



The United Nations
University

UNU/IIST

International Institute for
Software Technology

A Duration Calculus with Infinite Intervals

Zhou Chaochen, Dang Van Hung and
Li Xiaoshan

February, 1995

UNU/IIST

UNU/IIST enables developing countries to attain self-reliance in software technology by: (i) their own development of high integrity computing systems, (ii) highest level post-graduate university teaching, (iii) international level research, and, through the above, (iv) use of as sophisticated software as reasonable.

UNU/IIST contributes through: (a) advanced, joint industry-university advanced development projects in which rigorous techniques supported by semantics-based tools are applied in case studies to large scale software developments, (b) own and joint university and academy institute research in which new techniques for application domain and computing platform modelling, requirements capture, software engineering and programming are being investigated, (c) advanced, post-graduate and post-doctoral level courses which typically teach Design Calculi oriented software development techniques, (d) events [panels, task forces, workshops and symposia], and (e) dissemination.

Application-wise, the advanced development projects presently focus on software to support large-scale infrastructure systems such as transport systems (railways, airlines, air traffic, etc.), manufacturing industries, telecommunications, etc., and are thus aligned with UN and International Aid System concerns. UNU/IIST is a leading research centre in the area of Duration Calculi, i.e. techniques applicable to *real-time, reactive, hybrid & safety critical systems*. The research projects parallel and support the advanced development projects.

At present, the technical focus of UNU/IIST in all of the above is on applying, teaching, researching, and disseminating Design Calculi oriented techniques and tools for trustworthy software development. UNU/IIST currently emphasises techniques that permit proper development steps and interfaces. UNU/IIST also endeavours to promulgate sound project and product management principles.

UNU/IIST's primary dissemination strategy is to act as a clearing house for reports from research and technology centres in industrial countries to industries and academic institutions in developing countries. At present more than 200 institutions worldwide contribute to UNU/IIST's report collection while UNU/IIST at the same time subscribes to more than 125 international scientific and technical journals. Information on reports received (and produced) and on journal articles is to be disseminated regularly to developing country centres — which are then free to order a reasonable number of report and article copies from UNU/IIST.

Dines Bjørner, Director — 1.7.1992–31.6.1997

UNU/IIST Reports are either *R*esearch, *T*echnical, *C*ompendia or *A*dministrative reports:

\mathcal{R} Research Report • \mathcal{T} Technical Report • \mathcal{C} Compendium • \mathcal{A} Administrative Report



The United Nations
University

UNU/IIST

**International Institute for
Software Technology**

P.O. Box 3058
Macau

A Duration Calculus with Infinite Intervals

Zhou Chaochen, Dang Van Hung and
Li Xiaoshan

Abstract

This paper introduces infinite intervals into the Duration Calculus [32]. The extended calculus defines a state duration over an infinite interval by a property which specifies the limit of the state duration over finite intervals, and excludes the description operator. Thus the calculus can be established without involvement of unpleasant calculation of infinity. With limits of state durations, one can treat conventional liveness and fairness, and can also measure liveness and fairness through properties of limits. Including both finite and infinite intervals, the calculus can, in a simple manner, distinguish between terminating behaviour and nonterminating behaviour, and therefore directly specify and reason about sequentiality.

Zhou Chaochen is a Principle Research Fellow of UNU/IIST, on leave from the Software Institute, the Chinese Academy of Sciences. He is an Academician of the Chinese Academy. His research interest is in Formal Technique of Programming, including formal semantics, concurrency, and design calculi. E-mail: zcc@iist.unu.edu.

Dang Van Hung is from the Institute of information Technology of National Center for Natural Science and Technology of Vietnam, where he is a researcher. He is a Fellow of UNU/IIST from April 1994 to July 1995. His research interest is in Formal Technique of Programming, Real-Time system specification, Concurrent and Distributed systems. E-mail: dvh@iist.unu.edu

Li Xiaoshan is a fellow of UNU/IIST, on leave of absence from the Software Institute, the Chinese Academy of Sciences. His research interest is in formal methods to design real-time systems and hardware verification. E-mail: lxs@iist.unu.edu

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Duration Calculus (DC) | 5 |
| 3 | Duration Calculus with Infinite Intervals (DC^i) | 10 |
| 4 | Inference Rules of DC^i | 17 |
| 5 | DC^i Specifications | 23 |
| 5.1 | Limit, Liveness and Fairness | 24 |
| 5.2 | Program Semantics | 29 |
| 6 | Discussion | 31 |

1 Introduction

The Duration Calculus (abbr. DC) [32] is an extension of the Interval Temporal Logic [15]. It is restricted to *finite* intervals, and uses *chop* (\frown) as the only modality. *Chop* is a *constructing* operator, by which, from a given interval, we can reach its subintervals. This restriction prohibits the DC from specifying *unbounded* liveness and fairness properties of computing systems, such as a circuit which oscillates *forever*, or two users which are served so fairly that they have equal service durations *at last*.

In order to cope with unbounded liveness and fairness, [17], [3] and [21] introduce *expanding* modalities, while keep the restriction to finite intervals. [17] defines two *weakest inverses* of the chop. [3] generalizes the chop by introducing *backward* intervals. Inspired by [25], [21] introduces two *expanding* modalities into the DC. They are designated T and D in [25], and \triangleright and \triangleleft in [21]. By \triangleright and \triangleleft , from a given interval one can refer to its superintervals: an interval $[a, b]$ satisfies

$$D_1 \triangleright D_2$$

iff there exists c such that $c \geq b$, $[a, c]$ satisfies D_1 , and $[b, c]$ satisfies D_2 . \triangleleft is defined symmetrically. With \triangleright and \triangleleft , one can specify unbounded liveness and fairness. Let Boolean function W model the output of an oscillator.¹ The oscillator can therefore be specified by

$$\neg((\neg(true \triangleright (\neg[W] \wedge \neg[\neg W]))) \triangleright true)$$

where $[W]$ ($[\neg W]$) means that W has value 1 (0) everywhere inside an interval. The formula can be read as:

There is no such a right expansion that one cannot find out an interval right of the expansion, inside which W is neither everywhere 1 nor everywhere 0.

Let Boolean functions S_1 and S_2 model system service for the two users respectively. The specification that S_1 and S_2 at last have equal duration can be formulated by

$$\forall \epsilon > 0. (\neg(|\int S_1 - \int S_2| \geq \epsilon \triangleright true)) \triangleright true$$

where $\int S$ is a duration expression of the DC. It is a function from intervals to real numbers. Given interval $[a, b]$, the value of $\int S$ is defined as

$$\int_a^b S(t) dt$$

¹By $W(t) = 1$ (0), we mean that the output is connected to power (ground) at time t .

i.e. the presence duration of state S in the interval. The above formula can be read as:

For any $\epsilon > 0$, one can find out a right expansion, such that no further right expansion will make a difference between presence durations of S_1 and S_2 greater than or equal to ϵ .

Although the DC with the two additional modalities can express liveness and fairness properties of computing systems, it still has problems in differentiating finite system behaviour from infinite one *syntactically*. An infinite behaviour determines system states eternally, while a finite behaviour represents a termination, which determines system states up to some moment in time, and allows arbitrary continuation. It seems that, in order to define sequential composition ($;$) in a finite interval based DC, an extra state (like \surd in CSP [10]) might be necessary, which syntactically indicates a termination. The CSP school introduces not only \surd state, but also *refusals* (or *ready sets*), in order to deal with liveness properties in a *finite trace* based language. It therefore exhibits another possible approach to extending the expressiveness of the DC, which introduces new states instead of new modalities.

The third approach to extend the DC for specifying and reasoning about unbounded liveness and fairness properties is to remove the restriction of finite intervals by introducing infinite intervals into the calculus. In [20] and [16], a kind of infinite interval has been introduced. [20] is an extension of Temporal Logic, and [16] extends Interval Temporal Logic with infinite intervals. [16] lets *inf* stand for infinite intervals, and includes an axiom

$$(D \wedge inf); C \equiv D \wedge inf$$

which defines nicely the sequential composition of a nonterminate (infinite) behaviour: $(D \wedge inf)$. Unfortunately both of them have not taken account of interval length in their logics yet, and cannot be used for designing hard real-time systems.

Since the length of an infinite interval is *infinity*: ∞ , the treatment of infinity becomes an obstacle to the development of an infinite Duration Calculus. Many of the textbooks of mathematical analysis include algebraic laws of infinity, such as

$$\begin{aligned} \infty + \infty &= \infty, \\ \infty \cdot \infty &= \infty. \end{aligned}$$

However those laws are far from complete. For example, they do not provide laws for subtraction: $(\infty - \infty)$.

[35] attempts the problem by assigning \perp ('undefined') to the length of infinite intervals, and assigning false to atomic formulas with occurrence of \perp . Therefore

$$\ell = \ell$$

becomes false for infinite intervals, where ℓ designates interval length. It unfortunately opposes mathematical common sense.

