Digitizing Interval Duration Logic

Gaurav Chakravorty[1]* and Paritosh K. Pandya[2]

[1] Indian Institute of Technology, Kanpur, India
gchak@cse.iitk.ac.in
[2] Tata Institute of Fundamental Research
Colaba, Mumbai *400005*, India
pandya@tcs.tifr.res.in

Abstract. In this paper, we study the verification of dense time properties by discrete time analysis. Interval Duration Logic, (IDL), is a highly expressive dense time logic for specifying properties of real-time systems. Validity checking of IDL formulae is in general undecidable. A corresponding discrete-time logic QDDC has decidable validity.
In this paper, we consider a reduction of IDL validity question to QDDC validity using notions of digitization. A new notion of Strong Closure under Inverse Digitization, SCID, is proposed. For all SCID formulae, the dense and the discrete-time validity coincide. Moreover, SCID has good algebraic properties which can be used to conveniently prove that many IDL formulae are SCID. We also give some approximation techniques to strengthen/weaken formulae to SCID form. For SCID formulae, the validity of dense-time IDL formulae can be checked using the validity checker for discrete-time logic QDDC.

1 Introduction

Duration Calculus (DC) is a highly expressive logic for specifying properties of real-time systems [10]. Interval Duration Logic (IDL) [8] is a variant of Duration Calculus where formulae are interpreted over *timed state sequences*. Timed state sequences [1] are a well studied model of real-time behaviour with a well-developed automata theory and tools for the analysis of such automata. In practical applications, IDL inherits much of the expressive ability of the original DC.

Model/validity checking of IDL is undecidable in general. Even for some restricted class of formulae [8], their verification requires analysis of hybrid automata, which is computationally expensive. By contrast, the Discrete-time Duration Calculus (QDDC) is decidable [7] using well developed automata theoretic techniques. A tool DCVALID permits model/validity checking of QDDC formulae for many significant examples [7].

In this paper, we investigate the reduction of IDL validity question to QDDC validity question so that QDDC tools can be used to analyse IDL formulae.

* This work was partly carried out under the VSRP program of TIFR, Mumbai.

© Springer-Verlag Berlin Heidelberg 2003

Thus, we investigate verification of dense-time properties by discrete-time analysis. There is experimental evidence that with existing techniques, automatic verification of discrete-timed systems can be significantly easier than the verification of similar dense-time systems [2]. Hence, we believe that our approach of reduction from IDL to QDDC is practically useful.

IDL models are precisely timed state sequences where states are labelled with real-valued time stamps. Denote the set of such behaviours by TSS_R. We can also interpret IDL over timed state sequences where all time stamps have only integer values. Call the set of such behaviours as TSS_Z, and IDL restricted to such behaviours as ZIDL. An IDL behaviour can be digitized to a *set* of ZIDL behaviours by approximating its time stamps to nearby integer values. We follow the notion of digitization due to Henzinger *et al* and make use of their digitization theorem which gives sufficient conditions for reducing dense-time model checking to discrete-time model checking [4].

The reduction from IDL to QDDC is carried out in two stages. In the first stage, we give a reduction from IDL to ZIDL validity which is sound only for formulae which are "Closed under Inverse Digitization" (CID), a notion proposed by Henzinger, Manna and Pnueli [4]. Unfortunately, it is quite hard to establish whether IDL formulae have CID property. Towards this we propose a new notion of "Strong Closure under Inverse Digitization" (SCID). Fortunately, SCID is preserved by most IDL operators and we are able to give a *structural characterization* of a large class of IDL formulae which are SCID. For formulae which are not SCID, we give approximations to stronger and weaker formulae which are SCID. Finally, SCID implies CID and hence for such formulae reduction from IDL to ZIDL is sound. Our logic IDL includes a powerful $\frown$ (chop) operator and a notion of $\int P$, the accumulated amout of time for which proposition P holds in a time interval. Digitization of such properties is one of the contributions of this paper.

ZIDL is the logic of weakly monotonic integer-timed state sequences, where as QDDC is a logic of untimed state sequences. In our next reduction, we translate an arbitrary ZIDL formula D to an "equivalent" QDDC formula $\beta(D)$. Here, "equivalence" means preserving models under a suitable isomorphism.

The rest of the paper is organised as follows. Logic IDL is introduced in Section 2. Basic notions of digitization and closure under digitization are presented in Section 3. Digitization of IDL formulae to ZIDL formulae is presented in Section 4. Section 5 presents the reduction from ZIDL to QDDC. We illustrate our approach by a small example in Section 6, where the validity of a (dense-time) IDL formula is established using the QDDC validity checker DCVALID. The paper ends with a discussion of related work.

2 Interval Duration Logic

Let $Pvar$ be the set of propositional variables (called state variables in DC). The set of states is $\Sigma = 2^{Pvar}$ consisting of the set of subsets of $Pvar$. Let $\Re^0$ be the set of non-negative real numbers.

Definition 1. *A* **timed state sequence** *over Pvar is a pair* $\theta = (\sigma, \tau)$ *where* $\sigma = s_0 s_1 \ldots . s_{n-1}$ *with* $s_i \in 2^{Pvar}$ *is a finite non-empty sequence of states, and* $\tau = t_0\ t_1 \ldots\ t_{n-1}$ *is a finite sequence of time stamps such that* $t_i \in \Re^0$ *with* $t_0 = 0$ *and* τ *is non-decreasing. Let* $dom(\theta) = \{0, \ldots, n-1\}$ *be the set of positions within the sequence* θ*. Also, let the length of* θ *be* $\#\theta = n$*.*

Timed state sequence gives a sampled view of timed behaviour. Note that the time is *weakly monotonic* with several state changes occurring at same time [6].

