sigma.\Sigma^{k-1}.\Sigma & \text{if } i = \delta_{11} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{L}_i(\neg D) \;=\; \mathcal{L}_i - \mathcal{L}_i(D)$$

$$\mathcal{L}_i(D_1 \vee D_2) \;=\; \mathcal{L}_i(D_1) \cup \mathcal{L}_i(D)$$

When a formula D is of the form $D_1^\frown D_2$, the language for $\mathcal{L}(D)$ should be defined in terms of $\mathcal{L}(D_1)$ and $\mathcal{L}(D_2)$. An interval of type γ satisfying D will be chopped into to an interval of type α satisfying D_1 and an interval of type β satisfying D_2. All the possibilities of $\gamma,\ \beta,\ \alpha$ are given in the table in figure 3, which is obvious from the definition of the elements of Σ_δ. If an interval is chopped into two subintervals such that the first interval is of type $\alpha = \delta_{01L}$ and then the second is of type $\beta = \delta_{01R}$ then γ, the type of the original interval, must be either of δ_{01}, δ_{11} or δ_{12}. That is why the corresponding entry for γ in the table has three such entries. On the other hand, if $\alpha = \delta_{00}$ and $\beta = \delta_{01R}$ then it must be the case that $\gamma = \delta_{01R}$. Similarly, all other entries in the table could be explained.

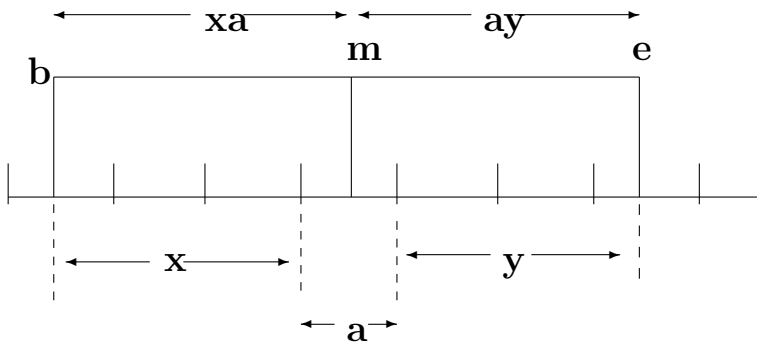

Fig. 2. A string represents chopped intervals

For an interval $[b, e]$ ($b \neq e$), for any synchronous interpretation Θ, $\Theta, [b, e] \models D$ iff for some $m \in [b, e]$, $\Theta, [b, m] \models D_1$ and $\Theta, [m, e] \models D_2$. If $b < m < e$, for each unit interval in side $[\lfloor b \rfloor, \lceil m \rceil]$ and $[\lfloor m \rfloor, \lceil e \rceil]$, there is a character in Σ representing Θ. Thus, if $\lfloor m \rfloor < \lceil m \rceil$ ($[b, m]$ is of one of the types $\delta_{01}, \delta_{12}, \delta_{11}, \delta_{01R}, \delta_{01H}$ and $[m, e]$ is of one the types $\delta_{01}, \delta_{12}, \delta_{11}, \delta_{01L}, \delta_{01H}$), the character in Σ to represent Θ for the unit interval $[\lfloor m \rfloor, \lceil m \rceil]$ is the last character of the string corresponding to Θ in the interval $[b, m]$ and also the first character of the string corresponding to Θ in the interval $[m, e]$ (see Figure 2). Furthermore, the string representing point intervals can only be ϵ. Therefore, we define $\mathcal{L}(D)$ as follows.

	δ_ϵ	δ_{01H}	δ_{01R}	δ_{01L}	δ_{01}	δ_{12}	δ_{00}	δ_{11}
δ_ϵ	δ_ϵ	δ_{01H}	δ_{01R}	δ_{01L}	δ_{01}	δ_{12}	δ_{00}	δ_{11}
δ_{01H}	δ_{01H}	δ_{01H}	$\times$	δ_{01L}	$\delta_{01}, \delta_{11}, \delta_{12}$	δ_{12}	$\times$	δ_{12}
δ_{01R}	δ_{01R}	δ_{01R}	$\times$	δ_{00}	δ_{01R}	δ_{01R}	$\times$	δ_{01R}
δ_{01L}	δ_{01L}	$\times$	$\delta_{01}, \delta_{11}, \delta_{12}$	$\times$	$\times$	$\times$	δ_{01L}	$\times$
δ_{01}	δ_{01}	$\delta_{01}, \delta_{11}, \delta_{12}$	$\times$	δ_{01L}	δ_{01}	$\delta_{01}, \delta_{11}, \delta_{12}$	$\times$	δ_{01}
δ_{12}	δ_{12}	δ_{12}	$\times$	δ_{01L}	$\delta_{01}, \delta_{11}, \delta_{12}$	δ_{12}	$\times$	δ_{12}
δ_{00}	δ_{00}	$\times$	δ_{01R}	$\times$	$\times$	$\times$	δ_{00}	$\times$
δ_{11}	δ_{11}	δ_{12}	$\times$	δ_{01L}	δ_{01}	δ_{12}	$\times$	δ_{11}