In the foundations of mathematics, the Intuitionism denies the existence of infinity, but recognizes a 'manifold of possibilities open towards infinity' [26]. Inspired by the Intuitionism, this paper will establish a Duration Calculus of both finite and infinite intervals (called DC^i), which treats ∞ as a property rather than an entity. DC^i is able to formulate and reason about properties which characterize infinity, but rejects calculations of infinity by excluding the description operator from the calculus.

In DC^i , a DC formula D is satisfied by infinite interval $[a, \infty)$, iff for any $b (\geq a)$, $[a, b]$ satisfies D . In other words, D is an invariant for all finite prefixes of the infinite interval. That D is satisfied by an infinite interval is designated as D^i , and called *infinite satisfaction*. Similarly the satisfaction of D by a finite interval is designated as D^f , and called *finite satisfaction*. DC^i is a first order logic of the infinite and finite satisfactions of DC formulas.

The unbounded liveness of system state S can be represented by infinite presence of S . It can be formulated in DC^i by

$$\forall x \exists y > x. (\ell > y \Rightarrow (\ell = y) \frown [S] \frown true)^i$$

where \frown designates the *chop* operator. An interval satisfies $(D_1 \frown D_2)$, iff the interval can be chopped into two subintervals such that the left subinterval satisfies D_1 , and the right one satisfies D_2 . The plain meaning of the previous formula is that, after any time x , one can always find a time y such that S appears right after y . A formulation of the oscillator can be obtained by postulating infinite presence of both W and $\neg W$.

One can also measure unbounded liveness of state S through its duration over an infinite time interval.² An infinite duration of state S over an infinite interval can be specified in DC^i

$$\forall x \exists y. (\ell \geq y \Rightarrow \int S > x)^i.$$

²That is why we choose the term – *unbounded* liveness and fairness, instead of *qualitative* liveness and fairness.

This formula is almost a direct translation of the Cauchy definition of the limit of infinity. Similarly we can specify finite limits such as v is the value of fS over an infinite interval:

$$\forall \epsilon > 0 \exists x. (\ell \geq x \Rightarrow |v - fS| < \epsilon)^i.$$

The unbounded fairness between states S_1 and S_2 can be measured by the ratio of their durations over an infinite interval. For example

$$\forall \epsilon > 0 \exists x. (\ell \geq x \Rightarrow |fS_1 - r \cdot fS_2| < \epsilon)^i.$$

specifies that the limit of the ratio of the duration of S_1 to the duration of S_2 is r .

With infinite intervals, we can establish a theory of limits of state durations. Unbounded liveness and fairness properties are essentially kinds of limit properties of state durations. A limit theory can facilitate specifications and verifications of liveness and fairness properties. It can also help us specify properties of hybrid systems, e.g. system stability.

All behaviour defined by formula \mathcal{G} of DC^i terminate, iff

$$\mathcal{G} \Rightarrow fin$$

where fin specifies finite intervals, and

$$fin \hat{=} true^f$$

Similarly, \mathcal{G} defines non-terminating behaviour, iff

$$\mathcal{G} \Rightarrow inf$$

where inf specifies infinite intervals, and

$$inf \hat{=} true^i$$

Therefore an operator of sequential composition can be simply defined in DC^i .

The syntax and semantics of DC^i are explained in Section 3 in detail. In Section 2, the DC is briefly reviewed. In Section 5, we list various examples of DC^i specifications, which include specifications of duration limit, liveness and fairness properties, and a definition of the sequential composition operator.

Regarding inference rules of DC^i , we will of course adopt the rules for first order predicate calculus. Besides, by the definition of finite and infinite satisfactions, we can derive DC^i theorems from DC theorems. For example, if

$$\vdash_{DC} D$$

then all finite intervals will satisfy D^f , and all infinite intervals will satisfy D^i . Hence

$$\vdash_i D^f \vee D^i$$

where \vdash_i is an abbreviation of \vdash_{DC^i} . An inference system is given in Section 4. We cannot conclude the completeness of the inference system in the paper.

2 Duration Calculus (DC)

In this section, the Duration Calculus is briefly reviewed [31].

Research into the Duration Calculus was started by the ProCoS project (Provably Correct Systems: Esprit BRA 3104) in 1989, when the project was developing formal techniques for designing real-time safety critical systems. Several calculi have been developed since then. They are the Duration Calculus, the Extended Duration Calculus, the Mean Value Calculus, the Probabilistic Duration Calculus and the three calculi as mentioned in Section 1.

The Duration Calculus is a real-time interval logic [32]. It formalizes integrals of Boolean functions over finite intervals, and can be used to specify and reason about timing and logical constraints on discrete states of a system. All the other calculi are extensions of the Duration Calculus. The Extended Duration Calculus [37] extends the Duration Calculus with piecewise continuity/differentiability of functions. It can capture properties of continuous states, and can be used for designing hybrid systems. The Mean Value Calculus [36] extends the Duration Calculus by replacing integrals of Boolean functions with their mean values, so that it can use δ -functions to represent instant actions such as communications and events. The Mean Value Calculus can be used to refine from state based specifications via mixed state and event specifications to event based specifications. The Probabilistic Duration Calculus [13, 14, 1] provides

designers with a set of rules to reason about and calculate dependability of a system with respect to its components. As explained in Section 1, the other three calculi introduce more modalities (or inverse intervals), so that they can deal with unbounded liveness and fairness properties.

The Duration Calculi have been used to specify a number of examples of hybrid systems [18, 19, 22, 2, 28, 29, 27, 9]. The Calculi have also been used to define real-time semantics for Occam-like languages [33, 8], and to specify real-time behaviour of schedulers [33, 30] and circuits [7].

As to mechanical support tool for the Duration Calculi, the decidability and undecidability results of the Duration Calculus have been published [34, 4], and an automatic model checker for a decidable subclass of the Duration Calculus has been implemented in Standard ML [24]. Efficient model checking algorithms for Linear Duration Invariants have been discovered [11, 38]. They employ the technique of Linear Programming. A tool for constructing DC specifications and checking DC proofs has been implemented by using PVS [23].

We present below some of the main features of DC.

In DC, a system state represents a logical property of the system. Presence of a state means that the property holds. Absence means that the property does not hold. State is modelled by Boolean functions over time: $\mathbf{R} \rightarrow \{0, 1\}$, where time is modelled by real numbers: \mathbf{R} . When the function has value 1 at time t , it represents presence of the state at t . Symmetrically, value 0 represents absence of the state at a time.

For an arbitrary state S and an arbitrary finite interval $[a, b]$, the duration of S , designated $\int S$, is defined by the value of the integral of S over the interval $[a, b]$

$$\int_a^b S(t) dt$$

It follows that $\int 1$ equals $(b - a)$, i.e. the length of the finite interval. Thus we introduce an abbreviation to designate interval length

$$\ell \triangleq \int 1$$

The following BNFs review the inductive definitions of the DC syntax. Let S stand for states, τ for terms, A for atomic formulas, and D for duration formulas.

$$S ::= P \mid \neg S \mid S \vee S$$

where P stands for primitive states.

$$\tau ::= \int S \mid r \mid x \mid f(\tau, \dots, \tau) \mid \tau + \tau \mid \tau - \tau \mid \dots$$

where r stands for constants, x for global variables, and f for function symbols.

According to the meaning of $\int S$ presented above, the denotations of terms are functions from finite intervals to reals, called *interval functions*: $\mathbf{I} \rightarrow \mathbf{R}$, where

$$\mathbf{I} \triangleq \{[a, b] \mid a, b \in \mathbf{R} \ \& \ b \geq a\}$$

Therefore a model of DC formula, designated Π , consists of interpretations for primitive states P , global variables x , and function symbols f . State P is interpreted as a Boolean function in $(\mathbf{R} \rightarrow \{0, 1\})$, x as a real number (a constant interval function), and f as a function in $(\mathbf{R}^n \rightarrow \mathbf{R})$ (a constant functional of interval functions), where n is the arity of f . The arithmetical operators and Boolean operators are here applied to functions, and their interpretations are pointwise extensions of the standard one.

The atomic formulas are

$$A ::= \text{true} \mid \text{false} \mid \tau = \tau \mid \tau > \tau \mid \dots$$

and the formulas are

$$D ::= A \mid \neg A \mid D \vee D \mid D \frown D \mid \exists x.D$$

where x is a global variable, and \frown designates the *chop* operator.³

The semantics of the formulas can be defined by formula satisfactions. Given model Π , a finite interval $[a, b]$ satisfies formula D under model Π , written as

$$\Pi, [a, b] \models_{DC} D$$

iff the values of the terms over $[a, b]$ satisfy D , where the meaning of operators $=$ and $>$, connectives \neg and \vee and quantifier $\exists x$ is standard, and the satisfaction of $B \frown C$ by finite

³We overload the Boolean operator (\neg and \vee). They apply to states and also formulas.

interval $[a, b]$ is defined as that there exists m ($a \leq m \leq b$), such that B is satisfied by $[a, m]$, and C is satisfied by $[m, b]$. When formula D is satisfied by all finite intervals under model Π , we say that D is satisfied by Π , written as

$$\Pi \models_{DC} D$$

If D is satisfied by any models, D is called *valid*, written as

$$\models_{DC} D$$

In DC, we also use the following abbreviations.

$$[S] \hat{=} (fS = \ell) \wedge (\ell > 0)$$

$[S]$ means that state S is present (almost) everywhere in a non-point interval.

$$[\] \hat{=} (\ell = 0)$$

$[\]$ defines point intervals.