The set of timed state sequences is denoted by TSS_R. We shall use TSS_Z to denote the subset of TSS_R where all time stamps have non-negative integer values.

Let $Prop$ be the set of propositions (boolean formulae) over $Pvar$ with 0 denoting false and 1 denoting true. The truth of proposition P can be evaluated at any position i in $dom(\theta)$. This is denoted by $\theta, i \models P$. We omit this obvious definition.

Logic IDL is a form of interval temporal logic. The set of *intervals* within a timed state sequence θ can be defined as follows, where $[b, e]$ denotes a pair of positions. Each interval uniquely identifies a subsequence of θ.

$$Intv(\theta) \quad = \quad \{[b, e] \ \in \ dom(\theta)^2 \ \mid \ b \leq e\}$$

Syntax of Interval Duration Logic Let p, q range over propositional variables from $Pvar$, let P, Q range over propositions and D_1, D_2 range over IDL formulae. Let c range over non-negative integer constants.

$$\lceil P \rceil^0 \mid \lceil P \rceil \mid D_1 \frown D_2 \mid D_1 \wedge D_2 \mid \neg D \mid \eta\ op\ c \mid \Sigma P\ op\ c \mid \ell\ op\ c \mid \int P\ op\ c$$

where $op \ \in \ < \ \mid \ > \ \mid \ = \ \mid \ \leq \ \mid \ \geq$. Let IDL_l denote the subset of IDL formulae in which duration formulae of the form $(\int P\ op\ c)$ do not occur.

Semantics of IDL Let $\theta, [b, e] \models D$ denote that formula D evaluates to true within a timed state sequence θ and interval $[b, e]$, as defined inductively below.

$\theta, [b, e] \models \lceil P \rceil^0$ **iff** $b = e$ and $\theta, b \models P$
$\theta, [b, e] \models \lceil P \rceil$ **iff** $b < e$ and for all $m :\ b < m < e.\ \ \theta, m \models P$
$\theta, [b, e] \models D_1 \frown D_2$ **iff** for some $m : b \leq m \leq e$.
$\theta, [b, m] \models D_1$ and $\theta, [m, e] \models D_2$
$\theta, [b, e] \models D_1 \wedge D_2$ **iff** $\theta, [b, e] \models D_1$ and $\theta, [b, e] \models D_2$
$\theta, [b, e] \models \neg D$ **iff** $\theta, [b, e] \not\models D$

Now we consider the semantics of measurement formulae. Logic IDL has four different types of measurement terms: $\eta \mid \Sigma P \mid \ell \mid \int P$.
These represent some specific quantitative measurements over the behaviour in a given interval. We shall denote the value of a measurement term t in a timed state sequence θ and an interval $[b, e]$ by $eval(t)(\theta, [b, e])$, as defined below. Step length η gives the number of steps within a given interval, whereas time length ℓ gives the amount of real-time spanned by a given interval. Step count ΣP counts the number of states for which P holds in the (left-closed-right-open) interval. Duration $\int P$ gives amount of real-time for which proposition P holds in the

given interval. Terms η and ΣP are called *discrete measurements*, whereas terms ℓ and $\int P$ are called *dense measurements.* A measurement formula compares a measurement term with an *integer* constant.

$$\begin{array}{lllll} eval(\eta)(\theta,[b,e]) & = & e-b, & eval(\ell)(\theta,[b,e]) & = & t_e - t_b \end{array}$$

$$eval(\Sigma P)(\theta,[b,e]) \quad = \quad \sum_{i=b}^{e-1} \begin{pmatrix} 1 & \text{if} \quad \theta, i \models P \\ 0 & \text{otherwise} \end{pmatrix}$$

$$eval(\textstyle\int P)(\theta,[b,e]) \quad = \quad \sum_{i=b}^{e-1} \begin{pmatrix} t_{i+1}-t_i & \text{if} \quad \theta, i \models P \\ 0 & \text{otherwise} \end{pmatrix}$$

$$\theta,[b,e] \models t \; op \; c \quad \textbf{iff} \quad eval(t)(\theta,[b,e]) \; op \; c$$

Finally, a formula D holds for a timed state sequence θ if it holds for the full interval spanning the whole sequence.

$$\theta \models D \quad \textbf{iff} \quad \theta,[0,\#\theta-1] \models D$$
$$\models D \quad \textbf{iff} \quad \theta \models D \quad \text{for all } \theta$$

Derived Operators

- $[\lceil P \rceil \stackrel{\text{def}}{=} (\lceil P \rceil^0 \frown \lceil P \rceil)$ and $[\lceil P \rceil] \stackrel{\text{def}}{=} ([\lceil P \rceil \frown \lceil P \rceil^0)$. Formula $[\lceil P \rceil]$ states that proposition P holds invariantly over the interval including its end points.
- $\Diamond D \stackrel{\text{def}}{=} true \frown D \frown true$ holds provided D holds for some subinterval.
- $\Box D \stackrel{\text{def}}{=} \neg\Diamond\neg D$ holds provided D holds for all subintervals.

Example 1. Formula $\Box([\lceil P \rceil \Rightarrow \ell \leq 10)$ states that in any interval, if P is invariantly true then the time length of the interval is at most 10. That is, P cannot last for more than 10 time units at a stretch.

$$Follows(P,Q,d) \quad \stackrel{\text{def}}{=} \quad \neg\Diamond(([\lceil P \rceil \wedge \ell \geq d) \frown \lceil \neg Q \rceil^0).$$

Formula $Follows(P,Q,d)$ states that if P has held continuously d or more time units in past, then Q must hold. Formula $FollowsWk$ requires Q to hold only after P has held for strictly more than d time units.

$$FollowsWk(P,Q,d) \quad \stackrel{\text{def}}{=} \quad \neg\Diamond(([\lceil P \rceil \wedge \ell > d) \frown \lceil \neg Q \rceil^0). \qquad \Box$$

Quantified Discrete-time Duration Calculus (QDDC) is the the subset of *IDL* where dense-time measurement constructs of the form $\ell \; op \; c$ or $\int P \; op \; c$ are not used. Note that discrete time measurement constructs $\eta \; op \; c$ or $\Sigma P \; op \; c$ can still be used. For *QDDC* formulae, the time stamps τ in behaviour $\theta = (\sigma, \tau)$ do not play any role. Hence, we can also define the semantics of *QDDC* purely using state sequences, i.e. $\sigma \models D$ (see [7]).