Fig. 3. Table for all possibilities of α, β, γ for $\frown$ operator: The entry of row α and column β gives the corresponding γ('s)

$$\mathcal{L}_\gamma(D_1 \frown D_2) = \bigcup_{\alpha \text{ and } \beta \text{ correspond to } \gamma \text{ in Fig. 3}} \mathcal{L}_\alpha(D_1) \# \mathcal{L}_\beta(D_2)$$

where

$$\mathcal{L}_\alpha(D_1)\#\mathcal{L}_\beta(D_2) = \begin{cases} \left\{ \begin{array}{l} xay \mid xa \in \mathcal{L}_\alpha(D_1) \\ \qquad ay \in \mathcal{L}_\beta(D_2), a \in \Sigma \end{array} \right\} \\ \quad \text{if } \alpha \in \{\delta_{01}, \delta_{12}, \delta_{11}, \delta_{01R}, \delta_{01H}\} \\ \quad \text{and } \beta \in \{\delta_{01}, \delta_{12}, \delta_{11}, \delta_{01L}, \delta_{01H}\} \\ \{xy \mid x \in \mathcal{L}_\alpha(D_1), y \in \mathcal{L}_\beta(D_2)\} \\ \quad \text{if } \alpha \in \{\delta_{00}, \delta_{01L}, \delta_\epsilon\} \text{ and } \beta \in \\ \quad \{\delta_{00}, \delta_{01R}, \delta_\epsilon\} \end{cases}$$

3.3 Regularity of the languages

Regular languages are closed under union, intersection, concatenation, Kleene closure and complementation [4]. Further it is also known that: if L_1 and L_2 are regular languages then the *quotient language*

$$L_1/L_2 \stackrel{\text{def}}{=} \{x \mid \exists y \in L_2 \text{ such that } x.y \in L_1\} \text{ is regular.}$$

From these properties, it could be easily verified that, for all $\delta \in \Sigma_\delta$, $\mathcal{L}_\delta(D)$ is regular [8].

Lemma 4. *Let Θ be a synchronous interpretation of formula $D \in DC^s$. Then D is satisfiable in interval $[b, e]$ of type δ, iff $word(\Theta, [b, e])$ belongs to $\mathcal{L}_\delta(D)$.*

Proof: By induction over the structure of D it can be shown [8] that for all Θ, for all $[b, e]$ of type δ

$$\Theta, [b, e] \models D \text{ iff } word(\Theta, [b, e]) \in \mathcal{L}_\delta(D). \qquad \square$$

As a consequence of this lemma, we have the following theorem.

Theorem 5. *The validity of a formula $D \in DC^s$ under synchronous interpretation is decidable.* $\square$

3.4 Decidability of DC^N

DC^N is the set of DC formulas, which contains, in addition to the formulas of DC^s, the formulas $\int P < k$ and $\int P > k$, where $k \in \mathcal{N}$. We will show that the validity of a formula in this set is decidable under synchronous interpretation.

The definitions of Σ and $DNF(P)$ remain same as above. We will now define Σ_1 as:
$\Sigma_1 = \Sigma \backslash DNF(P)$. So, when $a \in \Sigma_1$ holds in an interval, it is obvious that P will not hold in that interval. We will now give the definition of the languages that we will generate for the formulas we have added to DC^s.

$\mathcal{L}_i(\int P < 1)$, $i \in \Sigma_\delta$ are defined as:

