28

A Combination of Interval Logic and Linear Temporal Logic

*Z. Qiu**
Department of Informatics, School of Mathematics, Peking University, Beijing 100871, China. E-Mail: zyqiu@pku.edu.cn

*C. Zhou***
International Institute for Software Technology, The United Nations University, P.O.Box 3058, Macau. E-Mail: zcc@iist.unu.edu

Abstract

The *super-dense computation* model provides an abstraction of real-time behaviour of computing systems. We present a combination of a linear temporal logic and an interval logic that uses super-dense computation and demonstrate how it can be used to specify real-time semantics and real-time properties of an OCCAM-like programming language.

Keywords

Temporal Logic, Interval Logic, Duration Calculus, Real-time semantics and specifications, Super-dense computation, OCCAM

1 INTRODUCTION

A *super-dense behaviour* consists of a series of continuously evolving phases in which a sequence of discrete actions can take place at the juncture of any two successive phases. The discrete actions are assumed to be instantaneous (they take no time), but ordered. Time, as a sequence of real numbers, is dense, and the assumption that ordered action sequences can happen at time points introduces additional structure into the already dense time domain. Thus the name *super-dense.*

The super-dense computation model is an important abstract model for real-time systems [7]. Many articles deal with this model [5, 7, 8, 16], and some industrially applicable programming languages (e.g. Esterel [1]) adopt similar models.

*The work was done when this author visited UNU/IIST.
**On leave of absence from Software Institute, Beijing.

Programming Concepts and Methods D. Gries, & W-P. de Roever (Eds.)

Control systems can be used to explain the concept of a super-dense model. The main part of a process in a control system can be expressed as:

while TRUE **do**
$wait(SAMPLING\_TIME)$;
$x :=$ $get\_sample(sensor)$;
$y :=$ $calculate\_output(x)$;
$send\_output(actuator, y)$

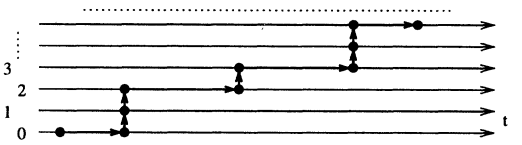
Execution of the process consists of continuous evolution (waiting between samplings) and discrete actions (computer calculations). The interaction frequency of the process with its environment is measured in seconds, while the discrete actions are performed in a computer with speed counted in nanoseconds. Thus, there is a 10^6–10^9 gap between sampling and calculation speeds. Control engineers therefore prefer to assume that the calculations take no time, and the super-dense model applies to this class of systems.

The super-dense model is reasonable and can make analyses of system behaviour more tractable under two conditions: (1) No infinite sequence of discrete actions takes place at a time point; and (2) the time consumed by each discrete action sequence is negligible in comparison to the continuous phases.

The literature contains several proposals to describe super-dense model mathematically. Some researchers use a timed sequence of actions to describe the behaviour of a real-time system (e.g. [7]). Here, a timed action is represented as a pair $\langle a, t\rangle$, where a is the action and t the time consumed by a. Thus, timeless discrete actions are special cases of pairs with $t = 0$. Article [12] adopts a similar view and suggests combining TLA [4] with the duration calculus [14] in describing the super-dense model.

Article [16] formalizes the intuition of two different levels of time measurement as a modal logic. By a modality named *super-dense chop*, a point in the *coarse* time space can be chopped into a sequence of points in the *fine* time space. The *intervals* in the fine time space are invisible at the coarse level, which achieves the hiding of internal states. However, because of the hiding, different action sequences become indistinguishable, causing inconvenience for describing the behaviour of concurrent systems with shared variables.

We take a view similar to that of [2]. We view a process under the super-dense model as a trajectory in a two-dimensional time space and establish a logic to describe and reason about it. We assume there are countably infinite time axes, which are indexed by natural numbers. A process starts at some time on a time axis. When the process executes time-consuming actions, time progresses horizontally and the process stays on the same axis. When the process executes a timeless action, it jumps to another axis and changes to a new state. These jumps are instantaneous, and they can happen sequentially when necessary. Thus, real-time behaviour is divided into two classes as far as time is concerned: staying on an axis (when time evolves) and jumping instantaneously to another axis. A trajectory of a super-dense behaviour is shown in the following figure:



For this two-dimensional time space, we need a logic for a single axis, where continuous time is assumed, and we need modalities to travel among axes. We use an interval logic for behaviours on a single axis and a linear temporal logic for traveling between axes. In the following sections, we define the combined logic and then use it to define the semantics of a real-time OCCAM-like programming language. We close with a discussion of future work.

2 TWO-DIMENSIONAL TEMPORAL LOGIC

We present the syntax and semantics of the logic, give an inference system, and prove a number of theorems.

2.1 Syntax

(a) Symbols

Symbols of the logic include those presented below. Note that there are two kinds of variables: a global variable has a fixed value on all time axes over all time intervals, while a temporal variable represents a function of intervals on each time axis.

- Global (or rigid) variables: $Gvar.\ u, u_1, \ldots \in Gvar.$
- Temporal variables: $Tvar.\ x, y, x_1, \ldots \in Tvar.$
- Interval length: ℓ, which is a specific temporal variable.
- Function symbols: $f, g, f_1, \ldots$, including $0, 1, +, -, \ldots$.
- Relation symbols: $F, G, F_1, \ldots$, including $\mathrm{tt}, \mathrm{ff}, =, <, \ldots$.

(b) Terms

In the presentation of the form of terms below, $\theta, \theta_1, \theta_2, \ldots$ represent arbitrary terms, f is an n-ary function symbol, and $\bigcirc$ is the next moment operator from linear temporal logic [6]:

$$\theta ::= u \mid x \mid \ell \mid f(\theta_1, \ldots, \theta_n) \mid \bigcirc \theta$$

(c) Formulas

We use $\mathcal{D}$ to denote the set of formulas of the logic and $\varphi, \psi, \varphi_1, \varphi_2, \ldots \in \mathcal{D}$ to represent arbitrary formulas. In the following formation rules for formulas, F is an n-ary predicate symbol, $\bigcirc$ is the *next* modality and $\mathcal{U}$ the *until* modality

(from linear temporal logic [6]), and $\Diamond_l$ and $\Diamond_r$ are two interval modalities (from neighbourhood logic [15]).

$$\varphi ::= F(\theta_1, \ldots \theta_n) \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists u.\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathcal{U} \varphi_2 \mid \Diamond_l\varphi \mid \Diamond_r\varphi$$

2.2 Semantics

Our semantics assumes countably infinite time axes. Interval modalities $\Diamond_l$ and $\Diamond_r$ define accessibility of left or right neighbourhoods of the reference interval on a single axis, while temporal modalities ($\bigcirc$ and $\mathcal{U}$) define accessibility between axes. A model of the logic contains three components: a valuation, an interpretation and a reference interval. Throughout, $\mathbb{R}$ denotes the set of real numbers and $\mathbb{N}$ the set of natural numbers.