Decidability and Model Checking Although, validity of full *IDL* is undecidable [8], the validity of QDDC formulae is decidable. A tool, called DCVALID, based on an automata-theoretic decision procedure for QDDC has been implemented, and found to be effective on many significant examples [7]. In the rest of the paper, we consider a reduction of *IDL* model/validity checking problem to *QDDC* model/validity checking problem. This provides a novel and, in our opinion, a practically useful technique for reasoning about IDL properties.

3 Digitization

In this section we provide a brief overview of the pioneering work of Henzinger, Manna and Pnueli [4] characterizing the set of systems and properties for which the real-time verification problem is equivalent to integer-time verification.

Notation For real numbers a, b with $a \leq b$, let $[a : b)$ denote the left closed right open interval. Similarly $(a : b)$, $(a : b]$ and $[a : b]$. Let $frac(a) \stackrel{\text{def}}{=} a - \lfloor a \rfloor$ denote the fractional part of a real number a.

Definition 2 (Digitization). *Let $x \in \Re^0$ and $\theta = (\sigma, \tau) \in TSS_R$. Let $\epsilon \in [0 : 1)$. Then, ϵ-digitization of θ, denoted by $[\theta]_\epsilon$, is defined as follow.*

- $x \downarrow \epsilon \stackrel{\text{def}}{=} \left\{ \begin{array}{c} \lfloor x \rfloor \quad if \;\; frac(x) \leq \epsilon \\ else \;\; \lceil x \rceil \end{array} \right\}$
- $[\theta]_\epsilon \stackrel{\text{def}}{=} (\sigma, \tau') \;\; s.t. \;\; \tau'(i) = \tau(i) \downarrow \epsilon$

Example 2. Let $\theta = (\sigma_0, 0.0) \longrightarrow (\sigma_1, 1.5) \longrightarrow (\sigma_2, 4.35) \longrightarrow (\sigma_3, 5.0)$.
Then, $[\theta]_{0.0} = (\sigma_0, 0) \longrightarrow (\sigma_1, 2) \longrightarrow (\sigma_2, 5) \longrightarrow (\sigma_3, 5)$
and $[\theta]_{0.4} = (\sigma_0, 0) \longrightarrow (\sigma_1, 2) \longrightarrow (\sigma_2, 4) \longrightarrow (\sigma_3, 5)$.

Definition 3. *Let $\theta \in TSS_R$ and $\Pi \subseteq TSS_R$. Then,*
$[\theta] \stackrel{\text{def}}{=} \{[\theta]_\epsilon \;\mid\; \epsilon \in [0, 1)\}$
$[\Pi] \stackrel{\text{def}}{=} \{[\theta]_\epsilon \;\mid\; \epsilon \in [0 : 1), \;\; \theta \in \Pi\}$*. Note that $[\Pi] \subseteq TSS_Z$.*
$Z(\Pi) \stackrel{\text{def}}{=} \Pi \cap TSS_Z$*, the set of integer valued traces of Π.*

Set $[\Pi]$ gives the set of digitized approximations of the behaviours in Π where as $Z(\Pi)$ gives the integer time fragment of Π. Closures under digitization and inverse digitization, defined below, are used to give a digitization theorem which reduces dense-time model checking to discrete time model checking.

Definition 4 (Closure under digitization (CD)).
$CD(\Pi) \stackrel{\text{def}}{=} \forall\theta \in TSS_R. \; (\theta \in \Pi \;\Rightarrow\; (\forall\epsilon \in [0, 1). \; [\theta]_\epsilon \in \Pi))$.

Proposition 1. $CD(\Pi)$ **iff** $[\Pi] \subseteq \Pi$ **iff** $Z(\Pi) = [\Pi]$.

Definition 5 (Closure under inverse digitization (CID)).
$CID(\Pi) \stackrel{\text{def}}{=} \forall\theta \in TSS_R. \; ((\forall\epsilon \in [0, 1). \; [\theta]_\epsilon \in \Pi) \;\Rightarrow\; \theta \in \Pi)$

Theorem 1 (Digitization). *Let $\Psi, \Pi \subseteq TSS_R$.*
– If $CD(\Psi)$ and $CID(\Pi)$, then $\Psi \subseteq \Pi$ **iff** *$Z(\Psi) \subseteq Z(\Pi)$*
– If $CID(\Pi)$ then $\Pi = TSS_R$ **iff** *$Z(\Pi) = TSS_Z$*

Typically, the set Ψ in the above theorem denotes the set of behaviours of a system where as Π denotes the set of behaviours satisfying some desired property. Hence, the theorem gives sufficient conditions under which the real time verification problem $\Psi \subseteq \Pi$ can be reduced to the integer time verification problem $Z(\Psi) \subseteq Z(\Pi)$. *The key requirement for reducing dense-time validity to discrete-time validitiy is that properties should be closed under inverse digitization CID.*

4 Digitization of Interval Duration Logic Formulae

We consider the real-time properties specified in logic IDL. We must determine the subset of IDL properties that are closed under inverse digitization (CID).