$$
\begin{aligned}
\mathcal{L}_{\delta_\epsilon}(\textstyle\int P < 1) &= \{\epsilon\} \\
\mathcal{L}_{\delta_{01H}}(\textstyle\int P < 1) &= \Sigma \\
\mathcal{L}_{\delta_{01}}(\textstyle\int P < 1) &= \left\{ a_1.a_2 \ldots a_{n-1}.a_n \;\middle|\; \begin{array}{l} n \geq 2 \text{ and } a_1, a_n \in \Sigma \text{ and} \\ a_2, \ldots, a_{n-1} \in \Sigma_1 \end{array} \right\} \\
\mathcal{L}_{\delta_{11}}(\textstyle\int P < 1) &= \left\{ a_1.a_2 \ldots a_{n-1}.a_n \;\middle|\; \begin{array}{l} n \geq 2 \text{ and either } (a_1 \in \Sigma \\ \text{and } a_2, \ldots, a_n \in \Sigma_1) \text{ or } (a_n \in \Sigma \\ \text{and } a_1, a_2, \ldots, a_n \in \Sigma_1) \end{array} \right\} \\
\mathcal{L}_{\delta_{12}}(\textstyle\int P < 1) &= \mathcal{L}_{\delta_{11}}(\textstyle\int P < 1) \\
\mathcal{L}_{\delta_{00}}(\textstyle\int P < 1) &= \{a_1.a_2 \ldots a_{n-1}.a_n \mid n \geq 1 \text{ and } a_1, a_2, \ldots, a_n \in \Sigma_1\} \\
\mathcal{L}_{\delta_{01R}}(\textstyle\int P < 1) &= \left\{ a_1.a_2 \ldots a_{n-1}.a_n \;\middle|\; \begin{array}{l} n \geq 1 \text{ and } a_n \in \Sigma \\ \text{and } a_1, a_2, \ldots, a_{n-1} \in \Sigma_1 \end{array} \right\} \\
\mathcal{L}_{\delta_{01L}}(\textstyle\int P < 1) &= \left\{ a_1.a_2 \ldots a_{n-1}.a_n \;\middle|\; \begin{array}{l} n \geq 1 \text{ and } a_1 \in \Sigma \\ \text{and } a_2, \ldots, a_n \in \Sigma_1 \end{array} \right\}
\end{aligned}
$$

$\mathcal{L}_i(\int P > 0)$ can be defined on similar lines.

Based on a case analysis when the atomic formulas $\int P < 1$ and $\int P > 0$ hold, it can be easily verified that Lemma 3 holds when the structural induction is extended over to such formulas. Moreover, with the above translation schema, it is also easy to verify that lemma 4 holds for cases when D is of the forms $\int P < 1$ and $\int P > 0$. Further, formulas $\int P < k$ and $\int P > k$, when $k \in \mathcal{N}$, could be generated from the atomic formulas as follows:

$$
\begin{aligned}
\textstyle\int P < k+1 &\equiv \textstyle\int P < k^\frown \int P < 1 \\
\textstyle\int P > k+1 &\equiv \textstyle\int P > k^\frown \int P > 1 \\
\textstyle\int P > 1 &\equiv \neg(\textstyle\int P < 1)^\frown \int P > 0
\end{aligned}
$$

So we have the following theorem.

Theorem 6. *The validity of a formula in DC^N, where $k \in \mathcal{N}$, under synchronous interpretation is decidable.* □

3.5 Complexity of the Decision Procedure

The construction procedure of a deterministic equivalent of a non-deterministic finite automaton (NFA), is of exponential complexity. The approach to check the validity of a DC formula D, needs to construct an automaton corresponding to the language of formula $\neg D$. For constructing an automaton for the complementation of a language, we need to transform the NFA of language into its deterministic equivalence. If a formula is of the form $((D_1 \rightarrow D_2) \rightarrow D_3) \rightarrow D_4$ then each negation will give rise to one degree rise in exponentiation in complexity. Skakkebaek and Sestoft, in their decision procedure for subset $\{\lceil P \rceil, \ell = k\}$ under discrete time DC, have shown that the decision problem is non-elementary [10]. What we can claim is that our decision procedure is of similar complexity as the procedure by Skakkebaek and Sestoft. But they claim that the worst case, i.e. the formula of the above type, occurs too rarely in practice. In many cases, their decision procedure has run with reasonable time; for instance, the validity of Fischer's Mutual Exclusion Algorithm has been checked within 12 minutes,

and the formula consisted of 3775 characters. Note that, Fischer's algorithm is specified by the subset $\{\llbracket P \rrbracket, \ell = k\}$ under discrete time DC, and hence $\mathrm{DC}^{\mathcal{Q}}$ can also specify the same.

4 Importance of the subset $\mathrm{DC}^{\mathcal{Q}}$

In this section, we illustrate two important systems to show that the subset $\mathrm{DC}^{\mathcal{Q}}$ is strong enough to to specify and verify many real-time systems.

4.1 The Mine Pump Controller

In a mine, water gets accumulated in a mine shaft which is pumped out of the mine. The water level inside the mine shaft is measured using H_2O sensor. The pump controller is supposed to keep the water level in the shaft below a critical level, called the DangerH_2O. If the water in the shaft exceeds this level a boolean state variable DH_2O is set to 1. For implementation purposes, a water level called the HighH_2O is slected which is slightly lower than the level DangerH_2O. Boolean state variable HH_2O is set to 1 when water level exceeds HighH_2O. When HH_2O flag remains set for δ amount of time, the monitor sets the boolean state variable H_2O-flag. Figure 4 shows the block diagram of a mine pump controller.