- A *valuation* $\mathcal{V}$ assigns values to all global variables:

  $$\mathcal{V} \in Gvar \to \mathbb{R}$$

- An *interval* $[b, e]$, where $(b, e \in \mathbb{R}) \wedge b \leq e$, is the set $\{x \mid b \leq x \leq e\}$. We use $Intv$ to denote the set of all intervals.
- An *interpretation* $\mathcal{I}$ gives meaning to every temporal variable on each time axis as a function from intervals to real numbers:

  $$\mathcal{I} \in \mathbb{N} \to (Tvar \to (Intv \to \mathbb{R}))$$

  Here, $\mathbb{N}$ serves as the indices of axes. An interpretation $\mathcal{I}$ can be written as an infinite sequence

  $$\mathcal{I} = \sigma_0, \sigma_1, \sigma_2, \ldots$$

  where each $\sigma_i = \mathcal{I}(i), i = 0, 1, \ldots$, assigns every temporal variable an interval function on axis i. We use $\mathcal{I}_k$ to denote the subsequence of $\mathcal{I}$ starting at σ_k: $\mathcal{I}_k \mathrel{\widehat{=}} \sigma_k, \sigma_{k+1}, \sigma_{k+2}, \ldots$. Obviously, $\mathcal{I}_k(i) = \mathcal{I}(k + i)$.

A model $\mathcal{M}$ of the logic is a triple

$$\mathcal{M} = (\mathcal{I}, \mathcal{V}, [b, e])$$

where $\mathcal{V}, \mathcal{I}$ are an arbitrary valuation and interpretation, and $[b, e]$ is an interval. In the following definitions, let: $\mathcal{M}_0 = \mathcal{M}$ and $\mathcal{M}_k = (\mathcal{I}_k, \mathcal{V}, [b, e])$. Also, $\underline{f}$ and $\underline{F}$ denote function and predicate in real arithmetic and represent the

standard meaning of f and F. The meaning of terms and formulas are defined as follows:

(a) For model $\mathcal{M}$, the meaning $\mathcal{M}(\theta)$ of a term θ is a real number defined by: $\mathcal{M}(\theta) \mathrel{\widehat{=}} \mathcal{M}_0(\theta) \in \mathbb{R}$, and $\mathcal{M}_i(\theta)$ is defined as:
1) $\mathcal{M}_i(u) \mathrel{\widehat{=}} \mathcal{V}(u)$.
2) $\mathcal{M}_i(\ell) \mathrel{\widehat{=}} e - b$.
3) $\mathcal{M}_i(x) \mathrel{\widehat{=}} \mathcal{I}(i)(x)([b, e])$.
4) $\mathcal{M}_i(f(\theta_1, \ldots, \theta_n)) \mathrel{\widehat{=}} \underline{f}(\mathcal{M}_i(\theta_1), \ldots, \mathcal{M}_i(\theta_n))$.
5) $\mathcal{M}_i(\bigcirc\theta) \mathrel{\widehat{=}} \mathcal{M}_{i+1}(\theta)$.

(b) Given $\mathcal{M}$, the meaning $\mathcal{M}(\varphi)$ of formula φ is one of the truth values tt and ff. We define $\mathcal{M}(\varphi) \mathrel{\widehat{=}} \mathcal{M}_0(\varphi) \in \{\mathrm{tt}, \mathrm{ff}\}$, where $\mathcal{M}_i(\varphi)$ is defined as:
1) $\mathcal{M}_i(F(\theta_1, \ldots, \theta_n)) \mathrel{\widehat{=}} \underline{F}(\mathcal{M}_i(\theta_1), \ldots, \mathcal{M}_i(\theta_n))$.
2) $\mathcal{M}_i(\neg\varphi) \mathrel{\widehat{=}} \mathrm{tt}$ iff $\mathcal{M}_i(\varphi) = \mathrm{ff}$.
3) $\mathcal{M}_i(\varphi_1 \vee \varphi_2) \mathrel{\widehat{=}} \mathrm{tt}$ iff $\mathcal{M}_i(\varphi_1) = \mathrm{tt}$ or $\mathcal{M}_i(\varphi_2) = \mathrm{tt}$.
4) $\mathcal{M}_i(\exists u.\varphi) \mathrel{\widehat{=}} \mathrm{tt}$ iff there exists $\mathcal{V}'$, $\mathcal{V}'$ u-equivalent to $\mathcal{V}$, such that $\mathcal{M}_i[\mathcal{V}'/\mathcal{V}](\varphi) = \mathrm{tt}$. By u-equivalent we mean that $\mathcal{V}'$ differs from $\mathcal{V}$ by at most the value assigned to u. $\mathcal{M}_i[\mathcal{V}'/\mathcal{V}]$ denotes the model formed from $\mathcal{M}_i$ by replacing the component $\mathcal{V}$ by $\mathcal{V}'$.
5) $\mathcal{M}_i(\bigcirc\varphi) \mathrel{\widehat{=}} \mathrm{tt}$ iff $\mathcal{M}_{i+1}(\varphi) = \mathrm{tt}$.
6) $\mathcal{M}_i(\varphi_1 \,\mathcal{U}\, \varphi_2) \mathrel{\widehat{=}} \mathrm{tt}$ iff there exists j, $i \leq j$, such that $\mathcal{M}_j(\varphi_2) = \mathrm{tt}$ and for every k, $i \leq k < j$, $\mathcal{M}_k(\varphi_1) = \mathrm{tt}$.
7) $\mathcal{M}_i(\diamond_l\varphi) \mathrel{\widehat{=}} \mathrm{tt}$ iff there exists a $\delta \geq 0$ such that $\mathcal{M}_i[(b-\delta)/b, b/e](\varphi) = \mathrm{tt}$. Here we use substitution to form a new model from the original one with a revised reference interval.
8) $\mathcal{M}_i(\diamond_r\varphi) \mathrel{\widehat{=}} \mathrm{tt}$ iff there exists a $\delta \geq 0$ such that $\mathcal{M}_i[e/b, (e+\delta)/e](\varphi) = \mathrm{tt}$.