We also use D^* as an abbreviation of $([\] \vee D)$.

For a formula D

$$\diamond D \hat{=} true \frown D \frown true$$

Thus $\diamond D$ is satisfied by a finite interval in which D holds for some subinterval, and similarly

$$\diamond D \hat{=} D \frown true$$

$\diamond D$ holds for a finite interval, iff D holds for a prefix of the interval.

The dualities are

$$\begin{aligned} \square D &\hat{=} \neg \diamond \neg D \\ \boxplus D &\hat{=} \neg \diamond \neg D \end{aligned}$$

They are true of a finite interval, in which D holds in every subinterval or prefix respectively.

DC is an extension of Interval Temporal Logic (ITL). It therefore employs all the axioms and rules of first order ITL, but also has a small set of additional axioms and rules, which constitute a relatively complete inference system [6]. They are

Axiom 1: $\int 0 = 0$

Axiom 2: For an arbitrary state S

$$\int S \geq 0$$

Axiom 3: For arbitrary states S_1 and S_2

$$\int S_1 + \int S_2 = \int (S_1 \vee S_2) + \int (S_1 \wedge S_2)$$

Axiom 4: Let S be a state and r, s non-negative reals

$$(\int S = r + s) \Leftrightarrow (\int S = r) \wedge (\int S = s)$$

States are assumed *finitely variable*. That is, a state can have only a finite number of alternations of its presence and absence in a finite interval. DC establishes two induction rules which axiomatize finite variability. Let X denote a formula letter occurring in the formula $R(X)$ and let S be a state.

Forward Induction Rule:

If $R(\lceil \])$ holds, and $R(X \vee (X \frown \lceil S \rceil)) \wedge R(X \vee (X \frown \lceil \neg S \rceil))$ is provable from $R(X)$, then $R(\text{true})$ holds.

Backward Induction Rule:

If $R(\lceil \])$ holds, and $R(X \vee (\lceil S \rceil \frown X)) \wedge R(X \vee (\lceil \neg S \rceil \frown X))$ is provable from $R(X)$, then $R(\text{true})$ holds.

3 Duration Calculus with Infinite Intervals (DCⁱ)

DCⁱ is a first order logic of finite and infinite satisfactions of DC.

DCⁱ designates finite satisfaction of DC formula D by D^f , and infinite satisfaction of D by D^i . D^f holds for finite intervals only, and D^i for infinite intervals only. DCⁱ shares models with DC. A finite interval satisfies D^f under a model, iff the interval satisfies D in terms of the semantics of DC. An infinite interval satisfies D^i under a model, iff all its *finite* prefixes satisfy D in terms of the semantics of DC. Let $[a, b]$ stand for finite intervals and $[a, \infty)$ for infinite intervals henceforth. We define

1. $\Pi, [a, b] \models_i D^f$ iff $\Pi, [a, b] \models_{DC} D$
where \models_i designates the satisfaction relation of DCⁱ.
2. $\Pi, [a, \infty) \not\models_i D^f$
3. $\Pi, [a, b] \not\models_i D^i$
4. $\Pi, [a, \infty) \models_i D^i$ iff $\Pi, [a, b] \models_{DC} D$ for all $b (\geq a)$.

D^f and D^i constitute atomic formulas of DCⁱ. Let \mathcal{G} stand for formulas of DCⁱ. The BNF that defines syntax of DCⁱ can be given by

$$\mathcal{G} ::= D^f \mid D^i \mid \neg\mathcal{G} \mid \mathcal{G} \vee \mathcal{G} \mid \exists x\mathcal{G}$$

where D stands for formulas of DC, and x for global variables of DC. We here adopt the standard semantics for \neg , \vee and $\exists x$, and the conventional way of introducing \wedge , \Rightarrow , \Leftrightarrow and $\forall x$.

Formula \mathcal{G} is satisfied by model Π , iff \mathcal{G} is satisfied by any interval (finite or infinite) under model Π . That is,

$$\Pi \models_i \mathcal{G}$$

iff for any a and b ($b \geq a$)

$$\begin{array}{l} \Pi, [a, b] \models_i \mathcal{G}, \text{ and} \\ \Pi, [a, \infty) \models_i \mathcal{G} \end{array}$$

Formula \mathcal{G} is valid, iff \mathcal{G} is satisfied by any models. That is,

$$\models_i \mathcal{G}$$

iff for any model Π

$$\Pi \models_i \mathcal{G}$$

Example 1: For any Π , a and $b (\geq a)$

$$\begin{aligned} \Pi, [a, b] &\models_i \textit{fin}, \text{ and} \\ \Pi, [a, \infty) &\models_i \textit{inf} \end{aligned}$$

where

$$\begin{aligned} \textit{fin} &\hat{=} \textit{true}^f \\ \textit{inf} &\hat{=} \textit{true}^i \end{aligned}$$

fin represents all finite intervals, and *inf* all infinite intervals. Thus, for any Π ,

$$\Pi \models_i (\textit{fin} \vee \textit{inf})$$

So

$$\models_i (\textit{fin} \vee \textit{inf})$$

We let

$$\begin{aligned} \textit{True} &\hat{=} (\textit{fin} \vee \textit{inf}) \\ \textit{False} &\hat{=} \neg \textit{True} \end{aligned}$$

They represent the *truth* and *falsehood* of DCⁱ.

Example 2: For Π which interprets P with constant function 1, we have

$$\Pi, [a, b] \models_i [P]^*{}^f$$

for all a and b ($\geq a$), and also

$$\Pi, [a, \infty) \models_i [P]^{*i}$$

Hence

$$\Pi \models_i ([P]^{*f} \vee [P]^{*i})$$

However $([P]^{*f} \vee [P]^{*i})$ is not valid

$$\not\models_i ([P]^{*f} \vee [P]^{*i})$$

since all other models do not satisfy the formula.

Example 3:

$$\Pi, [a, \infty) \models_i \exists x.(\ell > x \Rightarrow (\ell = x) \wedge [P])^i$$

iff Π assigns 1 to P from some time to eternity. However, for any Π and a ,

$$\Pi, [a, \infty) \models_i (\exists x(\ell > x \Rightarrow (\ell = x) \wedge [P]))^i$$

since for any Π and $[a, b]$, when $x > (b - a)$, we have

$$\Pi, [a, b] \models_{DC} \ell > x \Rightarrow (\ell = x) \wedge [P]$$

Note that the Example shows that

$$(\exists x D)^i \Rightarrow \exists x. D^i$$

is not valid. So \exists cannot be distributed over i .

The followings are useful properties of satisfaction and validity of DC^i formula. They can be easily derived from the definitions above.

Monotonicity: For any Π , if

$$\Pi \models_{DC} (D_1 \Rightarrow D_2)$$

then

$$\begin{array}{l} \Pi \models_i (D_1^f \Rightarrow D_2^f) \quad \text{and} \\ \Pi \models_i (D_1^i \Rightarrow D_2^i) \end{array}$$

***f*&*i*-Exclusion:** The finite and infinite satisfactions are mutually excluded.

1. ($fin \Leftrightarrow \neg inf$)
2. For any Π , if

$$\Pi \models_i (\mathcal{G}_1 \Rightarrow fin)$$

and

$$\Pi \models_i (\mathcal{G}_2 \Rightarrow inf)$$

then

$$\Pi \models_i \forall x.(\mathcal{G}_1 \vee \mathcal{G}_2) \Rightarrow (\forall x.\mathcal{G}_1 \vee \forall x.\mathcal{G}_2)$$

Proof. We only give proof for the second property, since the mutual exclusion of *fin* and *inf* is clear. Suppose that

$$\Pi, [a, b] \models_i \forall x.(\mathcal{G}_1 \vee \mathcal{G}_2)$$

Then for any x

$$\Pi, [a, b] \models_i (\mathcal{G}_1 \vee \mathcal{G}_2)$$

By the assumption

$$\Pi \models_i (\mathcal{G}_2 \Rightarrow inf)$$

we have

$$\Pi, [a, b] \not\models_i \mathcal{G}_2$$

Hence for any x

$$\Pi, [a, b] \models_i \mathcal{G}_1$$

That is

$$\Pi, [a, b] \models_i \forall x.\mathcal{G}_1$$

So

$$\Pi, [a, b] \models_i \forall x.(\mathcal{G}_1 \vee \mathcal{G}_2) \Rightarrow (\forall x.\mathcal{G}_1 \vee \forall x.\mathcal{G}_2)$$

Similarly we can prove

$$\Pi, [a, \infty) \models_i \forall x.(\mathcal{G}_1 \vee \mathcal{G}_2) \Rightarrow (\forall x.\mathcal{G}_1 \vee \forall x.\mathcal{G}_2)$$