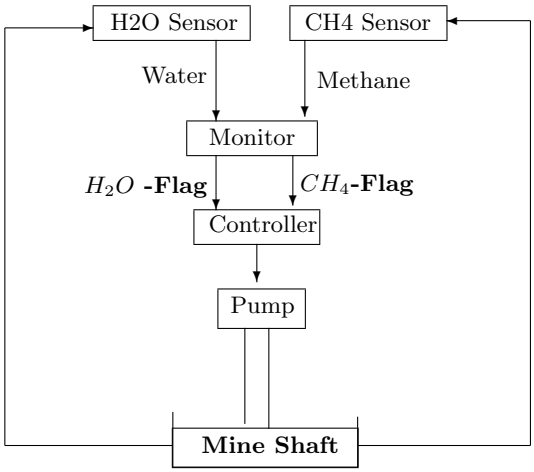

Fig. 4. Mine Pump Controller

Similarly the methane level in the mine shaft is measured by CH_4-sensor. Boolean variable DCH_4 is set to one if methane level in the shaft crosses a critical level, called the DangerCH_4. Like the previous case, in order to give monitor

the time to react, a methane level, called the HighCH_4, is selected which is slightly lower than DangerCH_4. A flag HCH_4 is set when methane level crosses HighCH_4 level. The monitor sets boolean variable CH_4-flag if the flag HCH_4 remains set for δ amount of time.

The pump controller sets ALARM flag when either HH_2O or HCH_4 remains set for δ amount oftime. The pump controller takes the values of HH_2O and HCH_4 to decide when to turn the pump on or off.

Specification of the requirement in DC The requirement of the mine pump system is that, in any observation interval of 1000 time units or less, the amount of time for which DH_2O flag remains set should not exceed 1 time unit. In terms of DC the above requirement is specified as:

$$\mathbf{Req} = \Box\ (l \leq 1000 \rightarrow \int DH_2O \leq 1)$$

Specification of the monitor and the design The actions of the monitor are implemented as follows.

1. H_2O-flag is set when HH_2O flag remains set for δ amount of time. H_2O-flag is also cleared when HH_2O flag does not remain set for δ amount of time. Similarly, the CH_4-flag is set and cleared.
2. The ALARM flag is set when either the water lever or the methane level remains dangerous for δ amount of time. The ALARM flag is cleared when none of them is dangerous.

The details of specifying the monitor have been worked out in [5]. We only here mention that the monitor can be specified by the subset DC^Q with the value of δ as a constant.

In any interval of at most 1000 time units, the HH_2O water level must always be reduced within w time units. And the duration of a *Failure* can be at most 1 time unit, where

$$\mathit{Failure} \stackrel{\text{def}}{=} \lceil HH_2O \rceil \wedge (\ell > w).$$

Then DES_1, the design decision 1 is:

$$\begin{aligned} DES_1 \stackrel{\text{def}}{=} & \Box(Failure \rightarrow l \leq 1) \wedge \\ & \Box((Failure^\frown \Diamond \neg Failure^\frown Failure) \rightarrow \ell \geq 1001) \end{aligned}$$

The second design decision can be specified as:

$$\begin{aligned} \mathrm{DES}_2 &\stackrel{\text{def}}{=} \mathrm{StartPump} \wedge \mathrm{StopPump}, \qquad \text{where,} \\ \mathrm{StartPump} &\stackrel{\text{def}}{=} \Box\neg((\lceil SafePump \rceil \wedge (l = \delta))^\frown \lceil \neg PumpON \rceil) \\ \mathrm{StopPump} &\stackrel{\text{def}}{=} \Box\neg((\lceil \neg SafePump \rceil \wedge (l = \delta))^\frown \lceil PumpON \rceil) \\ \mathrm{SafePump} &\stackrel{\text{def}}{=} HH_2O \wedge \neg HCH_4 \end{aligned}$$

Proof of Correctness The proof that the design decisions meet the requirement, could be stated through the following two theorems [5].

$$\text{Theorem 1: } AS_1 \wedge AS_2 \wedge DES_1 \rightarrow \text{Req}$$
$$\text{Theorem 2: } AS_3 \wedge AS_4 \wedge AS_5 \wedge Monitor \wedge DES_2 \rightarrow DES_1$$

Here, $AS_1, \ldots AS_5$ are the specifications of the assumptions about the environment under which the system will operate. The coding of such assumptions have been done in [5]. It is to be noted that the coding of assumptions use some constants like the parameters of the pump, the minimum time for which methane-level remains low, and the minimum time required for the water level to reach from HH_2O to reach DH_2O. So all such constants can be assumed to be rational numbers. Since they are either dependent on natural phenomena and the design of the pump, we cannot take them to be natural numbers. So we cannot specify the above theorems by using the subset DC^N. But it can be seen that such theorems could be specified by the subset DC^Q. So our decision procedure for subset DC^Q can verify the validity of the above two theorems.