From the definitions in (a) and (b), we see that the meanings of $\bigcirc$ and $\mathcal{U}$ correspond to their meanings in linear temporal logic: $\bigcirc$ accesses the same reference interval but on the next axis, and $\mathcal{U}$ accesses a sequence of intervals above the reference interval. Operator $\diamond_l$ (and $\diamond_r$) accesses a left (right) neighbourhood of the beginning (end) point of the reference interval. Thus, the combination $\diamond_l\diamond_l$ (and $\diamond_r\diamond_r$) can be used to access an interval to the left (right) of the reference interval, while $\diamond_l\diamond_r$ (and $\diamond_r\diamond_l$) can be used to access a right (and left) neighbourhood of the beginning (end) point of the reference interval.

We say that $\mathcal{M}$ is a *model* of formula φ, written $\mathcal{M} \models \varphi$, iff $\mathcal{M}(\varphi) = \mathrm{tt}$. Similarly, $\mathcal{M} \not\models \varphi$ is written for $\mathcal{M}(\varphi) = \mathrm{ff}$. A formula φ is called *satisfiable* iff $\mathcal{M} \models \varphi$ for some model $\mathcal{M}$, and φ is called *valid* iff $\mathcal{M} \models \varphi$ for all models $\mathcal{M}$.

Other modalities of linear temporal logic and interval logic can be defined. The $\diamond$ and $\Box$ modalities of linear temporal logic are defined as usual:

$$\diamond\varphi \mathrel{\widehat{=}} \mathrm{tt}\,\mathcal{U}\,\varphi \qquad \Box\varphi \mathrel{\widehat{=}} \neg\diamond\neg\varphi$$

The chop modality $\frown$ of interval logic is defined as:

$$\varphi_1 \frown \varphi_2 \mathrel{\widehat{=}} \exists u_1, u_2.((\ell = u_1 + u_2) \wedge \Diamond_l \Diamond_r (\ell = u_1 \wedge \varphi_1 \wedge \Diamond_r (\ell = u_2 \wedge \varphi_2)))$$

$\varphi_1 \frown \varphi_2$ holds for a reference interval with length ℓ iff this interval can be chopped into two adjacent parts, where φ_1 holds over the left part and φ_2 holds over the right part.

We also have the duals of $\Diamond_l$ and $\Diamond_r$. These modalities are used to express properties that are true over all the left and over all the right neighbourhoods:

$$\Box_l \varphi \mathrel{\widehat{=}} \neg \Diamond_l \neg \varphi \qquad \Box_r \varphi \mathrel{\widehat{=}} \neg \Diamond_r \neg \varphi$$

2.3 Inference System

We now present the axioms and inference rules of the logic. First of all, the axioms of the linear temporal logic and the neighbourhood logic are still valid. For a complete set of axioms to formalize the class of our model, the following additional axioms are necessary.

1. $\bigcirc$ retains the length of the reference interval.
   (LT1) $(\ell = u) \Leftrightarrow \bigcirc(\ell = u)$
   From this axiom and the inference system of linear temporal logic, we can prove theorems like: $(\ell = u) \Leftrightarrow \Diamond(\ell = u)$, $(\ell = u) \Leftrightarrow \Box(\ell = u)$.
2. Commutativity of the *next* modality and the neighbourhood modalities.
   (NN1) $\Diamond_l \bigcirc \varphi \Leftrightarrow \bigcirc \Diamond_l \varphi$
   (NN2) $\Diamond_r \bigcirc \varphi \Leftrightarrow \bigcirc \Diamond_r \varphi$
   To access a neighbourhood on next axis, we can either reach a neighbourhood on the current axis and then go up to the next axis or do it the other way round.
3. Distributivity of neighbourhood modalities over the *until* modality:
   (UN1) $\Diamond_l(\varphi_1 \,\mathcal{U}\, \varphi_2) \Rightarrow (\Diamond_l \varphi_1) \,\mathcal{U}\, (\Diamond_l \varphi_2)$
   (UN2) $\Diamond_r(\varphi_1 \,\mathcal{U}\, \varphi_2) \Rightarrow (\Diamond_r \varphi_1) \,\mathcal{U}\, (\Diamond_r \varphi_2)$
   The other direction of the implications are valid under certain conditions on interval length:
   (UN3) $(\Diamond_l(\ell = u \wedge \varphi_1)) \,\mathcal{U}\, (\Diamond_l(\ell = u \wedge \varphi_2)) \Rightarrow \Diamond_l(\ell = u \wedge (\varphi_1 \,\mathcal{U}\, \varphi_2))$
   (UN4) $(\Diamond_r(\ell = u \wedge \varphi_1)) \,\mathcal{U}\, (\Diamond_r(\ell = u \wedge \varphi_2)) \Rightarrow \Diamond_r(\ell = u \wedge (\varphi_1 \,\mathcal{U}\, \varphi_2))$
4. The complementary axioms of (UN1) and (UN2):
   (UN5) $(\Box_l \varphi_1) \,\mathcal{U}\, (\Box_l \varphi_2) \Rightarrow \Box_l(\varphi_1 \,\mathcal{U}\, \varphi_2)$
   (UN6) $(\Box_r \varphi_1) \,\mathcal{U}\, (\Box_r \varphi_2) \Rightarrow \Box_r(\varphi_1 \,\mathcal{U}\, \varphi_2)$
   The converse implications are not true.

Soundness of these axioms is easy to check semantically. However, we do not know whether they form a complete set.

2.4 Theorems

With the axioms described above, we can prove the theorems listed below.

- Commutativity of $\bigcirc$ and neighbourhood with length:

$$\Diamond_l(\ell = u \wedge \bigcirc\varphi) \Leftrightarrow \bigcirc\Diamond_l(\ell = u \wedge \varphi)$$
$$\Diamond_r(\ell = u \wedge \bigcirc\varphi) \Leftrightarrow \bigcirc\Diamond_r(\ell = u \wedge \varphi)$$

The proofs are simple. We give one only for $\Diamond_l$.

$$\begin{aligned}\Diamond_l(\ell = u \wedge \bigcirc\varphi) &\Leftrightarrow \Diamond_l(\bigcirc(\ell = u) \wedge \bigcirc\varphi) \\ &\Leftrightarrow \Diamond_l \bigcirc (\ell = u \wedge \varphi) \quad \Leftrightarrow \bigcirc\Diamond_l(\ell = u \wedge \varphi)\end{aligned}$$

- $\bigcirc$ is distributive over chop.

$$\bigcirc(\varphi_1 \frown \varphi_2) \Leftrightarrow \bigcirc\varphi_1 \frown \bigcirc\varphi_2$$