4.1 Closure Properties in IDL

Notation Let $[[D]]_R \overset{\mathrm{def}}{=} \{\theta \mid \theta \models D\}$ denote the set of timed state sequences satisfying the IDL formula D, and let $[[D]]_Z \overset{\mathrm{def}}{=} [[D]]_R \cap TSS_Z$ denote the set of integer timed sequences satisfying D. Define $\models_R D \overset{\mathrm{def}}{=} [[D]]_R = TSS_R$ and $\models_Z D \overset{\mathrm{def}}{=} [[D]]_Z = TSS_Z$. Then, $CID([[D]]_R)$ states that the set of timed state sequences satisfying D is closed under inverse digitization. We shall abbreviate $CID([[D]]_R)$ by $CID(D)$. Similarly, also $CD(D)$ and $SCID(D)$ (defined below).

Proposition 2. *Algebraic properties of CD:*
(a) $CD(D_1) \land CD(D_2) \Rightarrow CD(D_1 \land D_2)$, *(b)* $CD(D_1) \land CD(D_2) \Rightarrow CD(D_1 \lor D_2)$, *(c)* $CD(D_1) \land CD(D_2) \Rightarrow CD(D_1 \frown D_2)$, *(d)* $CD(D) \Rightarrow CD(\Box D)$, *and*
(e) $CD(D) \Rightarrow CD(\Diamond D)$.

Proposition 3. *Algebraic properties of CID:*
(a) $CID(D_1) \land CID(D_2) \Rightarrow CID(D_1 \land D_2)$, *and (b)* $CID(D) \Rightarrow CID(\Box D)$.

Unfortunately, operators $\lor, \frown, \neg$ do not preserve the crucial CID property making it difficult to establish that a formula is CID. Below we introduce a stronger notion of closure, SCID, which has vastly superior preservation properties. Also, $SCID(D)$ implies $CID(D)$.

Definition 6 (Strong Closure under Inverse Digitization(SCID)). *For* $\Pi \subseteq TSS_R$, *let* $SCID(\Pi) \overset{\mathrm{def}}{=} \forall \theta \in TSS_R \; ((\exists \epsilon \in [0, 1). \; [\theta]_\epsilon \in \Pi) \;\Rightarrow\; \theta \in \Pi)$.

Proposition 4. *(a)* $SCID(D) \;\Leftrightarrow\; CD(\neg D)$, *and (b)* $SCID(D) \;\Rightarrow\; CID(D)$.

Proposition 5. *Algebraic properties of SCID:*
(a) $SCID(D_1) \land SCID(D_2) \Rightarrow SCID(D_1 \land D_2)$,
(b) $SCID(D_1) \land SCID(D_2) \Rightarrow SCID(D_1 \lor D_2)$,
(c) $SCID(D_1) \land SCID(D_2) \Rightarrow SCID(D_1 \frown D_2)$,
(d) $SCID(D) \Rightarrow SCID(\Box D)$ *and (e)* $SCID(D) \Rightarrow SCID(\Diamond D)$. □

Proposition 6. $SCID(D_1) \land CID(D_2) \Rightarrow CID(D_1 \lor D_2)$ □

Lemma 1. *Formulae of IDL which are free of dense measurements (i.e. QDDC formulae) are CD as well as SCID; hence CID.*

Proof. Let $\theta = (\sigma, \tau)$ and let $D \in QDDC$. Then, $[\theta]_\epsilon = (\sigma, \tau')$. Note that the interpretation of D does not depend upon the time stamp sequence τ. Hence, $(\sigma, \tau), [b, e] \models D$ **iff** $(\sigma, \tau'), [b, e] \models D$. □

4.2 Digitization of Dense Measurements

We consider the effect of digitization on dense measurements ℓ and $\int P$. We first study some number theoretic properties of digitization.

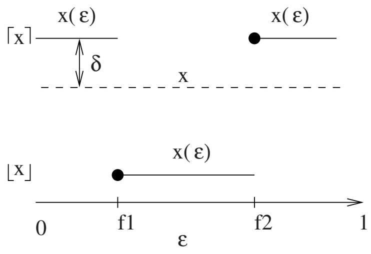
Lemma 2. *Let $c_1 \geq c_2$. Let $f_1 = frac(c_1)$ and $f_2 = frac(c_2)$ be the fractional parts of c_1 and c_2. Let $\delta = f_1 - f_2$. Let $x = c_1 - c_2$ and $x(\epsilon) = c_1 \downarrow \epsilon - c_2 \downarrow \epsilon$. We characterize the difference $x(\epsilon) - x$ for $0 \leq \epsilon < 1$ below, and plot it alongside.*

Let $f_1 \leq f_2$. Hence $\delta \in (-1 : 0]$. Then,

$$\forall\epsilon \in [0 : f_1) \left\{ \begin{array}{c} c_1 \downarrow \epsilon = \lceil c_1 \rceil, c_2 \downarrow \epsilon = \lceil c_2 \rceil \\ x(\epsilon) - x = -\delta \end{array} \right\}$$

$$\forall\epsilon \in [f_1 : f_2) \left\{ \begin{array}{c} c_1 \downarrow \epsilon = \lfloor c_1 \rfloor, c_2 \downarrow \epsilon = \lceil c_2 \rceil \\ x(\epsilon) - x = -(\delta + 1) \end{array} \right\}$$

$$\forall\epsilon \in [f_2 : 1) \left\{ \begin{array}{c} c_1 \downarrow \epsilon = \lfloor c_1 \rfloor, c_2 \downarrow \epsilon = \lfloor c_2 \rfloor \\ x(\epsilon) - x = -\delta \end{array} \right\}$$