***f*-Distributivity:** \neg , \vee and \exists distribute over f .

1. $\models_i \neg D^f \Leftrightarrow (inf \vee (\neg D)^f)$
2. $\models_i D_1^f \vee D_2^f \Leftrightarrow (D_1 \vee D_2)^f$
3. $\models_i \exists x. D^f \Leftrightarrow (\exists x D)^f$

***i*-Closure:** The infinite satisfaction implies a property of prefix closure.

$$\models_i D^i \Leftrightarrow (\Box D)^i$$

Proof. Suppose that there are Π and $[a, \infty)$ which satisfy D^i

$$\Pi, [a, \infty) \models_i D^i$$

Then for any $b (\geq a)$

$$\Pi, [a, b] \models_{DC} D$$

This implies that for arbitrary given $b (\geq a)$ and any $c (b \geq c \geq a)$

$$\Pi, [a, c] \models_{DC} D$$

Hence by the definition of $\Box D$

$$\Pi, [a, b] \models_{DC} \Box D$$

Therefore

$$\Pi, [a, \infty) \models_i (\Box D)^i$$

So the proof of the first half of the equivalence is completed. The second half can be derived from Monotonicity, since

$$\models_{DC} \Box D \Rightarrow D$$

***i*-Distributivity:** *i*-Distributivity is more complicated than *f*-Distributivity.

1. $\models_i \neg D^i \Leftrightarrow (fin \vee \exists x. (\ell = x \Rightarrow \neg D)^i)$

Proof. By the satisfaction definition, for any Π and $[a, b]$

$$\Pi, [a, b] \models_i \neg D^i$$

Thus

$$\Pi, [a, b] \models_i \neg D^i \Leftrightarrow (fin \vee \exists x. (\ell = x \Rightarrow \neg D)^i)$$

It remains to show the equivalence with respect to the infinite satisfactions. Suppose that there are Π and $[a, \infty)$ such that

$$\Pi, [a, \infty) \models_i \neg D^i$$

This means

$$\Pi, [a, \infty) \not\models_i D^i$$

By the satisfaction definition, there must exist $c (\geq a)$

$$\Pi, [a, c] \not\models_{DC} D$$

That is,

$$\Pi, [a, c] \models_{DC} \neg D$$

Then we let $x = (c - a)$, and for any $b (\geq a)$ we have

$$\Pi, [a, b] \models_{DC} (\ell = x \Rightarrow \neg D)$$

Hence by the satisfaction definition again

$$\Pi, [a, \infty) \models_i (\ell = x \Rightarrow \neg D)^i$$

By the rule \exists_+ of first order logic

$$\Pi, [a, \infty) \models_i \exists x. (\ell = x \Rightarrow \neg D)^i$$

Thus we complete the proof of \Rightarrow of the equivalence. The proof of \Leftarrow can be presented in a similar way. We leave it out.

2. $\models_i (D_1^i \vee D_2^i) \Leftrightarrow (\Box D_1 \vee \Box D_2)^i$

Proof. The first half of the equivalence can be proved as follows.

$$\begin{aligned} \models_i D_1^i &\Rightarrow (\Box D_1)^i && \text{(by } i\text{-Closure)} \\ \models_{DC} \Box D_1 &\Rightarrow (\Box D_1 \vee \Box D_2) && \text{(by } \vee_+ \text{)} \\ \models_i (\Box D_1)^i &\Rightarrow (\Box D_1 \vee \Box D_2)^i && \text{(by Monotonicity)} \\ \models_i D_1^i &\Rightarrow (\Box D_1 \vee \Box D_2)^i && \text{(by Transitivity of } \Rightarrow \text{)} \end{aligned}$$

Similarly, we can prove

$$\models_i D_2^i \Rightarrow (\Box D_1 \vee \Box D_2)^i$$

Thus we conclude

$$\models_i (D_1^i \vee D_2^i) \Rightarrow (\Box D_1 \vee \Box D_2)^i$$

It remains to show the second half of the equivalence. Suppose

$$\Pi, [a, \infty) \models_i (\Box D_1 \vee \Box D_2)^i$$

Then for any $b (\geq a)$ by the satisfaction definition

$$\Pi, [a, b] \models_{DC} (\Box D_1 \vee \Box D_2)$$

That is,

$$\Pi, [a, b] \models_{DC} \Box D_1$$

or

$$\Pi, [a, b] \models_{DC} \Box D_2$$

Therefore at least one of $\Box D_1$ and $\Box D_2$ must be satisfied by infinitely many intervals under model Π . They all start from a , and cover $[a, \infty)$. Suppose that there are b_j ($\geq a$) ($j = 1, 2, \dots$)

$$\Pi, [a, b_j] \models_{DC} \Box D_1$$

and

$$\lim_{j \rightarrow \infty} b_j = \infty.$$

Then, for any b ($\geq a$), we can find a b_n such that ($b_n \geq b$). Hence

$$\Pi, [a, b] \models_{DC} \Box D_1$$

by the definition of \Box and

$$\Pi, [a, b_n] \models_{DC} \Box D_1.$$

Thus by the satisfaction definition

$$\Pi, [a, \infty) \models_i (\Box D_1)^i.$$

So by Monotonicity we conclude

$$\Pi, [a, \infty) \models_i D_1^i,$$

and then the second half of the equivalence.

3. $\models_i \forall x.D^i \Leftrightarrow (\forall x D)^i$

Proof.

$$\models_i (\forall x D)^i \Rightarrow \forall x.D^i$$

can be derived from Monotonicity and \forall_+ . Conversely, suppose that

$$\Pi, [a, \infty) \models_i \forall x.D^i$$

That is, for any x

$$\Pi, [a, \infty) \models_i D^i$$

Then by the satisfaction definition, for any b ($\geq a$)

$$\Pi, [a, b] \models_i D$$

Hence

$$\Pi, [a, b] \models_i \forall x D$$

Therefore

$$\Pi, [a, \infty) \models_i (\forall x D)^i$$

The proof is completed.

4 Inference Rules of DC^i

In this section, we establish an inference system for DC^i . Since DC^i is a first order logic, the inference system of DC^i will adopt all the axioms and rules of first order predicate calculus. DC^i also includes real numbers and their arithmetical operations, so the DC^i inference system contains real arithmetic. Here we do not repeat the rules taken from first order predicate calculus and real arithmetic. Of course, DC^i has its own rules for inferring the finite and infinite satisfactions of DC formulas. Those inferences in DC^i shall very much involve DC inferences. They may take DC theorems as their premises. According to the properties of satisfaction and validity of DC^i formulas which are listed in the previous section, we can introduce the following four groups of inference rules of DC^i . (*i*-Closure is implied by *i*-Distributivity of \vee .)

Monotonicity: If

$$\vdash_{DC} (D_1 \Rightarrow D_2)$$

then

$$\vdash_i (D_1^f \Rightarrow D_2^f)$$

and

$$\vdash_i (D_1^i \Rightarrow D_2^i)$$

***f*&*i*-Exclusion:** There are two rules, regarding mutual exclusion of the finite and infinite satisfactions.

1. $\vdash_i (fin \Leftrightarrow \neg inf)$
2. If

$$\vdash_i \mathcal{G}_1 \Rightarrow fin$$

and

$$\vdash_i \mathcal{G}_2 \Rightarrow inf$$

then

$$\vdash_i \forall x.(\mathcal{G}_1 \vee \mathcal{G}_2) \Rightarrow (\forall x.\mathcal{G}_1 \vee \forall x.\mathcal{G}_2)$$

***f*-Distributivity:** It contains three rules.

1. $\vdash_i \neg D^f \Leftrightarrow (inf \vee (\neg D)^f)$
2. $\vdash_i (D_1^f \vee D_2^f) \Leftrightarrow (D_1 \vee D_2)^f$
3. $\vdash_i (\exists x.D^f) \Leftrightarrow (\exists xD)^f$

***i*-Distributivity:** It also contains three rules.

1. $\vdash_i \neg D^i \Leftrightarrow (fin \vee \exists x.(\ell = x \Rightarrow \neg D)^i)$
2. $\vdash_i (D_1^i \vee D_2^i) \Leftrightarrow (\Box D_1 \vee \Box D_2)^i$
3. $\vdash_i (\forall x.D^i) \Leftrightarrow (\forall x D)^i$

With the rules, we can prove

Theorem 1.

1. $\vdash_i (D^f \Rightarrow fin)$
2. $\vdash_i (D^i \Rightarrow inf)$
3. If

$$\vdash_{DC} D,$$

then

$$\begin{aligned} \vdash_i & (fin \Rightarrow D^f) \\ & \wedge (inf \Rightarrow D^i) \\ & \wedge (D^f \vee D^i) \end{aligned}$$

Proof. The proof can be easily obtained by applying the rules of Monotonicity and the (mutual) Exclusion of *fin* and *inf*. We omit the proof.

Theorem 2. (*i*-Closure)

$$\vdash_i D^i \Leftrightarrow (\Box D)^i$$

Proof. Let D_1 be D and D_2 be *true* in the *i*-Distributivity of \vee . We have

$$(D^i \vee inf) \Leftrightarrow (\Box D \vee \Box true)^i$$

By $(\Box true \Leftrightarrow true)$ in DC, Theorem 1(2) and Monotonicity, we can derive the theorem from the previous equivalence.