Here is a proof:

$$\begin{aligned}&\bigcirc(\varphi_1 \frown \varphi_2) \\ \Leftrightarrow\ &\bigcirc\exists u_1, u_2.((\ell = u_1 + u_2) \wedge \Diamond_l\Diamond_r(\ell = u_1 \wedge \varphi_1) \wedge \Diamond_r\Diamond_l(\ell = u_2 \wedge \varphi_2)) \\ \Leftrightarrow\ &\exists u_1, u_2.\bigcirc((\ell = u_1 + u_2) \wedge \Diamond_l\Diamond_r(\ell = u_1 \wedge \varphi_1) \wedge \Diamond_r\Diamond_l(\ell = u_2 \wedge \varphi_2)) \\ \Leftrightarrow\ &\exists u_1, u_2.(\bigcirc(\ell = u_1 + u_2) \wedge \bigcirc\Diamond_l\Diamond_r(\ell = u_1 \wedge \varphi_1) \wedge \bigcirc\Diamond_r\Diamond_l(\ell = u_2 \wedge \varphi_2)) \\ \Leftrightarrow\ &\exists u_1, u_2.((\ell = u_1 + u_2) \wedge \Diamond_l\Diamond_r(\ell = u_1 \wedge \bigcirc\varphi_1) \wedge \Diamond_r\Diamond_l(\ell = u_2 \wedge \bigcirc\varphi_2)) \\ \Leftrightarrow\ &\bigcirc\varphi_1 \frown \bigcirc\varphi_2\end{aligned}$$

- Commutativity of *eventual* modality $\Diamond$ and neighbourhood modalities:

$$\Diamond_l\Diamond\varphi \Leftrightarrow \Diamond\Diamond_l\varphi \qquad \Diamond_r\Diamond\varphi \Leftrightarrow \Diamond\Diamond_r\varphi$$

An axiom of the neighbourhood logic is:

$$\Diamond_l\varphi \Rightarrow \exists u \geq 0.\Diamond_l(\ell = u \wedge \varphi)$$

Thus:

$$\begin{aligned}\Diamond\Diamond_l\varphi &\Rightarrow \Diamond(\exists u \geq 0.\Diamond_l(\ell = u \wedge \varphi)) \\ &\Rightarrow \exists u \geq 0.\Diamond\Diamond_l(\ell = u \wedge \varphi) \\ &\Leftrightarrow \exists u \geq 0.(\mathrm{tt}\ \mathcal{U}\ \Diamond_l(\ell = u \wedge \varphi)) \\ &\Leftrightarrow \exists u \geq 0.((\Diamond_l(\ell = u \wedge \mathrm{tt}))\ \mathcal{U}\ (\Diamond_l(\ell = u \wedge \varphi))) \\ &\Rightarrow \exists u \geq 0.\Diamond_l(\ell = u \wedge (\mathrm{tt}\ \mathcal{U}\ \varphi)) \\ &\Leftrightarrow \exists u \geq 0.\Diamond_l(\ell = u \wedge \Diamond\varphi) \\ &\Leftrightarrow \Diamond_l\Diamond\varphi\end{aligned}$$

The other direction is simpler:

$$\Diamond_l\Diamond\varphi \Leftrightarrow \Diamond_l(tt\,\mathcal{U}\,\varphi) \Rightarrow (\Diamond_l tt)\,\mathcal{U}\,(\Diamond_l\varphi) \Leftrightarrow tt\,\mathcal{U}\,(\Diamond_l\varphi) \Leftrightarrow \Diamond\Diamond_l\varphi$$

- Commutativity of *always* modality $\Box$ with $\Box_l$ and $\Box_r$:

$$\Box\Box_l\varphi \Leftrightarrow \Box_l\Box\varphi \qquad \Box\Box_r\varphi \Leftrightarrow \Box_r\Box\varphi$$

The proofs are straightforward. For example:

$$\begin{aligned}\Box\Box_l\varphi &\Leftrightarrow \neg\Diamond\neg\neg\Diamond_l\neg\varphi \Leftrightarrow \neg\Diamond\Diamond_l\neg\varphi\\ &\Leftrightarrow \neg\Diamond_l\Diamond\neg\varphi \Leftrightarrow \neg\Diamond_l\neg\neg\Diamond\neg\varphi \Leftrightarrow \Box_l\Box\varphi\end{aligned}$$

- Commutativity among $\Diamond$, $\Box_l$, $\Box_r$, and their dualities:

$$\Diamond\Box_l\varphi \Rightarrow \Box_l\Diamond\varphi \qquad \Diamond\Box_r\varphi \Rightarrow \Box_r\Diamond\varphi$$
$$\Diamond_l\Box\varphi \Rightarrow \Box\Diamond_l\varphi \qquad \Diamond_r\Box\varphi \Rightarrow \Box\Diamond_r\varphi$$

We give proofs of two of them. The first:

$$\Diamond\Box_l\varphi \Leftrightarrow tt\,\mathcal{U}\,(\Box_l\varphi) \Leftrightarrow (\Box_l tt)\,\mathcal{U}\,(\Box_l\varphi) \Rightarrow \Box_l(tt\,\mathcal{U}\,\varphi) \Leftrightarrow \Box_l\Diamond\varphi$$

Because $\Diamond\Box_l\neg\varphi \Rightarrow \Box_l\Diamond\neg\varphi$, then $\neg\Box_l\Diamond\neg\varphi \Rightarrow \neg\Diamond\Box_l\neg\varphi$. The third:

$$\Diamond_l\Box\varphi \Leftrightarrow \Diamond_l\neg\Diamond\neg\varphi \Leftrightarrow \neg\Box_l\Diamond\neg\varphi \Rightarrow \neg\Diamond\Box_l\neg\varphi \Leftrightarrow \Box\Diamond_l\varphi$$

3 REAL-TIME LANGUAGE

In this section, we use the logic to give semantics of a real-time programming language, thus illustrating the power of the logic and a style of real-time semantics. The chosen language is an OCCAM-like language with variables, assignments, the usual control structures, message-passing synchronous communication, choice, etc. No sharing of variables is present in the language. Similar studies can be found in, for example [3, 10, 16].

3.1 Syntax

The language uses the following symbols:

- Time: $t \in (0,\infty) = \mathbb{R}^+$. $\mathbb{R}^+$ is the set of positive real numbers.
- Variables: A finite set of *program variables*, $x, y, x_1, x_2, \ldots$.
- Communication channels: A finite set of *channels*, $c, d, c_1, \ldots$.
- Arithmetic expressions: $e, e_1, e_2, \ldots$.

- Boolean expressions: $B, B_1, B_2, \ldots$.
- Sequential processes: $S, S_1, S_2, \ldots$.
- Parallel processes: $P, P_1, P_2, \ldots$.