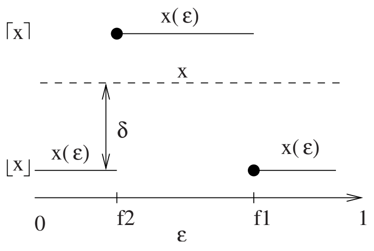


Let $f_1 > f_2$. Hence $\delta \in (0 : 1)$. Then,

$$\forall\epsilon \in [0 : f_2) \left\{ \begin{array}{c} c_1 \downarrow \epsilon = \lceil c_1 \rceil, c_2 \downarrow \epsilon = \lceil c_2 \rceil \\ x(\epsilon) - x = -\delta \end{array} \right\}$$

$$\forall\epsilon \in [f_2 : f_1) \left\{ \begin{array}{c} c_1 \downarrow \epsilon = \lceil c_1 \rceil, c_2 \downarrow \epsilon = \lfloor c_2 \rfloor \\ x(\epsilon) - x = -(\delta - 1) \end{array} \right\}$$

$$\forall\epsilon \in [f_1 : 1) \left\{ \begin{array}{c} c_1 \downarrow \epsilon = \lfloor c_1 \rfloor, c_2 \downarrow \epsilon = \lfloor c_2 \rfloor \\ x(\epsilon) - x = -\delta \end{array} \right\}$$



From this it also follows that,

$$\int_0^1 (x(\epsilon) - x).d\epsilon \quad = \quad 0.0 \tag{1}$$

As a consequence of above case analysis, we have the following three results.

Proposition 7. *Let $c_1 \geq c_2$ be non-negative reals, and c be non-negative integer. Then,*
(A) $(c_1 - c_2) > c \Rightarrow \exists\epsilon.\ \ (c_1 \downarrow \epsilon - c_2 \downarrow \epsilon) > c$
(B) $(c_1 - c_2) \geq c \Rightarrow \forall\epsilon.\ \ (c_1 \downarrow \epsilon - c_2 \downarrow \epsilon) \geq c$
(C) $(c_1 - c_2) \leq c \Rightarrow \forall\epsilon.\ \ (c_1 \downarrow \epsilon - c_2 \downarrow \epsilon) \leq c$
(D) $(c_1 - c_2) < c \Rightarrow \exists\epsilon.\ \ (c_1 \downarrow \epsilon - c_2 \downarrow \epsilon) < c$

Proof. The result can be easily seen by examination of Figures in Lemma 2. We omit a detailed algebraic proof. □

Theorem 2. *$CD(\ell \geq c)$ and $CD(\ell \leq c)$. Also $SCID(\ell > c)$ and $SCID(\ell < c)$.*

Proof. We prove that $CD(\ell \geq c)$. Proofs of the other parts are omitted. Let $\theta, [b, e] \models \ell \geq c$. This implies $t_e - t_b \geq c$.
By Proposition 7(B), $\forall\epsilon \in [0 : 1),\ (t_e \downarrow \epsilon - t_b \downarrow \epsilon) \geq c$.
Hence, $\forall\epsilon \in [0 : 1),\ [\theta]_\epsilon, [b, e] \models \ell \geq c$. □

Theorem 3. *$CID(\int P\ op\ c)$ where $op \in \{<, \leq, \geq, >\}$.*

Proof. By the semantics of $\int P$, we have

$$eval(\int P)([\theta]_\epsilon, [b,e]) \quad = \quad \sum_{i=b}^{e-1} \begin{pmatrix} t_{i+1} \downarrow \epsilon - t_i \downarrow \epsilon & \text{if} \quad \sigma, i \models P \\ 0 & \text{otherwise} \end{pmatrix}$$

By Equation 1, we have,

$$\int_0^1 \; ((t_{i+1} \downarrow \epsilon - t_i \downarrow \epsilon) - (t_{i+1} - t_i)).d\epsilon \quad = \quad 0.0.$$

Hence, $\left(\int_0^1 \; (eval(\int P)([\theta]_\epsilon, [b,e])) - eval(\int P)(\theta, [b,e])).d\epsilon\right) = 0.0$

Therefore, one of the following must hold

– $\{\forall\epsilon.\ eval(\int P)([\theta]_\epsilon, [b,e]) = eval(\int P)(\theta, [b,e])\}$, or

– $\{\ (\exists\epsilon.\ eval(\int P)([\theta]_\epsilon, [b,e]) > eval(\int P)(\theta, [b,e])) \quad \wedge$
$(\exists\epsilon.\ eval(\int P)([\theta]_\epsilon, [b,e]) < eval(\int P)(\theta, [b,e]))\ \}$

Hence, for $op \in \{<, \leq, \geq, >\}$ we have,

$$\{(\theta, [b,e]) \not\models \int P\ op\ c \quad \Rightarrow \quad \exists\epsilon.\ [\theta]_\epsilon, [b,e] \not\models \int P\ op\ c\}.$$

The result follows immediately from this. □

4.3 Proving IDL Formulae CID

The algebraic properties of the previous section can be used to infer that an IDL formula is $SCID$ or CID *purely from its syntactic structure*. Recall that IDL_ℓ is the IDL subset without $\int P$ terms. Formulae of form $(\ell\ op\ c)$ will be called *length constraints*.

Example 3. $CID(\Box(\ell \leq 60 \ \Rightarrow\ \int Leak \leq 5))$.
Proof. By Theorem 2, $CD(\ell \leq 60)$. Hence, by Proposition 4(a) , $SCID(\neg(\ell \leq 60))$. Also, by Theorem 3, $CID(\int Leak \leq 5)$. Hence using Propositions 6 and 3(b), we have $CID(\Box(\neg(\ell \leq 60)\ \vee\ \int Leak \leq 5))$ which gives the result. □

Theorem 4. *For $D \in IDL_l$, if the following conditions hold then $SCID(D)$.*

– *every length constraint occurring within the scope of even number negation has the form $\ell > c$ or $\ell < c$, and*
– *every length constraint occurring within the scope of odd number of negations has the form $\ell \leq c$ or $\ell \geq c$.* □

Using the Digitization Theorem 1, we get the following Theorem.