Theorem 3.

1. $\vdash_i \neg false^i$

$$2. \vdash_i \forall x. \neg(\ell = x)^i$$

Proof. The proof of the first statement:

$$\begin{aligned} \neg false^i &\Leftrightarrow (fin \vee \exists x. (\ell = x \Rightarrow \neg false)^i) && (i\text{-Distributivity of } \neg) \\ &\Leftrightarrow (fin \vee \exists x. inf) && (\text{Monotonicity}) \\ &\Leftrightarrow (fin \vee inf) && (\text{first order logic}) \\ &\Leftrightarrow True && (f\&i\text{-Exclusion}) \end{aligned}$$

The proof of the second statement: for any x ,

$$\neg(\ell = x)^i \Leftrightarrow (fin \vee \exists y. (\ell = y \Rightarrow \ell \neq x)^i) \quad (i\text{-Distributivity of } \neg)$$

Let $f(x) = x + 1$.

$$\begin{aligned} (\ell = f(x) \Rightarrow \ell \neq x)^i &\Leftrightarrow inf \quad (\text{arithmetic \& Monotonicity}) \\ \exists y. (\ell = y \Rightarrow \ell \neq x)^i &\Leftrightarrow inf \quad (\exists_+) \end{aligned}$$

Hence

$$\neg(\ell = x)^i \Leftrightarrow (fin \vee inf)$$

and the proof can be completed by using \forall_+ .

In the proof of Theorem 3(2), we apply Skolemisation to DC^i formulas. We can prove a general theorem of application of Skolemisation. For example, let \mathcal{G}_1 be

$$\exists x_1 \forall y_1 \exists z_1. D_1(x_1, y_1, z_1)^i$$

and \mathcal{G}_2 be

$$\forall y_2 \exists z_2. D_2(y_2, z_2)^i$$

Theorem 4. If for any x_1 and f_1 there exists f_2 such that

$$\vdash_{DC} (\Box \forall y_1. D_1(x_1, y_1, f_1(y_1))) \Rightarrow \forall y_2. D_2(y_2, f_2(y_2))$$

then

$$\mathcal{G}_1 \Rightarrow \mathcal{G}_2$$

Proof. A proof can be derived from *i*-Closure (Theorem 2), *i*-Distributivity of \forall and rules of first order logic. We omit here the proof details.

Theorem 5. (*i*-Distributivity of \wedge)

$$\vdash_i (D_1^i \wedge D_2^i) \Leftrightarrow (D_1 \wedge D_2)^i$$

Proof. By Monotonicity, we can easily prove the \Rightarrow part of the equivalence. Conversely, we first distribute \wedge over *i* by *i*-Distributivity of \neg and \vee , and rules of first order logic

$$\begin{aligned} (D_1^i \wedge D_2^i) &\Leftrightarrow \neg(\neg D_1^i \vee \neg D_2^i) \\ &\Leftrightarrow \neg(\exists x.(\ell = x \Rightarrow \neg D_1)^i \vee \exists y.(\ell = y \Rightarrow \neg D_2)^i) \\ &\Leftrightarrow \neg\exists x, y.(\Box(\ell = x \Rightarrow \neg D_1) \vee \Box(\ell = y \Rightarrow \neg D_2))^i \\ &\Leftrightarrow \forall x, y\exists z.(\ell = z \Rightarrow \neg(\Box(\ell = x \Rightarrow \neg D_1) \vee \Box(\ell = y \Rightarrow \neg D_2)))^i \\ &\Leftrightarrow \forall x, y\exists z.(\ell = z \Rightarrow (\Diamond(\ell = x \wedge D_1) \wedge \Diamond(\ell = y \wedge D_2)))^i \end{aligned}$$

We now apply *reductio ad absurdum* to prove the conclusion. By *i*-Distributivity of \neg and *i*-Closure

$$\neg(D_1 \wedge D_2)^i \Leftrightarrow \exists u.\Box(\ell = u \Rightarrow \neg(D_1 \wedge D_2))$$

For any *u* and *f*, let *g* = *f*(*u*, *u*). We can prove

$$\begin{aligned} \vdash_{DC} & ((\ell = f(u, u) \Rightarrow (\Diamond(\ell = u \wedge D_1) \wedge \Diamond(\ell = u \wedge D_2))) \\ & \wedge \Box(\ell = u \Rightarrow \neg(D_1 \wedge D_2)) \wedge \ell = g) \Rightarrow false \end{aligned}$$

Therefore by \forall_-

$$\begin{aligned} \vdash_{DC} & (\forall x, y.(\ell = f(x, y) \Rightarrow (\Diamond(\ell = x \wedge D_1) \wedge \Diamond(\ell = y \wedge D_2))) \\ & \wedge \Box(\ell = u \Rightarrow \neg(D_1 \wedge D_2)) \wedge \ell = g) \Rightarrow false \end{aligned}$$

Hence by *reductio ad absurdum*

$$\begin{aligned} \vdash_{DC} & \forall x, y.(\ell = f(x, y) \Rightarrow (\Diamond(\ell = x \wedge D_1) \wedge \Diamond(\ell = y \wedge D_2))) \\ & \Rightarrow (\ell = g \Rightarrow \neg\Box(\ell = u \Rightarrow \neg(D_1 \wedge D_2))) \end{aligned}$$

Then by Monotonicity and *i*-Distributivity of \forall

$$\begin{aligned} \vdash_i \quad & \forall x, y. (\ell = f(x, y) \Rightarrow (\diamond(\ell = x \wedge D_1) \wedge \diamond(\ell = y \wedge D_2)))^i \\ \Rightarrow \quad & (\ell = g \Rightarrow \neg \square(\ell = u \Rightarrow \neg(D_1 \wedge D_2)))^i \end{aligned}$$

By \exists_+ ,

$$\begin{aligned} \vdash_i \quad & \forall x, y \exists z. (\ell = z \Rightarrow (\diamond(\ell = x \wedge D_1) \wedge \diamond(\ell = y \wedge D_2)))^i \\ \Rightarrow \quad & \exists v. (\ell = v \Rightarrow \neg \square(\ell = u \Rightarrow \neg(D_1 \wedge D_2)))^i \end{aligned}$$

By *i*-Distributivity of \neg

$$(D_1^i \wedge D_2^i) \Rightarrow \neg(\square(\ell = u \Rightarrow \neg(D_1 \wedge D_2)))^i$$

By \forall_+

$$(D_1^i \wedge D_2^i) \Rightarrow \forall u. \neg(\square(\ell = u \Rightarrow \neg(D_1 \wedge D_2)))^i$$

That is

$$(D_1^i \wedge D_2^i) \Rightarrow \neg \exists u. (\square(\ell = u \Rightarrow \neg(D_1 \wedge D_2)))^i$$

So

$$\begin{aligned} (D_1^i \wedge D_2^i) & \Rightarrow \neg \neg(D_1 \wedge D_2)^i \\ & \Rightarrow (D_1 \wedge D_2)^i \end{aligned}$$

We can establish a general theorem about *reductio ad absurdum* in DCⁱ. Let \mathcal{G}_1 and \mathcal{G}_2 be introduced as above. We can prove

Theorem 6. If for any x_1, f_1 and f_2 there exists g such that

$$\vdash_{DC} (\square \forall y_1. D_1(x_1, y_1, f_1(y_1)) \wedge \square \forall y_2. D_2(y_2, f_2(y_2)) \wedge \ell = g) \Rightarrow \text{false}$$

then

$$\vdash_i \mathcal{G}_1 \Rightarrow \neg \mathcal{G}_2$$

Proof. Similar to the proof given in Theorem 5. The proof is omitted here.

By f - and i -Distributivity, we can reduce DCⁱ formulas to a kind of normal form, called DCⁱ *prefix normal form*:

Theorem 7. For any DCⁱ formula \mathcal{G} , there exists DC formulas D_1 and D_2 and a prefix (a sequence of quantifiers), designated α , such that

$$\vdash_i \mathcal{G} \Leftrightarrow (D_1^f \vee \alpha.D_2^i)$$

Proof. By applying rules of first order logic, one can reduce \mathcal{G} to prefix form with a matrix of disjunctive normal form. By f - and i -Distributivity of \neg , we can transform the matrix into a disjunctive normal form without negations of atomic formulas of DCⁱ. Of course, when the i -Distributivity of \neg is applied during the transformation, the prefix will be augmented. Suppose that the prefix after the augmentation is α .