The syntax of the language is:

$$
\begin{aligned}
S &::= (x := e) \mid S_1; S_2 \mid \textbf{if } B \textbf{ then } S \mid \textbf{while } B \textbf{ do } S \mid \textbf{wait } t \mid \\
&\quad c?x \mid c!e \mid (c?x \rightarrow S_1 [\!] d?y \rightarrow S_2) \mid (c?x \rightarrow S_1 [\!] \textbf{wait } t \rightarrow S_2) \\
P &::= S \mid (P_1 || P_2)
\end{aligned}
$$

where:

$x := e$ is an assignment.
$S_1; S_2$ is the sequential composition of processes S_1 and S_2.
if B **then** S behaves like S if B evaluates to *true*; otherwise, it does nothing.
while B **do** S is the ordinary iteration.
wait t delays a process for t time units.
$c?x$ stands for receiving a message from channel c. If the communication is confirmed (i.e. it is synchronized with a send request on c), the message passed via c is assigned to variable x; otherwise, it waits forever. We adopt the *prompt* model of communication: a communication takes place as soon as the two partners are ready.
$c!e$ stands for sending a message through channel c. The process waits (possibly forever) for confirmation and then sends the value of e.
$(c?x \rightarrow S_1 \,[\!]\, d?y \rightarrow S_2)$ is a choice. Only the communication that is confirmed first (via channel c or channel d) takes place. When two communications are confirmed at the same time, the choice is arbitrary.
$(c?x \rightarrow S_1 \,[\!]\, \textbf{wait } t \rightarrow S_2)$ is the timeout structure. If communication via channel c is confirmed in less than t time units, the first branch is chosen. Otherwise, the second branch is chosen.
$(P_1 || P_2)$ is the parallel composition of sequential processes P_1 and P_2. The two processes have no shared variables, so the only means of interaction is communication via channels. Communication channels are directed. A channel connects two processes, one at each end.

3.2 Semantic Model

(a) Variables and Channels

The semantics of the language defines *denotations* for sequential and parallel processes by formulas of the logic over specific temporal variables. First, we determine the denotations for program variables and channels.

program variable: We assume real numbers as the only data type for variables, but the value of a variable may change over time. Temporal variables

of the logic are used to denote variables. For each variable x, a temporal variable x is introduced in the semantic definition, where the same name is used for simplicity. Given an interpretation $\mathcal{I}$ and a point interval $[b, b]$, $\mathcal{I}(i)(x)([b, b]) = r$ means that the value of x at time b on axis i is r.

communication channel: For each channel c, three temporal variables c, $c?$ and $c!$ are introduced, where the name c is also overloaded. All communication values are of type $\mathbb{R}$. Variable c in a point interval records the value passed over channel c at that communication time. For the 0/1 valued temporal variable $c?$, $\mathcal{I}(i)(c?)[b, b] = 1$ indicates a request at time b on axis i to receive a message from channel c, while value 0 means no request is present. Similarly, $c!$ has value 1 to denote a sending request via c and 0 otherwise.

Temporal variables $c?$ and $c!$ are called *communication ports*. The set of all ports of process P is denoted by $ports(P)$. We say that a port is *enabled* at a time when it has value 1 at that time and *disabled* otherwise. We use $vars(P)$ to designate the set of all variables of P. In a parallel process $(P_1 \parallel P_2)$, P_1 and P_2 have no common ports and no shared variables. Thus, $port(P_1) \cap port(P_2) = \emptyset$ and $vars(P_1) \cap vars(P_2) = \emptyset$.

(b) Abbreviations

In order to simplify the semantic definitions, we introduce a set of abbreviations. Operator $\doteq$ denotes equality over point intervals and $=$ denotes equality on all points of an interval.

$$x \doteq \theta \;\widehat{=}\; (x = \theta) \wedge (\ell = 0)$$
$$x = \theta \;\widehat{=}\; \neg(\mathrm{tt}^\frown \neg(x \doteq \theta)^\frown \mathrm{tt})$$

The following abbreviations are used to describe value-preservation of variables and ports.

$$\overset{\bullet}{kp}_P\ (X) \;\widehat{=}\; \bigwedge_{q \in ((vars(P) \cup ports(P)) - X)} q \doteq \bigcirc q$$
$$\bar{kp}_P \;\widehat{=}\; \bigwedge_{q \in (vars(P) \cup ports(P))} \exists u.(q = u)$$

$\overset{\bullet}{kp}_P\ (X)$ carries values of variables and ports of process P that are not in set X from a time point of the current axis to the same point of the next axis. When P is clear from the context, we omit the subscript P and simply use $\overset{\bullet}{kp}\ (X)$. Furthermore, $\overset{\bullet}{kp}\ (\{x_1, c_2\})$ is abbreviated as $\overset{\bullet}{kp}\ (x_1, c_2)$. The special case $\overset{\bullet}{kp}\ ()$ is written as $\overset{\bullet}{kp}$, which carries values of all variables and ports of a process into the next axis. Secondly, $\bar{kp}_P$ is used to pass values of all variables

and ports at the beginning of an interval to every point of the interval. Similar abbreviations, such as $\bar{kp}$, are also used.

Abbreviation $\bar{np}\,(X)$ (shown below) indicates that no port in X is enabled in a left-closed, right-open interval, while $\bar{np}\,(c?, d?)$ means that ports $c?$ and $d?$ are disabled at all points of a half closed interval. Note that $\bar{np}$ is true for all point intervals. The abbreviation $\dot{dis}$ says that all ports in X are disabled at a point.

$$\bar{np}\,(X) \;\widehat{=}\; (\ell = 0) \vee \neg(\mathtt{tt}^\frown(\bigvee_{p \in X}(p \doteq 1))^\frown(\ell > 0))$$

$$\dot{dis}\,(X) \;\widehat{=}\; \bigwedge_{p \in X}(p \doteq 0)$$

(c) Two Non-logical Axioms

The semantic model of a process will be defined as the trajectory of the process. On the trajectory, the values of all relevant temporal variables are determined. In order to propagate the values on the trajectory to other areas of the two-dimensional space, we stipulate two non-logical axioms:

$$(\ell > 0) \Rightarrow ((x \doteq u) \Rightarrow \bigcirc((x \doteq u)^\frown(\ell > 0)))$$
$$(\ell > 0) \Rightarrow (\bigcirc(x \doteq u) \Rightarrow ((\ell > 0)^\frown(x \doteq u)))$$

The first axiom says that, for any closed non-point interval, the values will be carried upward to all corresponding left-closed, right-open intervals. The second axiom carries value downward to left-open, right-closed intervals below. Note that values at isolated points are not propagated by these axioms.