4.2 Gas Burner System

Computer controlled gas burner system is a widely quoted example, whose requirements and implementations have been specified and verified by DC; though not mechanically [7]. It is a safety critical system in which an accident can occur if an excessive amount of unburned gas is leaked to the environment. Before the flame burns, some amount of gas will anyway leak. When burning flame is blown out, some gas can leak before the failure can be detected. The gas burner is controlled by a thermostat and gas is ignited by ignition transformer.

In [7], the requirements and the design decisions of the gas burner system have been specified by using the subset DC^Q. The refinement of the design decisions there have been proved manually by using the axioms and inference rules of DC. Now that DC^Q is decidable under synchronous interpretaion, our decision procedure could be used to prove the refinement. Since the specifications of the system as in [7] require rational numbers in the form of constants, DC^N will not serve the purpose.

5 Conclusion

The (i) synchronous assumption (ii) k as a rational number in the subset DC^Q have enough practical significance. The significance of (i) lies in the fact that at implementation level, the synchronous assumption holds. Though the set of real numbers is taken as the time domain, hardly anybody ever uses irrational numbers in a formula as a specification. That is why (ii) has much practical significance. So, in conclusion, the decidability result of DC^Q under synchronous assumption is a significant step towards automatic verification of the properties of real-time systems.

Acknowledgements

Prof. Zhou Chaochen has been a constant source of guidance throughout this research work. Martin Fränzle has gone through an earlier version of this paper and offered many valuable suggestions. Swarup Mohalik and Xu Qiwen have checked most of the proofs. Thanks to all of them.

References

1. Dutertre B., Complete Proof systems for First Order Interval Logic, Tenth Anual IEEE Symposium on *Logic in Computer Science*, pp. 36-43, IEEE Press, 1995.
2. Fränzle M., *Manuscript on Duration Calculus*, Technical Report, University of Keil, 1996.
3. Hansen M.R. & Zhou Chaochen, Duration Calculus: Logical Foundations, To appear in *Formal Aspects of Computing*, Springer Verlag.
4. Hopcroft J.E. & Ullman J.D., *Introduction to Automata Theory, Languages and Computation*, Addition-Wesley, 1979.
5. Joseph M., Real-Time Systems:*Specification, Verification and Analysis*, Printice Hall Intl. Series in Computer Science, 1996.
6. Ravn A.P. & Rischel H., Requirements Capture of Embedded Real-time Systems, Proc. of IMACS-MCT Symp. on Modelling and Control of Technical Systems, France, 1991.
7. Ravn A.P., Rischel H. & Hansen K.M., Specifying and Verifying Requirements of Real-Time Systems, IEEE Tr. on Software Engineering, Vol 19(1), January 1993.
8. Satpathy M., Hung D.V. & Pandya P.K., Some Results on the Decidability of Duration Calculus under Synchronous Interpretation, TR No. 86, UNU/IIST, Macau, 1996.
9. Skakkebaek J.U., Ravn A.P., Rischel H. & Zhao Chao Chen, Specification of Embedded real-time systems, Proc. of Euromicro Workshop on real-time systems, IEEE Computer Society Press, 1992.
10. Skakkebaek J.U. & Sestoft P, Checking Validity of Duration Calculus Formulas, Report ID/DTH/ JUS 3/1, Technical University of Denmark, Denmark, 1994.
11. Yuhua Z. & Zhou Chaochen, A Formal Proof of the Deadline Driven Scheduler, Third International Symp. on *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Lubeck (Germany), LNCS No. 863, Springer Verlag, 1994, pp. 756-775.
12. Zhou Chaochen, Hoare C.A.R. & Ravn A.P., A calculus of durations, *Information Processing Letters*, Vol 40(5), pp. 269-276, 1991.
13. Zhou Chaochen, Hansen M.R. & Sestoft P, Decidability and Undecidability Results in Duration Calculus, Proc. of the 10th Annual Symposium on *Theoretical Aspects of Computer Science (STACS 93)*, LNCS No. 665, Springer Verlag, 1993.