Then applying f - and i -Distributivity of \wedge , one can obtain a matrix of

$$D_{k_1}^f \vee \dots \vee D_{k_m}^f \vee D_{k_{m+1}}^i \vee \dots \vee D_{k_p}^i \bigvee_j (D_{j_1}^f \wedge D_{j_2}^i)$$

By the first rule of f & i -Exclusion

$$(D_{j_1}^f \wedge D_{j_2}^i) \Rightarrow \text{False}$$

Therefore the matrix can be reduced to

$$D_{k_1}^f \vee \dots \vee D_{k_m}^f \vee D_{k_{m+1}}^i \vee \dots \vee D_{k_p}^i$$

Applying f - and i -Distributivity of \vee , we can transform the matrix into

$$(D_q^f \vee D_2^i)$$

By the distributivity of \exists over \vee and the distributivity of \forall over \vee with formulas of mutual exclusion (stated as the second rule of f & i -Exclusion), we can move the prefix into the matrix, and obtain

$$(\alpha.D_q^f \vee \alpha.D_2^i)$$

Applying the f -Distributivity of \exists and \forall , one can reduce

$$\alpha.D_q^f$$

to formula

$$(\alpha.D_q)^f$$

designated D_1^f . Therefore formula \mathcal{G} is at last reduced to its equivalent prefix normal form

$$D_1^f \vee \alpha.D_2^i$$

Example. Reduce \mathcal{G}

$$\forall x.(D_1^f \vee \neg\forall y\exists z.D_2^i) \wedge (D_3^f \vee \exists u.D_4^i)$$

to prefix normal form.

Following the procedure presented in the proof of Theorem 7:

$$\begin{aligned} \mathcal{G} &\Leftrightarrow \exists u\forall x\exists y\forall z.((D_1^f \wedge D_3^f) \vee (D_1^f \wedge D_4^i) \\ &\quad \vee (\neg D_2^i \wedge D_3^f) \vee (\neg D_2^i \wedge D_4^i)) \\ &\Leftrightarrow \exists u\forall x\exists y\forall z\exists v\exists w.((D_1^f \wedge D_3^f) \vee (D_1^f \wedge D_4^i) \\ &\quad \vee ((\ell = v \Rightarrow \neg D_2^i) \wedge D_3^f) \vee ((\ell = w \Rightarrow \neg D_2^i) \wedge D_4^i)) \\ &\Leftrightarrow \exists u\forall x\exists y\forall z\exists v\exists w.((D_1 \wedge D_3)^f \vee (D_1^f \wedge D_4^i) \\ &\quad \vee ((\ell = v \Rightarrow \neg D_2^i) \wedge D_3^f) \vee ((\ell = w \Rightarrow \neg D_2) \wedge D_4)^i) \\ &\Leftrightarrow \exists u\forall x\exists y\forall z\exists v\exists w.((D_1 \wedge D_3)^f \vee ((\ell = v \Rightarrow \neg D_2) \wedge D_4)^i) \\ &\Leftrightarrow (\forall x.(D_1 \wedge D_3)^f) \vee \exists u\forall x\exists y\forall z\exists w.((\ell = w \Rightarrow \neg D_2) \wedge D_4)^i \\ &\Leftrightarrow (\forall x.(D_1 \wedge D_3))^f \vee \exists u\forall x\exists y\forall z\exists w.((\ell = w \Rightarrow \neg D_2) \wedge D_4)^i \end{aligned}$$

5 DCⁱ Specifications

This section explains how to use DCⁱ to specify duration limits, liveness and fairness of states, and program semantics, in particular the semantics of the sequential composition.

5.1 Limit, Liveness and Fairness

Due to the assumption of the finite variability, a model of DCⁱ can be regarded as a countable sequence of states.

A sequence of states has a limit, iff one of the states appears constantly from some time forever. It can be specified by

$$\exists x.(\ell > x \Rightarrow (\ell = x) \wedge [S]^i)$$

specifies models, which take state S as limit, abbreviated $[S]^\infty$. If S appears everywhere in a model, then the model satisfies

$$[S]^{*i}$$

State S is a limit of a subsequence of a model, iff

$$\forall x \exists y.(y > x \wedge (\ell > y \Rightarrow (\ell = y) \wedge \diamond [S]))^i$$

That is, for any x there exists $y (> x)$ such that S appears right after y . The formula is abbreviated as $[S]^\omega$. $[S]^\omega$ is also a specification of the conventional liveness of state S .

An oscillator with W as its output can be specified by

$$[W]^\omega \wedge [\neg W]^\omega$$

With the inference system of DCⁱ, we can prove

Assertion 1.

1. $[S]^{*i} \Rightarrow [S]^\infty$
2. $[S]^\infty \Rightarrow [S]^\omega$
3. $([S]^\omega \wedge [\neg S]^\omega) \Rightarrow \neg([S]^\infty \vee [\neg S]^\infty)$

Fairness can be regarded as relations between live states. Let S_1 stand for request, and S_2 for response. A *strong* fairness for a request can be specified as

$$[S_1]^\omega \Rightarrow [S_2]^\omega$$

and a *weak* fairness as

$$[S_1]^\infty \Rightarrow [S_2]^\omega$$

Trivially the strong fairness implies the weak fairness.

Assertion 2.

$$([S_1]^\omega \Rightarrow [S_2]^\omega) \Rightarrow ([S_1]^\infty \Rightarrow [S_2]^\omega)$$

A model can also be regarded as a set of Boolean valued functions, which interpret states. Therefore we can investigate limits of state durations over infinite intervals.

That state S has infinite duration over an infinite interval can be specified by

$$\forall x \exists y. (\ell > y \Rightarrow \int S > x)^i$$

abbreviated ($\lim \int S = \infty$). That is, for any x there exists y such that the duration of S in the intervals with length longer than y is greater than x . A state with infinite limit must be live.

That state S takes v as the limit of its duration over an infinite interval⁴ can be specified by

$$\forall \epsilon > 0 \exists y. (\ell > y \Rightarrow |v - \int S| < \epsilon)^i$$

abbreviated ($\lim \int S = v$). The above two formulas are actually translations of Cauchy definitions of limits. We can generalize these two definitions to any term τ of DC, and write them as ($\lim \tau = \infty$) and ($\lim \tau = v$).

⁴[5] investigates duration limits of states in finite intervals, when the states violate the finite variability in the intervals.

A live state with limit v can be specified as

$$\forall \epsilon > 0 \exists y. (\ell > y \Rightarrow 0 < (v - fS) < \epsilon)^i$$

abbreviated $\lim_{\omega} S = v$. Therefore one can measure unbounded liveness by duration limits. For example, for two live states S_1 and S_2 (i.e. $\lceil S_1 \rceil^{\omega} \wedge \lceil S_2 \rceil^{\omega}$), the limit (d) of their duration difference

$$\lim(fS_1 - fS_2) = d$$

could be used to define the *distance* of S_1 from S_2 in a *metric* space of live states. Instead of duration difference, one might prefer to use limit (r) of duration ratio to compare liveness of states.

$$\lim(fS_1 - r \cdot fS_2) = 0$$

Example:

The probabilistic automaton in the Figure has two states S_1 and S_2 . We assume that the state transitions take place randomly after each time unit, according to their probabilities (shown in the Figure). The behaviour of the automaton can be specified in DC^i by

$$\lim S_i = \infty \quad (i = 1, 2)$$

and

$$\lim(fS_1 - 1.5 \cdot fS_2) = 0$$

Namely, each of the two states is live, and has infinite duration. Moreover the liveness ratio between them is 1.5. We believe that extending the Probabilistic Duration Calculus in [13] with infinite intervals can help verify the specifications by showing that the probabilities of the probabilistic automaton satisfying the specifications are equal to 1.

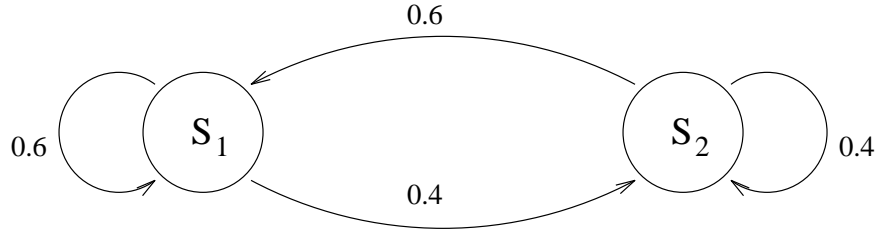


Fig. Probabilistic Automaton

With DCⁱ rules, we can prove

Assertion 3.

1. States with infinite durations are live.

$$(\lim fS = \infty) \Rightarrow \lceil S \rceil^\omega$$

2. $\exists v. (\lim_\omega fS = v) \Rightarrow \lceil S \rceil^\omega$

3. If state S_1 has liveness distance d from state S_2 , and one has finite limit, then so does the other, and the limit has difference d . That is

$$(\lim fS_2 = v \Rightarrow \lim fS_1 = v + d)$$

4. If state S_1 has liveness distance d from state S_2 , and one has infinite duration, then so does the other. That is

$$\bigwedge_{i \neq j} ((\lim fS_i = \infty) \Rightarrow (\lim fS_j = \infty))$$

5. If state S_1 has liveness ratio $r (> 0)$ to state S_2 , and one has finite limit, then so does the other, and the limits have ratio r . That is

$$(\lim fS_2 = v \Rightarrow \lim fS_1 = r \cdot v)$$

6. If state S_1 has liveness ratio $r (> 0)$ to state S_2 , and one has infinite duration, then so does the other. That is

$$\bigwedge_{i \neq j} ((\lim fS_i = \infty) \Rightarrow (\lim fS_j = \infty))$$

With the inference system of DCⁱ, one can also establish a calculus of limits of terms, such as

Assertion 4.