(d) Semantic Definitions

We use a *continuation* approach to describe the semantics of the language. Let S be a sequential process. We will define:

$$[\![S]\!] \in \mathcal{D} \to \mathcal{D}$$

where $\mathcal{D}$ is the set of logical formulas. Let $C \in \mathcal{D}$ be a formula defining the semantic model of a continuation of S. $[\![S]\!]C$ defines the meaning of S with continuation C. It is assumed that a program begins its execution at an arbitrary time point on time axis 0. All sequential processes start simultaneously. Semantic definitions of various processes are listed below, where the abbreviations of $\dot{kp}$ and $\bar{kp}$ are widely used in the contexts.

• $x := e$ The statement, a timeless discrete action, assigns the value of e to x on the next axis, and leaves all other variables and ports unchanged.

$$[\![x := e]\!]C \;\widehat{=}\; \dot{kp}\,(x) \wedge \bigcirc x \doteq e \wedge \bigcirc C$$

• $S_1; S_2$ With continuation, sequential composition is defined by:

$$[\![S_1; S_2]\!]C \mathrel{\hat{=}} [\![S_1]\!]([\![S_2]\!]C)$$

• **if** B **then** S In the semantics of this process, $[\![\dot{B}]\!]$ denotes the value of B at the current time on the current axis:

$$[\![\textbf{if } B \textbf{ then } S]\!]C \mathrel{\hat{=}} ([\![\dot{B}]\!] \wedge [\![S]\!]C) \vee (\neg\, [\![\dot{B}]\!] \wedge C)$$

• **wait** t This statement delays its continuation C for t time units, keeping all relevant variables and ports unchanged within this period.

$$[\![\textbf{wait } t]\!]C \mathrel{\hat{=}} \Diamond_r((\bar{kp} \wedge \ell = t)^\frown(\ell = 0 \wedge C))$$

• $c?x$ Execution moves to next axis, where it retains the values of all variables and ports but enables $c?$. Thereafter, it looks up and rightward for $c! \doteq 1$ to reach a synchronization. When a synchronization is reached, value passing is triggered: x gets a new value from c and $c?, c!$ are disabled. Here, we assume that waiting for synchronization may take time, but value passing is a timeless action. When synchronization fails, the process goes into an infinite wait. These two cases are described by a disjunction.

$$\begin{aligned}[\![c?x]\!]C \mathrel{\hat{=}}\ & \dot{kp}\,(c?) \wedge \bigcirc(c? \doteq 1 \wedge \\ & (\ \Diamond_r((\dot{kp}^\frown \bar{kp})\,\mathcal{U}\,(\bar{kp} \wedge (\bar{np}\,(c!)^\frown(c! \doteq 1 \wedge \mathit{Com}_1)))) \\ & \ \vee \Box_r(\bar{kp} \wedge c! \doteq 0)\ )\ )\end{aligned}$$

where

$$\mathit{Com}_1 \mathrel{\hat{=}} \dot{kp}\,(c?, x) \wedge \bigcirc(x \doteq c \wedge \dot{dis}\,(c?, c!) \wedge C)$$

Note that when $(c! \doteq 1)$ occurs on a time axis below and/or at a time before $(c? \doteq 1)$, by the symmetric semantic definition for $c!e$, the communication partner will look for $(c? \doteq 1)$ for the synchronization.

• $c!e$ The semantics of $c!e$ is symmetric to that for $c?x$.

$$\begin{aligned}[\![c!e]\!]C \mathrel{\hat{=}}\ & \dot{kp}\,(c!) \wedge \bigcirc(c! \doteq 1 \wedge \\ & (\ \Diamond_r((\dot{kp}^\frown \bar{kp})\,\mathcal{U}\,(\bar{kp} \wedge (\bar{np}\,(c?)^\frown(c? \doteq 1 \wedge \mathit{Com}_2)))) \\ & \ \vee \Box_r(\bar{kp} \wedge c? \doteq 0)\ )\ )\end{aligned}$$

where

$$\mathit{Com}_2 \mathrel{\hat{=}} \dot{kp}\,(c!, c) \wedge \bigcirc(c \doteq e \wedge \dot{dis}\,(c?, c!) \wedge C)$$

• $c?x \to S_1 \parallel d?x \to S_2$ The semantics of this process mimics the one for $c?x$. Initially, both ports $c?$ and $d?$ are enabled. Then the process looks for $(c! \doteq 1) \vee (d! \doteq 1)$. When found, a value is passed via the synchronized channel; when both are synchronized at same time, the choice is arbitrary.

$$[\![c?x \to S_1 \parallel d?x \to S_2]\!]C \mathrel{\widehat{=}} \dot{kp}\,(c?, d?) \wedge \bigcirc(c? \doteq 1 \wedge d? \doteq 1 \wedge \left(\begin{array}{l} \Diamond_r(\,(\dot{kp} \frown \bar{kp})\,\mathcal{U} \\ \qquad (\bar{kp} \wedge (\bar{np}\,(c!, d!) \frown ((c! \doteq 1 \wedge Com_3) \vee (d! \doteq 1 \wedge Com_4))))) \\ \vee\ \Box_r(\bar{kp} \wedge c! \doteq 0 \wedge d! \doteq 0) \end{array}\right))$$

where

$$Com_3 \mathrel{\widehat{=}} \dot{kp}\,(c?, d?, x) \wedge \bigcirc(x \doteq c \wedge \dot{dis}\,(c?, d?, c!, d!) \wedge [\![S_1]\!]C)$$
$$Com_4 \mathrel{\widehat{=}} \dot{kp}\,(c?, d?, x) \wedge \bigcirc(x \doteq d \wedge \dot{dis}\,(c?, d?, c!, d!) \wedge [\![S_2]\!]C)$$

• $c?x \to S_1 \parallel \mathbf{wait}\, t \to S_2$ If a synchronization with $c!$ can be reached within t time units, the first branch is chosen; otherwise, the second one is chosen.

$$[\![c?x \to S_1 \parallel \mathbf{wait}\, t \to S_2]\!]C \mathrel{\widehat{=}} \dot{kp}\,(c?) \wedge \bigcirc(c? \doteq 1 \wedge \left(\begin{array}{l} \Diamond_r(\ell < t \wedge ((\dot{kp} \frown \bar{kp})\,\mathcal{U}\,(\bar{kp} \wedge (\bar{np}\,(c!) \frown (c! \doteq 1 \wedge Com_5))))) \\ \vee\ (\neg\Diamond\Diamond_r(\ell < t \wedge (\mathrm{tt} \frown (c! \doteq 1))) \wedge \\ \qquad \dot{kp}\,(c?) \wedge \bigcirc\Diamond_r(\ell = t \wedge (((c? \doteq 0) \frown \bar{kp}) \frown (\ell = 0 \wedge [\![S_2]\!]C)))) \end{array}\right))$$

where

$$Com_5 \mathrel{\widehat{=}} \dot{kp}\,(c?, x) \wedge \bigcirc(x \doteq c \wedge \dot{dis}\,(c?, c!) \wedge [\![S_1]\!]C)$$

• **while** B **do** S To define the semantics of iteration, we extend the logic with infinite disjunction and conjunction*. Let A_i $(i \in \mathbb{N})$ be formulas. $\bigvee_{i\in\mathbb{N}} A_i$ is an infinite disjunction of A_i, which defines models satisfying at least one A_i. Similarly, $\bigwedge_{i\in\mathbb{N}} A_i$ defines models that satisfy every A_i.

