



The United Nations
University

UNU/IIST

International Institute for
Software Technology

Hierarchical Design of a Chemical Concentration Control System

Xu Qiwen and He Weidong

October 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

Hierarchical Design of a Chemical Concentration Control System

Xu Qiwen and He Weidong

Abstract

We use a variant of the Duration Calculus to design a chemical concentration control system which has both nontrivial dynamics and control programs. The system is developed by refinement along the lines proposed in formal methods of software construction. Refinement rules for durational specification formulae are investigated for this purpose. The overall specification is refined into several lower level specifications. One such lower level specification is implemented by software and a control program is proposed and shown to be correct.

Xu Qiwen is a Research Fellow of UNU/IIST. His research interest is in Formal Techniques of Programming, including concurrency, verification, and design calculi. E-mail: qxu@iist.unu.edu

He Weidong is a Fellow of UNU/IIST, on leave from the Department of Control Engineering, Beijing University of Aeronautics and Astronautics, where he is a PhD student. His research interests include Intelligent Control, Modern Flight Control, Qualitative Reasoning and Simulation. At UNU/IIST he studies Hybrid Control Systems. E-mail: hwd@iist.unu.edu

Contents

1	Introduction	1
2	A variant of the Duration Calculus	3
2.1	Basics	3
2.2	Refinement rules	4
3	Specification	6
4	Design of $\llbracket \mathcal{H} \rrbracket$	9
5	Refinement of $\llbracket \mathcal{A} \rrbracket$	14
6	Development of $\llbracket \mathcal{S} \rrbracket; \llbracket \mathcal{C} \rrbracket$	14
7	Discussion	16

1 Introduction

Hybrid systems are dynamic systems consisting of interacting discrete event components and continuous components. This topic is actively studied in computer science and control engineering, due to the increasing importance of embedded and real-time systems. Typically, in a hybrid system, a computer system is used to supervise the behaviour of the continuous part. The presence of both discrete and continuous components has made the correctness of the system much harder to ensure. The difficulty is proportional to the complexity of discrete as well as continuous components. In this paper, using formal methods we design a chemical concentration control system which has both nontrivial dynamics and control programs. For a complex system like this one, formal methods offer a strong tool for achieving high assurance.

A formal development theory has two parts. First is a formal language for expressing requirements and design. Second is a set of rules which support the design, in particular, rules that facilitate the transformation of design from one level to another. Our formal tool is a variant of the Duration Calculus. Along the lines proposed in formal software construction, the system is developed by refinement.

Suppose that the system we are to construct is P (here P is merely a name without any structural contents), and it is expected to satisfy requirement R under assumption A . In a durational calculus framework, A and R are durational formulae, and the semantics of P is also defined by a durational formula, say $\llbracket P \rrbracket$.

Then the correctness of the design is expressed by

$$A \wedge \llbracket P \rrbracket \Rightarrow R$$

This is called a proof obligation, since it is not a fact but a property that has to be satisfied by the system yet to be designed. The pair $\langle A, R \rangle$ is the specification of P . We say that P satisfies specification $\langle A, R \rangle$ if the above proof obligation is true. In a hierarchical design, the system is developed by refinement. The correctness of the refinement is guaranteed by rules specially designed for this purpose. In this paper, a refinement rule is basically of the form

$$\frac{A_1 \wedge \llbracket P_1 \rrbracket \Rightarrow R_1, \dots, A_n \wedge \llbracket P_n \rrbracket \Rightarrow R_n}{A \wedge (\llbracket P_1 \rrbracket \oplus_1 \dots \oplus_n \llbracket P_n \rrbracket) \Rightarrow R}$$

and it means that if the formulae above the line, namely, $A_i \wedge \llbracket P_i \rrbracket \Rightarrow R_i$ are valid, then the formula under the line, namely, $A \wedge \llbracket P_1 \oplus_1 \dots \oplus_n P_n \rrbracket \Rightarrow R$ is also valid. When $n > 1$, we can apply this rule to verify the decomposition of $\llbracket P \rrbracket$ into