Theorem 5. *If $CID(D)$ then $\models_R D$* **iff** *$\models_Z D$.*

Digitization Approximation of IDL_l formulae Not all IDL_l formulae are SCID. We now define strengthening and weakening transformations ST and WT of IDL_l formulae which result in SCID formulae.

Definition 7. ST*(D)* $\overset{\text{def}}{=}$ *Substituting in D every atomic 'ℓ' constraint*
$(\ell \geq c)$*, under an even number of negations, to* $(\ell > c)$
$(\ell \leq c)$*, under an even number of negations, to* $(\ell < c)$
$(\ell > c)$*, under an odd number of negations, to* $(\ell \geq c)$
$(\ell < c)$*, under an odd number of negations, to* $(\ell \leq c)$

Definition 8. WT*(D)* $\overset{\text{def}}{=}$ *Substituting in D every atomic 'ℓ' constraint*
$(\ell \geq c)$ *under an even number of negations to* $(\ell > c - 1)$
$(\ell \leq c)$ *under an even number of negations to* $(\ell < c + 1)$
$(\ell > c)$ *under an odd number of negations to* $(\ell \geq c + 1)$
$(\ell < c)$ *under an odd number of negations to* $(\ell \leq c - 1)$

Example 4. Refer to the formulae of Example 1. Using Theorem 4, we can conclude that $SCID(Follows(P, Q, d))$ and $CD(FollowsWk(P, Q, d))$. Using the definitions of WT and ST we get that

$ST(FollowsWk(P, Q, d)) \;=\; Follows(P, Q, d)$, and
$WT(FollowsWk(P, Q, d)) \;=\; Follows(P, Q, d + 1)$. □

Theorem 6. *For every $D \in IDL_l$, we have*
1. $\models_R \; ST(D) \;\Rightarrow\; D$ *and* $\models_R \; D \;\Rightarrow\; WT(D)$.
2. $SCID(ST(D))$ *and* $SCID(WT(D))$. □

By combining the above with Theorem 5 we obtain a method of reducing IDL validity to ZIDL validity. This is outlined in the following theorem, which also suggests how to promote counter examples from discrete-time to dense-time. Validity of $ZIDL$ is decidable as shown in the next section.

Theorem 7. *For any formula $D \in IDL_l$,*
1. $\models_Z ST(D) \quad\Rightarrow\quad \models_R D$
2. $\theta \not\models_Z WT(D) \quad\Rightarrow \theta \not\models_R D$.
3. $\models_Z WT(D) \quad\Rightarrow\quad \models_R WT(D)$.

5 ZIDL to QDDC

We now consider a reduction from ZIDL to QDDC with the aim that we can utilize tools for QDDC to reason about ZIDL. Note that ZIDL is a logic of weakly monotonic integer-timed state sequences where as QDDC is a logic of untimed state sequences.

We first define an encoding α of a TSS_Z behaviour by means of a untimed sequence of states. This is depicted in Figure 5. The line with dark circles represents TSS_Z behaviour $\theta \;=\; (s_0, 0) \rightarrow (s_1, 3) \rightarrow (s_2, 3) \rightarrow (s_3, 3) \rightarrow (s_4, 5)$. The bottom line denotes QDDC behaviour $\alpha(\theta)$. A function *place* maps positions in θ to corresponding positions in $\alpha(\theta)$ and is denoted by dashed lines. A new boolean variable E marks exactly the positions of $\alpha(\theta)$ which are image of *place*. Note that θ is weakly monotonic having "micro steps" which do not change the time stamp. A new boolean variable M marks exactly the positions in $\alpha(\theta)$ where the next step is a micro step. If $Pvar$ is the set of original

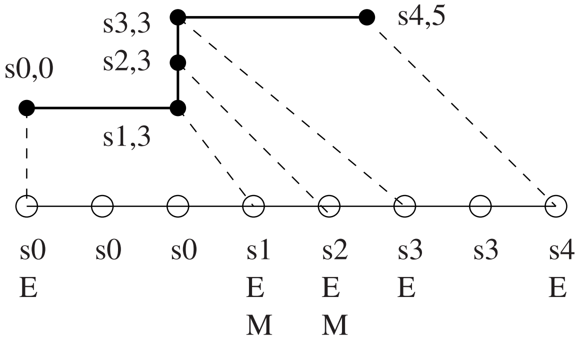


Fig. 1. Encoding of ZIDL Behaviours

propositional variables, then $ST_{E,M} = 2^{(Pvar \cup \{E,M\})}$ denotes the states assigning truth values to $Pvar \cup \{E, M\}$. We omit the formal definition of encoding $\alpha : TSS_Z \to (ST_{E,M})^+$ which can be found in the full paper.