For **while** B **do** S we introduce the following formulas

$$W^0_{B,S}(C) \mathrel{\widehat{=}} \neg\,[\![\dot{B}]\!] \wedge C$$
$$W^{n+1}_{B,S}(C) \mathrel{\widehat{=}} [\![\dot{B}]\!] \wedge [\![S]\!] W^n_{B,S}(C)$$
$$W^{\omega}_{B,S}(C) \mathrel{\widehat{=}} \bigvee_{n\in\mathbb{N}} W^n_{B,S}(C)$$

and

$$F^0_{B,S} \mathrel{\widehat{=}} \mathrm{tt}$$
$$F^{n+1}_{B,S} \mathrel{\widehat{=}} [\![\dot{B}]\!] \wedge [\![S]\!] F^n_{B,S}$$
$$F^{\omega}_{B,S} \mathrel{\widehat{=}} \bigwedge_{n\in\mathbb{N}} F^n_{B,S}$$

*We don't discuss the fixed point operator of the logic, cf. [13].

where $W^{\omega}_{B,S}(C)$ defines the finite iteration of the process and $F^{\omega}_{B,S}$ defines the infinite iterations. Then, the semantics of the **while** statement is defined as:

$$[\![\textbf{while } B \textbf{ do } S]\!]C \mathrel{\widehat{=}} W^{\omega}_{B,S}(C) \vee F^{\omega}_{B,S}$$

We can now define the semantics of parallel processes. Suppose P is a parallel composition of n sequential processes, $P = (S_1 \,||\, \ldots \,||\, S_n)$, $(n \geq 1)$. The semantics of P is defined as:

$$[\![P]\!] \mathrel{\widehat{=}} \bigwedge_{i \in 1..n} [\![S_i]\!](\Box_r \, \bar{kp}_{S_i})$$

We use $\Box_r \, \bar{kp}_{S_i}$ as the continuation of process S_i. This means that, once S_i terminates, the values of all its variables and ports will stay the same forever.

3.3 Examples

We give two simple examples to illustrate the semantics defined above.

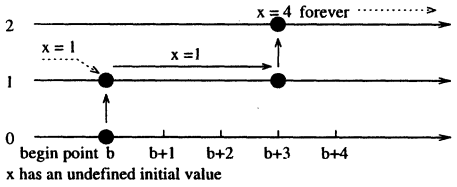
Example 1

$x := 1$; **wait** 3; $x := x + 3$

We assume that x is the only program variable and that there are no communication ports. The semantics of this process can be derived as:

$$\begin{aligned}
&[\![x := 1;\ \textbf{wait } 3;\ x := x + 3]\!](\Box_r \, \bar{kp}) \\
&= \bigcirc x \doteq 1 \wedge \bigcirc([\![\textbf{wait } 3;\ x := x + 3]\!](\Box_r \, \bar{kp})\,) \\
&= \bigcirc(x \doteq 1 \wedge \Diamond_r((x \doteq 1 \wedge \ell = 3)^\frown(\ell = 0 \wedge [\![x := x + 3]\!](\Box_r \, \bar{kp})))) \\
&= \bigcirc(x \doteq 1 \wedge \Diamond_r((x \doteq 1 \wedge \ell = 3)^\frown(\ell = 0 \wedge (\bigcirc x \doteq 4 \wedge \bigcirc(\Box_r \, \bar{kp}))))) \\
&= \bigcirc(x \doteq 1 \wedge \Diamond_r((x \doteq 1 \wedge \ell = 3)^\frown(\ell = 0 \wedge \bigcirc(x \doteq 4 \wedge \Box_r(x \doteq 4)))))
\end{aligned}$$

The picture below shows the execution trajectory of this process, where it is supposed that the process starts at time point b on axis 0.

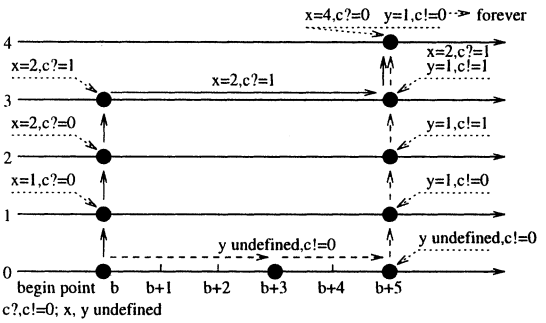


By the two non-logical axioms, the value of x is determined from time b onward on all axes. In interval $(b, b+3)$, x has value 1 on all axes. In interval $(b+3, \infty)$ of any axis, x has value 4. At points b and $b+3$, the situation is different. For example, at time b of axis 0, x has an undefined initial value, but it has value 1 on all axes from axis 1 upward.

Example 2

$x := 1;\ x := x + 1;\ c?x \parallel \textbf{wait}\ 3; \textbf{wait}\ 2;\ y := 1;\ c!(y + 3)$

Here x and $c?$ are the only program variable and communication port of the left sequential process, while y and $c!$ the only ones for the right. We give here only a picture instead of a semantic formula to explain the defined semantics of the parallel process. We assume that all ports are initially disabled.



The execution trajectories of the two sequential processes are presented by solid arrows and dashed arrows respectively. We can see that the right sequential process has one stuttering step to reach synchronization with its partner.

3.4 Process Properties

We now demonstrate how to formulate various program properties in the logic.

Precondition: A precondition gives initial values to all variables and disables all ports. When $\{x_i\}$ and $\{p_j\}$ are the sets of variables and ports of the process, and $\{r_i\}$ a set of real numbers, a precondition is in the form:

$$pre \mathrel{\widehat{=}} \bigwedge_{\text{all variables } x_i} (x_i \doteq r_i) \wedge \bigwedge_{\text{all ports } p_j} (p_j \doteq 0)$$

Convergence: A process is *divergent* if it performs infinite timeless discrete actions in a finite time period. For example, if x is greater than 0 initially,

while $x > 0$ **do** $x := x + 1$ will perform infinite discrete action $x := x + 1$ at a time point. Hence, a divergent process has a vertical boundary. A process is *convergent* if it is not divergent. Therefore, a convergent process can progress horizontally forever and can determine a value of any of its variables in the right half of the plane. Let P be a process and x be a variable of P. P is convergent under pre-condition *pre* if

$$pre \wedge [\![P]\!] \Rightarrow \Box_r \Diamond_r \Diamond_r \exists u.(x \doteq u \wedge \ell > 0)$$

Subsequently, we are interested only in properties of convergent processes.