1. $\lim fS = v \Rightarrow \lim f\neg S = \infty$
2. $(\lim fS_1 = v_1 \wedge \lim fS_2 = v_2) \Rightarrow ((\lim f(S_1 \vee S_2) + \lim f(S_1 \wedge S_2)) = (v_1 + v_2))$
3. If $\lim \tau_1 = v_1$ and $\lim \tau_2 = v_2$, then
 - (a) $\lim(\tau_1 + \tau_2) = (v_1 + v_2)$
 - (b) $\lim(\tau_1 - \tau_2) = (v_1 - v_2)$
 - (c) $\lim(\tau_1 \cdot \tau_2) = (v_1 \cdot v_2)$

States can be used to model system properties. For any real valued function F , a *point* property of F , such as

$$F \geq v \text{ and } |F - v| < \epsilon,$$

can be regarded as states. Therefore one can specify *divergence* and *convergence* of functions with DCⁱ. Function F is divergent, designated $\lim F = \infty$, if

$$\forall x \exists y. (\ell > y \Rightarrow (\ell = y) \wedge [F > x])^i$$

and F is convergent to v , designated $\lim F = v$, if

$$\forall \epsilon > 0 \exists x. (\ell > x \Rightarrow (\ell = x) \wedge [|F - v| < \epsilon])^i$$

Similarly we can also derive rules for calculating limits of real valued functions. With limits of real valued functions, one can specify properties of continuous variables of control systems, such as system stability [12]. For example, let c stand for the output function of a controlled system, and the *asymptotic* stability of the system can be specified by

$$\exists x. [|c| \leq x]^{*i} \wedge \lim c = 0$$

That is, c is bounded, and the magnitude of c reaches 0 as time approaches ∞ . Let r stand for the input of the system. The *bounded-input bounded-output* stability can be specified by

$$\exists x. [|r| \leq x]^{*i} \Rightarrow \exists x. [|c| \leq x]^{*i}$$

5.2 Program Semantics

A real time semantics of an Occam-like language has been defined in DC [33]. Lacking infinite intervals, the semantics denotes behaviour of communicating processes by all prefixes of the behaviour. The parallel operator (\parallel) can be defined by conjunction of the parallel processes. However the sequential operator ($;$) is defined indirectly by employing the notion of program continuation. With both finite and infinite intervals, DCⁱ is able to improve the semantic definitions given in [33].

Let $c!$ and $c?$ stand for the output and input commands for channel c . With no confusion, we also use $c!$ and $c?$ to designate the states where output to channel c and input from channel c are ready. With DCⁱ, we can define the semantics of commands $c!$ and $c?$ of a process as

$$\begin{aligned} \llbracket c! \rrbracket &\hat{=} ([c! \wedge \neg c?]^* \frown [c! \wedge c?])^f \vee [c! \wedge \neg c?]^*{}^i \\ \llbracket c? \rrbracket &\hat{=} ([c? \wedge \neg c!]^* \frown [c? \wedge c!])^f \vee [c? \wedge \neg c!]^*{}^i \end{aligned}$$

The first formula defines the semantics of command $c!$ by specifying that, when $c!$ is executed, the output partner becomes ready to output (i.e. $[c!]$), and wait synchronization from the input counterpart (i.e. $[c?]$) either forever (i.e. $[c! \wedge \neg c?]^*{}^i$), if the communication fails, or for finite time, if the communication succeeds (i.e. $([c! \wedge \neg c?]^* \frown [c! \wedge c?])^f$). The second formula defines the semantics of command $c?$ in a symmetric way. In the semantics, we disregard the behaviour of the process on other channels. Please refer to [33] for a complete description.

In order to define the semantics of sequential operator with DCⁱ, we first introduce a correspondent operator (designated also as $;$) in DCⁱ. By Theorem 7 in Section 4, any DCⁱ formula \mathcal{G} can be reduced to its prefix normal form

$$D_1^f \vee \alpha.D_2^i$$

where α stand for a prefix. Therefore

$$\begin{aligned} (\mathcal{G} \wedge fin) &\Leftrightarrow D_1^f \\ (\mathcal{G} \wedge inf) &\Leftrightarrow \alpha.D_2^i \end{aligned}$$

Thus, without loss of generality, the definition of $;$ can be given as follows. For DC formulas D , C_1 and C_2 , prefix α , and DCⁱ formulas \mathcal{G}_1 and \mathcal{G}_2 , let

$$\begin{aligned} D^f; (C_1^f \vee \alpha.C_2^i) &\hat{=} (D \frown C_1)^f \vee \exists x \alpha. (\ell \geq x \Rightarrow (D \wedge (\ell = x)) \frown C_2)^i \\ \mathcal{G}_1; \mathcal{G}_2 &\hat{=} (fin \wedge \mathcal{G}_1); \mathcal{G}_2 \vee (inf \wedge \mathcal{G}_1) \end{aligned}$$

By the definition of $;$, a finite behaviour can be sequentially extended by either a finite behaviour or an infinite one. The former is simply defined by \wedge (i.e. $(D \wedge C_1)^f$), and the latter one by a behaviour, a prefix of which is determined by the extended finite behaviour and the rest part by the extending infinite one (i.e. $\exists x \alpha. (\ell \geq x \Rightarrow (D \wedge (\ell = x)) \wedge C_2)^i$). However, any infinite behaviour ($inf \wedge \mathcal{G}_1$) cannot be sequentially extended.

A sequential composition of programs can be defined straightforwardly now.

$$\llbracket \mathcal{S}_1; \mathcal{S}_2 \rrbracket \hat{=} (\llbracket \mathcal{S}_1 \rrbracket; \llbracket \mathcal{S}_2 \rrbracket)$$

where \mathcal{S}_1 and \mathcal{S}_2 stand for two programs.

A semantics of the parallel operator can be defined with DCⁱ in a way similar to [33]. Let \mathcal{S}_1 and \mathcal{S}_2 stand for two programs and σ_1 and σ_2 for their *alphabets* (namely, the input and output commands occurring in \mathcal{S}_1 and \mathcal{S}_2) respectively. For $j = 1, 2$, let

$$\neg\sigma_j \hat{=} \left(\bigwedge_{c! \in \sigma_j} \neg c! \right) \wedge \left(\bigwedge_{c? \in \sigma_j} \neg c? \right)$$

In fact, $\neg\sigma_j$ specifies the *inactive* state of \mathcal{S}_j . We use it to define the state of \mathcal{S}_j after termination. Therefore

$$(fin \wedge \llbracket \mathcal{S}_1 \rrbracket) \wedge ((fin \wedge \llbracket \mathcal{S}_2 \rrbracket); \lceil \neg\sigma_2 \rceil^{*f})$$

specifies the parallel result of finite behaviour of the two programs, where \mathcal{S}_2 may terminate before \mathcal{S}_1 does. Symmetrically

$$((fin \wedge \llbracket \mathcal{S}_1 \rrbracket); \lceil \neg\sigma_1 \rceil^{*f}) \wedge (fin \wedge \llbracket \mathcal{S}_2 \rrbracket)$$

specifies the parallel result of finite behaviour of the two programs, where \mathcal{S}_1 may terminate before \mathcal{S}_2 does. Furthermore

$$(inf \wedge \llbracket \mathcal{S}_1 \rrbracket) \wedge ((fin \wedge \llbracket \mathcal{S}_2 \rrbracket); \lceil \neg\sigma_2 \rceil^{*i})$$

specifies the parallel result of infinite behaviour of \mathcal{S}_1 and finite behaviour of \mathcal{S}_2 . Symmetrically we can specify the result of finite behaviour of \mathcal{S}_1 and infinite behaviour of \mathcal{S}_2 by

$$((fin \wedge \llbracket \mathcal{S}_1 \rrbracket); \lceil \neg\sigma_1 \rceil^{*i}) \wedge (inf \wedge \llbracket \mathcal{S}_2 \rrbracket)$$

The last specification is for the parallel result of two infinite behaviours of the two programs. That is

$$(inf \wedge \llbracket \mathcal{S}_1 \rrbracket) \wedge (inf \wedge \llbracket \mathcal{S}_2 \rrbracket)$$

A disjunction of all previous five formulas defines a semantics of the parallel operator:

$$\llbracket \mathcal{S}_1 \parallel \mathcal{S}_2 \rrbracket$$

We can therefore derive from the semantics of the parallel operator

$$\vdash_i \llbracket c! \parallel c? \rrbracket \Leftrightarrow \lceil c! \wedge c? \rceil^f$$

With the semantics defined above, program termination can be verified by proving

$$\vdash_i \llbracket \mathcal{S} \rrbracket \Rightarrow fin$$

where \mathcal{S} stands for the verified program. For instance, we can prove the termination of $(c! \parallel c?)$, since

$$\vdash_i \lceil c! \wedge c? \rceil^f \Rightarrow fin$$

6 Discussion

1. [32] and [36] have tried to formalize *integrals*, *mean values* and *function germs* of Boolean valued functions. [5] and DC^i tries to formalize a notion of limit. It has been shown that introducing some of continuous mathematics into design calculi can assist the formal technique of programming in specifying and designing computing systems.
2. By introducing limits, DC^i can deal with unbounded liveness and fairness, and can also measure live states by duration limits. One can develop theory of such measurements, and consider interesting applications of it.
3. DC^i introduces infinite intervals by establishing a kind of metalogic of DC (a logic of a metatheory of DC). In the same way, one can introduce infinite intervals into Interval Temporal Logic. We believe that DC^i has paid the least cost to introduce infinite intervals, since, in order to formalize mathematical definition of limits, quantifications is unavoidable.
4. Although the inference system of DC^i seems powerful, we cannot conclude its (relative) completeness in the paper.