Not all elements of $ST^+_{E,M}$ correspond to TSS_Z behaviours. We give a formula CONSIST specifying consistent behaviours. Every consistent QDDC behaviour *uniquely* denotes a ZIDL behaviour and vice versa.

$$\begin{array}{l} CONSIST(Pvar) \stackrel{\text{def}}{=} \\ \quad \{(\lceil E \rceil^0 \frown true \frown \lceil E \land \neg M \rceil^0) \ \bigwedge \ (\Box \lceil\lceil M \Rightarrow E \rceil\rceil) \ \bigwedge \\ \quad (\neg \Diamond (\lceil M \rceil^0 \frown (\eta = 1) \frown \lceil \neg E \rceil^0)) \ \bigwedge \\ \quad \forall p \in Pvar. \ (\Box (\lceil E \rceil^0 \frown (\eta = 1) \frown \lceil\lceil \neg E \rceil\rceil \Rightarrow (\lceil\lceil p \rceil\rceil \lor \lceil\lceil \neg p \rceil\rceil)))\} \end{array}$$

Proposition 8. *(1)* $\forall \theta \in TSS_Z. \ \ \alpha(\theta) \models CONSIST$.
(2) Map $\alpha : TSS_Z \to \{\sigma \in ST^+_{E,M} \ \mid \ \sigma \models CONSIST\}$ *is an isomorphism.* □

We now give the translation (β) of ZIDL formulae into QDDC formulae over $ST^+_{E,M}$, where β is overloaded to map ZIDL measurement terms to QDDC expressions also.

Definition 9.

$$\begin{array}{ll} \beta(\eta) = \sum \ E & \quad \beta(\sum P) = \sum \ (E \land P) \\ \beta(\ell) = \sum \ (\neg M) & \quad \beta(\int P) = \sum \ (\neg M \land P) \end{array}$$

Proposition 9. $eval(t)(\theta, [b, e]) \ = \ eval(\beta(t))(\alpha(\theta), [place(b), place(e)])$ □

Since $\alpha(\theta)$ has positions (intervals) which do not correspond to positions (intervals) of θ, we translate the formulae ensuring that all chopping points correspond to pre-existing positions in θ, i.e. points where E is true.

Definition 10.

$$\begin{array}{ll} \beta(\lceil P \rceil^0) = \lceil P \rceil^0 & \quad \beta(\lceil P \rceil) = \lceil P \rceil \\ \beta(t \ op \ c) = \beta(t) \ op \ c & \quad \beta(\neg D) = \neg \beta(D) \\ \beta(D_1 \frown D_2) = \beta(D_1) \frown \lceil E \rceil^0 \frown \beta(D_2) & \quad \beta(D_1 \land D_2) = \beta(D_1) \land \beta(D_2) \end{array}$$

Theorem 8. *Let $\theta \in TSS_Z$. Then,*
$\theta, [b,e] \models_Z D$ **iff** $\alpha(\theta), [place(b), place(e)] \models_{QDDC} \beta(D)$ □

Theorem 9. *(1)* $\models_Z D$ **iff** $\models_{QDDC} (CONSIST(Pvar) \Rightarrow \beta(D))$
(2) $\sigma \not\models_{QDDC} (CONSIST \Rightarrow \beta(D))$ *then* $\alpha^{-1}(\sigma) \not\models_Z D$. □

Since, validity of *QDDC* is decidable [7], we have the following corollary.

Corollary 1. *Validity of ZIDL formulae is decidable.* □

6 Verification by Digitization: An Example

It is our belief that techniques developed in this paper are of practical importance. These techniques allow dense-time properties to be checked by reducing them to discrete-time properties which can be efficiently analysed. We illustrate this approach by proving validity of a small IDL formula.

Recall the formulae $Follows(P,Q,d)$ and $FollowsWk(P,Q,d)$ given in Examples 1 and 4. Let, $Within(P,Q,d) \stackrel{\text{def}}{=} \Box((\lceil\lceil P\rceil \wedge (\ell \geq d)) \Rightarrow \Diamond\lceil Q\rceil^0)$. It states that in any interval with P invariantly true and having time length of d or more, there must be some position with Q true. Our aim is to check the validity of the following *GOAL* formula for various integer values of d_1, d_2.

$$GOAL \;\stackrel{\text{def}}{=}\; Follows(P,Q,d_1) \;\Rightarrow\; Within(P,Q,d_2)$$

Unfortunately, $SCID(GOAL)$ does not hold and we cannot reduce the problem to equivalent *QDDC* validity checking. However, we can use the digitization approximation technique. We compute QDDC approximations $\beta(ST(GOAL))$ and $\beta(WT(GOAL))$ of the IDL formula *GOAL* using Definitions 7, 8, 9, 10.

$$\beta(ST(GOAL)) \;=\; \beta(FollowsWk(P,Q,d_1)) \;\Rightarrow\; \beta(Within(P,Q,d_2))$$

$$\beta(WT(GOAL)) \;=\; \beta(FollowsWk(P,Q,d_1-1)) \Rightarrow \beta(Within(P,Q,d_2))$$

$$\beta(FollowsWk(P,Q,d)) \;=\; \neg(true\frown\lceil E\rceil^0\frown ((\lceil\lceil P\rceil \wedge (\Sigma\neg M > d))\frown\lceil E\rceil^0\frown\lceil\neg Q\rceil^0)\frown\lceil E\rceil^0\frown true)$$

$$\beta(Within(P,Q,d)) \;=\; \neg(true\frown\lceil E\rceil^0\frown ((\lceil\lceil P\rceil \wedge (\Sigma\neg M \geq d)) \wedge \neg(true\frown\lceil Q \wedge E\rceil^0\frown true))\frown\lceil E\rceil^0\frown true)$$

The resulting formulae can be analysed using the QDDC validity checker DCVALID [7] for various constants d_1, d_2.

Experimental Verification with DCVALID The verification was carried out using DCVALID1.4 tool running on Pentium4 1.4GHz PC system running Linux 2.4.18 kernel.

Case 1 For $d_1 = 10$, $d_2 = 12$, the validity checker returned the result that $\models_{QDDC}\; CONSIST(P,Q) \;\Rightarrow \beta(ST(GOAL))$.
Its verification took 1.57 seconds of CPU time. From this, by using Theorems 9(1) and 7(1), we concluded that $\models_R\; GOAL$.