Termination: Let $P = (S_1 \,||\, \ldots \,||\, S_n)$, $(n \geq 1)$. We introduce an additional boolean valued temporal variable tm_i into the continuation of each S_i.

$$cnt_i \mathrel{\hat{=}} \bar{kp}_{S_i} \wedge (tm_i \doteq 1)$$

Then the semantics of P is changed to

$$[\![P]\!] \mathrel{\hat{=}} \bigwedge_{i \in 1..n} [\![S_i]\!](\Box_r cnt_i)$$

Termination of P can therefore be formulated as

$$tm(P) \mathrel{\hat{=}} \Diamond_r \Box_r \bigwedge_i (tm_i \doteq 1)$$

A postcondition of P can be specified as a relation among variables of P that eventually becomes true and stays true forever. Using $\overline{post}$ for "*post* is true everywhere in an interval where the beginning point is not taken into account", the proof obligation for total correctness of P with respect to precondition *pre* and postcondition *post* is:

$$pre \wedge [\![P]\!] \Rightarrow (tm(P) \wedge \Diamond_r \Box_r \overline{post})$$

Deadlock-Freedom: A deadlock is an endless waiting for a synchronization between, say, $c?$ and $c!$, where one of $c?$ and $c!$ always has value 1 and the other 0. Therefore, the property of deadlock-freedom of process P with private channels $c_1, c_2, \ldots, c_n$ can be formulated as:

$$\bigwedge_{j \in 1..n} \Box_r \Diamond_r \Diamond_r ((\ell > 0) \wedge (c_j? \doteq c_j!))$$

This formula specifies that, for every channel c_j, after any time interval, a

non-point interval can be found where $c_j?$ and $c_j!$ have the same value. In fact, they are both disabled in that interval.

Real-Time Properties: We give two formulations as examples.

(1) Relation φ is never continuously true for longer than t time units:

$$\Box_r \Box_r (\overline{\varphi} \Rightarrow (\ell < t))$$

(2) Any two appearances of φ have a separation of at least t time units:

$$\Box_r \Box_r ((\overline{\varphi}^\frown \overline{\neg\varphi}^\frown \overline{\varphi}) \Rightarrow (\ell \geq t))$$

4 DISCUSSION

Our work aims to establish a two-dimensional modal logic that can be used as a logical foundation for specifying and reasoning about real-time systems, where super-dense computation is assumed. In this paper, we investigate a combination of linear temporal logic and interval logic. Combinations of other logics may also be interesting (see e.g. [9]).

Based on this combination, one can develop logics for system trajectories. The one shown in the paper is the execution trajectory of real-time processes. The mathematical models suggested in [5, 8, 11] provide abstractions of system trajectories, and, we believe, can be formalized as an extension of this combined logic. A general trajectory logic can be used as a logical framework for various models of real-time systems and can become an interesting research topic. How to develop a complete calculus for the combined logic is another research topic. It would be interesting to define the semantics of a synchronous language, such as Esterel, in terms of this logic.

ACKNOWLEDGEMENT

We gratefully acknowledge the instructive comments from the five referees, the carefully reading and correction of the paper by David Gries, and helpful suggestions from our colleagues in UNU/IIST.

REFERENCES

[1] G. Berry and G. Gonthier. The Esterel synchronous programming language: design, semantics, implementation. In *Science of Computer Programming 19* (2), 87–152, Elsevier, 1992.

[2] T. Bolognesi and F. Lucidi. LOTOS-like process algebras with urgent or timed interactions. In *Formal Description Techniques, IV*, 249-264, North-Holland, 1992.

[3] R. Koymans. *Specifying message passing and time-critical systems with temporal logic.* LNCS 651, Springer-Verlag, 1992.

[4] Leslie Lamport. The temporal logic of actions. Technical Report 79, DEC System Research Centre, 1991.

[5] Z. Liu, A. P. Ravn and X. Li. Duration properties of timed transition systems. Technical Report 1997/12, Leicester University, 1997.

[6] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: specification.* Springer-Verlag, 1991.

[7] Z. Manna and A. Pnueli. Models of reactivity. *Acta Informatica 30* (7), 609–678, Springer-Verlag, 1993.

[8] Paritosh K. Pandya and Dang Van Hung. Duration calculus of weakly monotonic time. Technical Report No.122, UNU/IIST, 1997.

[9] Rana Barua and Zhou Chaochen. Neighbourhood logic: NL and NL^2. Technical Report No.120, UNU/IIST, 1997.

[10] G. M. Reed and A timed model for communicating sequential processes. In *ICALP 86: Automata, Language, and Programming*, LNCS 226, 314-323, Springer-Verlag, 1986.

[11] Mahalik Swarup and Xu Qiwen. Compositional verification in duration calculus. Technical Notes, UNU/IIST, 1997.

[12] Morten U. Sørensen, Odd E. Hansen and Hans H. Løvengreen. Combining temporal specification techniques. In Dov M. Gabbay and Hans J. Ohlbach (Eds.) Proceedings of *Temporal Logic, First International Conference, ICTL'94*, 1-16, Springer-Verlag, 1994.

[13] Wang Hanpin and Xu Qiwen. Infinite duration calculus with fix-point operators. Technical Notes, UNU/IIST, 1997.

[14] Zhou Chaochen, C.A.R Hoare and A.P.Ravn. A calculus of durations. In *Information Processing Letters 40* (5), 269-276, Elsevier, 1991.

[15] Zhou Chaochen and Michael R. Hansen. An adequate first order interval logic. Technical Report No.91, UNU/IIST, 1996.

[16] Zhou Chaochen and Michael R. Hansen. Chopping a point. In J. F. He *et al* (Eds.), *BCS-FACS 7th Refinement Workshop*, Electronic Workshops in Computing, Springer-Verlag, 1996.

BIOGRAPHIES

Zongyan Qiu is an associate professor of Computer Science in the Department of Informatics in Peking University. His main interests are Programming Languages and Formal Methods.

Chaochen Zhou is the director of UNU/IIST. He is on leave of absence from the Software Institute, the Chinese Academy of Science, where he is a professor. His main interest is Formal Technique of Programming, including Formal Semantics, Specification, Verification and Design Calculi for Real-time Systems.