References

- [1] Dang Van Hung and Zhou Chaochen: Probabilistic Duration Calculus for Continuous Time. *UNU/IIST Report No. 25*, 1994.
- [2] M. Engel, M. Kubica, J. Madey, D.J. Parnas, A.P. Ravn and A.J. van Schouwen: A Formal Approach to Computer Systems Requirements Documentation. In *Proc. the Workshop on Theory of Hybrid Systems*, LNCS 736, R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel (Editors), pp. 452-474, 1993.
- [3] M. Engel and H. Rischel: Dagstuhl-Seminar Specification Problem – a Duration Calculus Solution. Personal communication, September 1994.
- [4] M.R. Hansen: Model-Checking Discrete Duration Calculus. In *Formal Aspects of Computing*. Vol. 6, No. 6A, pp. 826-845,. 1994.
- [5] M.R. Hansen, P.K. Pandya and Zhou Chaochen: Finite Divergence. In *Theoretical Computer Science*, Vol.138, pp 113-139, 1995.
- [6] M.R. Hansen and Zhou Chaochen: Semantics and Completeness of Duration Calculus. In *Real-Time: Theory in Practice, REX Workshop*, LNCS 600, J.W. de Bakker, C. Huizing, W.-P. de Roever and G. Rozenberg (Editors), pp. 209-225, 1992.
- [7] M.R. Hansen, Zhou Chaochen and J. Staunstrup: A Real-Time Duration Semantics for Circuits. In *Proc. of the 1992 ACM/SIGDA Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Princeton, March 1992.
- [8] He Jifeng and J. Bowen: Time Interval Semantics and Implementation of A Real-Time Programming Language. In *Proc. 4th Euromicro Workshop on Real Time Systems*, IEEE Press, June 1992.
- [9] He Weidong and Zhou Chaochen: A Case Study of Optimization. *UNU/IIST Report No. 34*, December 1994.
- [10] C.A.R. Hoare: *Communicating Sequential Processes*. Prentice Hall International (UK) Ltd., 1985.
- [11] Y. Kesten, A. Pnueli, J. Sifakis and S. Yovine: Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid Systems*, LNCS 736, R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel (Editors), pp. 179-208, 1993.
- [12] B.C. Kuo: *Automatic Control Systems* (sixth edition), Prentice-Hall International Inc., 1991.
- [13] Liu Zhiming, A.P. Ravn, E.V. Sørensen and Zhou Chaochen: A Probabilistic Duration Calculus. In, *Dependable Computing and Fault-Tolerant Systems Vol. 7: Responsive Computer Systems*. H. Kopetz and Y. Kakuda (Editor), pp. 30-52, Springer Verlag, 1993.

- [14] Liu Zhiming, A.P. Ravn, E.V. Sørensen and Zhou Chaochen: Towards a Calculus of Systems Dependability. In *Journal of High Integrity System*, Vol. 1, No. 1, Oxford University Press, pp. 49-65, 1994.
- [15] B. Moszkowski: A Temporal Logic for Multilevel Reasoning about Hardware. In *IEEE Computer*, Vol. 18, No. 2, pp. 10-19, 1985.
- [16] B. Moszkowski: Some Very Compositional Temporal Properties, In *Programming Concepts, Methods and Calculi (A-56)*, E.-R. Olderog (Editor), Elsevier Science B.V. (North-Holland), pp. 307-326, 1994.
- [17] P.H. Pandya: Weak Chop Inverses and Liveness in Duration Calculus. *Technical Report TR-95-1*, Computer Science Group, TIFR, India, 1994.
- [18] A.P. Ravn and H. Rischel: Requirements Capture for Embedded Real-Time Systems. In *Proc. IMACS-MCTS'91 Symp. Modelling and Control of Technological Systems*, Vol. 2, pp. 147-152, Villeneuve d'Ascq, France, 1991.
- [19] A.P. Ravn, H. Rischel and K.M. Hansen: Specifying and Verifying Requirements of Real-Time Systems. In *IEEE Trans. Software Eng.*, Vol. 19, No. 1, pp. 41-55, January 1993.
- [20] R. Rosner and A. Pnueli: A Choppy Logic. In *First Annual IEEE Symposium on Logic In Computer Science*, pp 306-314, IEEE Computer Society Press, June, 1986.
- [21] J.U. Skakkebæk: Liveness and Fairness in Duration Calculus. In *CONCUR'94: Concurrency Theory*, LNCS 836, B. Jonsson and J. Parrow(Editors), pp. 283-298, 1994.
- [22] J.U. Skakkebæk, A.P. Ravn, H. Rischel and Zhou Chaochen: Specification of Embedded Real-Time Systems. In *Proc. 4th Euromicro Workshop on Real-Time Systems*, pp. 116-121, IEEE Press, June 1992.
- [23] J.U. Skakkebæk and N. Shankar: Towards a Duration Calculus Proof Assistant in PVS. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, H. Langmaack, W.-P. de Roever and J. Vytupil (Editors), pp. 660-679, Sept. 1994.
- [24] J.U. Skakkebæk, and P. Sestoft: Checking Validity of Duration Calculus Formulas. *ProCoS II Report ID/DTH JUS 3/1*, January 1993
- [25] Y. Venema: A Modal Logic for Chopping Intervals. In *Journal of Logic Computation*, Vol. 1, No. 4, pp. 453-476, 1991.
- [26] H. Weyl: Mathematics and Logic. A Brief Survey Serving as a Preface to a View of "The Philosophy of Bertrand Russell". In *Amer. Math. Monthly*, Vol. 53, pp. 2-13, 1946.
- [27] B.H. Widjaja, Chen Zongji, He Weidong and Zhou Chaochen: A Cooperative Design for Hybrid Control System. *UNU/IIST Report No.36*, 1995.

- [28] Yu Huiqun, P.K. Pandya and Sun Yongqiang: A Calculus for Hybrid Sampled Data Systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, H. Langmaack, W.-P. de Roever and J. Vytupil (Editors), pp. 716-737, Sept. 1994.
- [29] Yu Xinyao, Wang Ji, Zhou Chaochen and P.K. Pandya: Formal Design of Hybrid Systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, H. Langmaack, W.-P. de Roever and J. Vytupil (Editors), pp. 738-755, Sept. 1994.
- [30] Zheng Yuhua and Zhou Chaochen: A Formal Proof of the Deadline Driven Scheduler. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, H. Langmaack, W.-P. de Roever and J. Vytupil (Editors), pp. 756-775, Sept. 1994.
- [31] Zhou Chaochen: Duration Calculi: An Overview. In *the Proceedings of Formal Methods in Programming and Their Applications*, LNCS 735, D. Bjørner, M. Broy and I.V. Pottosin (Editors), pp. 256-266, July 1993.
- [32] Zhou Chaochen, C.A.R. Hoare and A.P. Ravn: A Calculus of Durations. In *Information Processing Letters*, Vol. 40, No. 5, pp. 269-276, 1991.
- [33] Zhou Chaochen, M.R. Hansen, A.P. Ravn and H. Rischel: Duration Specifications for Shared Processors. In *Proc. of the Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 571, J. Vytupil (Editor), pp. 21-32, January 1992.
- [34] Zhou Chaochen, M.R. Hansen and P. Sestoft: Decidability and Undecidability Results for Duration Calculus. In *Proc. of STACS '93. 10th Symposium on Theoretical Aspects of Computer Science*, LNCS 665, P. Enjalbert, A. Finkel and K.W. Wagner (Editor), pp. 58-68, Feb. 1993.
- [35] Zhou Chaochen and Li Xiaoshan: Infinite Duration Calculus. Draft, August 1992.
- [36] Zhou Chaochen and Li Xiaoshan: A Mean Value Calculus of Durations. In *A Classical Mind (Essays in Honour of C.A.R. Hoare)*, A.W.Roscoe (Editor), Prentice-Hall, pp. 431-451, 1994.
- [37] Zhou Chaochen, A.P. Ravn and M.R. Hansen: An Extended Duration Calculus for Hybrid Real-Time Systems. In *Hybrid Systems*, LNCS 736, R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel (Editors), pp. 36-59, 1993.
- [38] Zhou Chaochen, Zhang Jingzhong, Yang Lu and Li Xiaoshan: Linear Duration Invariants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, H. Langmaack, W.-P. de Roever and J. Vytupil (Editors), pp. 86-109, Sept. 1994.