Case 2 For $d_1 = 10$, $d_2 = 7$, the validity checker returned the result that $\not\models_{QDDC}$ $CONSIST(P,Q) \Rightarrow \beta(ST(GOAL))$. The tool gave a counter example, but as this is not guaranteed to be a counter example for the original formula $GOAL$, we disregarded it. Instead, we invoked the tool with the weak approximation $\beta(WT(GOAL))$. The validity checker returned the result that $\not\models_{QDDC}$ $CONSIST \Rightarrow \beta(WT(GOAL))$ and gave the following counter example in 0.17 seconds of CPU time.

```
MT                    00000000
ES                    10000001
P                     11111110
Q                     00000000
```

This corresponds to the IDL behaviour $\theta = (P \wedge \neg Q, 0) \rightarrow (\neg P \wedge \neg Q, 7)$. By using Theorems 9(2) and 7(3) we concluded that $\theta \not\models_R GOAL$. Thus, we generated a counter-example for $GOAL$.

One limitation of our method is that it would fail to conclude anything about $\models_R GOAL$ in case we get $\not\models_Z \beta(ST(GOAL))$ and $\models_Z \beta(WT(GOAL))$, as this would only establish that $\not\models_R ST(D)$ and $\models_R WT(D)$.

7 Related Work

The technique of digitization is aimed at reducing a dense-time verification problem to a discrete-time verification problem. There is some experimental evidence that, with existing techniques, automatic verification of discrete-time systems may be significantly easier than the verification of similar dense-time systems [2]. This makes digitization a practically important approach for the verification of dense-time properties.

In their poineering work Henzinger, Manna and Pnueli [4] proposed the digitization technique for the verification of dense-time properties, and formulated the digitization theorem which gives sufficient conditions for reducing the model checking of dense-time properties to the model checking of discrete-time properties. In particular, they showed that for CID properties, dense-time validity and discrete-time validity are equivalent. They also studied some properties of logic MTL which are CID.

In this paper, we have considered digitization of IDL properties. It is quite hard to establish which IDL formulae are CID. To obviate this, we have given a new notion of *Strong Closure Under Inverse Digitization* (SCID) which implies CID. Almost all operators of IDL preserve SCID, giving us powerful structural characterisations of formulae which are SCID (see Theorem 4). Moreover, for formulae which are not SCID, we have given approximations to stronger and weaker formulae which are SCID (Theorem 7). IDL is a highly expressive dense-time logic with a powerful $\frown$ (chop) operator and a notion of $\int P$ giving the accumulated amount of time for which proposition P holds in a time interval. Digitization (CID) of such properties is one of the main contributions of this paper. For example, we have shown that the "critical duration" formulae [9] like $\Box(\ell \leq c \Rightarrow \int P \leq d)$ are CID. Such formulae are quite hard to verify in dense-time.

Digitization reduces verification of IDL properties to verificaiton of ZIDL properties. ZIDL formulae are interpreted over weakly monotonic integer-timed state sequences. In our next reduction, we have translated ZIDL formulae to "equivalent" QDDC formulae (Theorem 9). The translation preserves models under a suitable isomorphism. This also establishes the decidability of logic ZIDL. The use of the "count" construct ΣP of $QDDC$ in this translation is significant.

Putting all these together, we are able to mechanically reduce the validity of IDL to the validity of QDDC for a large class formulae, and to approximate this reduction in other cases. We have illustrated the use of this technique by a small example in Section 6. A more extensive example of the verification of a Minepump Specification [8] can be found in the full version of this paper.

Digitization of Duration calculus has been studied before. Under bounded-variability assumption Franzle [3] has shown the decidability of Duration Calculus. Hung and co-authors have modelled digitized behaviours directly within DC and used axioms of DC to reason about digitized behaviours [5].

References

1. R. Alur and D. L. Dill: Automata for modeling Real-time systems. In: 17^{th} *ICALP*, Lecture Notes in Computer Science, Vol 443. Springer-Verlag (1990)
2. D. Beyer: Rabiit: Verification of Real-Time Systems. In: *Workshop on Real-time Tools (RT-TOOLS'2001)*. Aalborg, Denmark (2001)
3. M. Franzle: Decidability of Duration Calculi on Restricted Model Classes. *ProCoS Technical Report Kiel MF/1*. Christian-Albrechts Universitat Kiel, Germany (1996)
4. T. A. Henzinger, Z. Manna and A. Pnueli: What good are digital clocks?. In: *ICALP'92*, Lecture Notes in Computer Science, Vol 623. Springer-Verlag (1992) 545-558
5. D. V. Hung and P. H. Giang: Sampling Semantics of Duration Calculus. In: *FTRTFT'96*, Lecture Notes in Computer Science, Vol 1135. Springer-Verlag (1996) 188-207
6. P.K. Pandya and D.V. Hung: A Duration Calculus of Weakly Monotonic Time, In: A.P.Ravn and H. Rischel (eds.): *FTRTFT'98*. Lecture Notes in Computer Science, Vol 1486. Springer-Verlag (1998)
7. P.K. Pandya: Specifying and Deciding Quantified Discrete-time Duration Calculus Formulae using DCVALID: An Automata Theoretic Approach. In: *Workshop on Real-time Tools (RTTOOLS'2001)*. Aalborg, Denmark (2001)
8. P.K. Pandya: Interval Duration Logic: Expressiveness and Decidability. In: *Workshop on Theory and Practice of Timed Systems (TPTS'2002)*, Electronic Notes in Theoretical Computer Science, ENTCS **65.6**. Elsevier Science B.V. (2002)
9. A.P. Ravn: Design of Real-time Embedded Computing Systems. Department of Computer Science, Technical University of Denmark (1994)
10. Zhou Chaochen, C. A. R. Hoare and A. P. Ravn. A Calculus of Durations, Information Processing Letters, **40**(5). (1991) 269-276