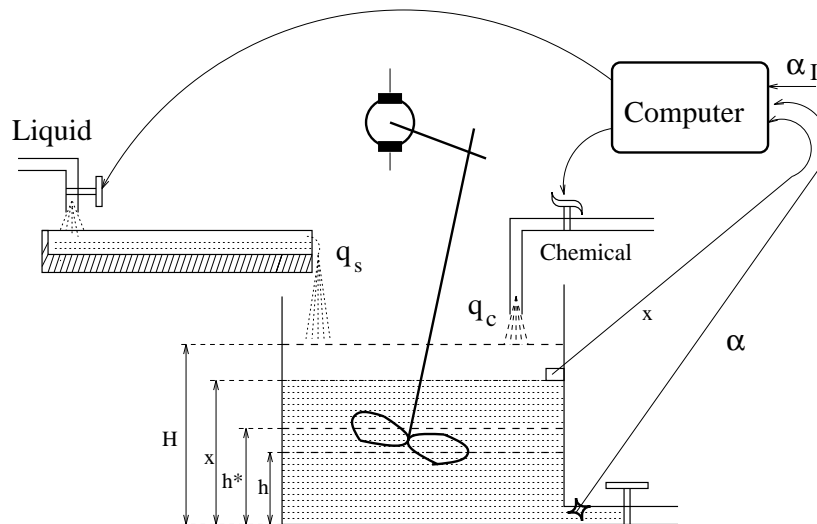
$$\llbracket P_1 \rrbracket \oplus_1 \dots \oplus_n \llbracket P_n \rrbracket$$

and the original proof obligation $A \wedge (\llbracket P_1 \rrbracket \oplus_1 \dots \oplus_n \llbracket P_n \rrbracket) \Rightarrow R$ is discharged into proof obligations about the components $A_i \wedge \llbracket P_i \rrbracket \Rightarrow R_i$. The design process continues with the new formulae $A_i \wedge \llbracket P_i \rrbracket \Rightarrow R_i$. When $n = 1$, the system structure is not changed, and only the specification is mapped into another one (the rule says that any implementation which satisfies the new specification also satisfies the original one). This happens typically when the new specification either gives more information towards implementation or is more suitable for further refinement.

In an ideal top-down design, the system is refined repeatedly until the components can be easily implemented. The advantage of such a constructive approach is that as soon as a design step is carried out one can check its correctness, so that the design and verification proceed at the same time. However, it is not always convenient to follow such a top-down paradigm strictly. For example, when refining P into $P_1 \oplus P_2$. In a strict top-down order, one would define the specifications of P_1 and P_2 before moving on to the design of these components. But sometimes it is difficult to know the exact logical connections of P_1 and P_2 . For example, if P_1 passes a value to P_2 , it is not always clear what property this value should have before the components are further developed. In this case, it may be easier to design P_2 bottom-up. After the construction of P_2 , one extracts a specification of it based on which the specification of P_1 can then be constructed. Therefore, in practice, top-down and bottom-up designs are often used together.

Illustrative Example: Chemical Concentration Control

Our method is illustrated by the design of a chemical concentration control system:



The main components of the system are a tank in which a chemical is dissolved in a liquid and a computer which controls the process. The principal requirement is that the output solution of the tank should be within a limited bound of the requested level and the the solution should be sufficiently uniform.

2 A variant of the Duration Calculus

2.1 Basics

The Duration Calculi is a family of real-time interval logics [8, 9, 10, 11] and have been used in a number of examples of hybrid systems [1, 5]. They are extensions of Moszkowski's Interval Temporal Logic [3]. We shall use a variant of the Duration Calculus as our formal language. In this section, we introduce some basic features of the calculus.

The real-time model of the Duration Calculus consists of a collection of state functions which map from non-negative reals (representing time) to values.

Definition 1 *For an arbitrary state P and an arbitrary closed interval $[c, d]$ ($d \geq c$) the accumulation of P , $\int P$, is defined by the value of integral of P over the interval $[c, d]$:*

$$\int P[c, d] = \int_c^d P(t)dt$$

When P is a function from time to Booleans (represented by $\{0, 1\}$), $\int P$ gives the duration that P holds in the interval.

Durational formulae are functions from the set of all the intervals to Booleans. We say that durational formula B is satisfied over interval $[c, d]$ if

$$B[c, d] = \text{true}$$

Following [11], we provide a way to refer to the state at the beginning and the end of the interval. We adopt the common notation in programming and use an unprimed variable to denote the state at beginning and a primed variable to denote the state at the end of the interval. A state assertion is thus lifted to a durational formula. For example, for a state function x and a constant m

$$((x = m) \wedge (x' = m + 1)[c, d]) = (x(c) = m \wedge x(d) = m + 1)$$

For a state assertion r , let r' be the same assertion with all the state functions primed. By this convention, the above formula can also be written as $(x = m) \wedge (x = m + 1)'$.

As another example,

$$x = x_0 \wedge \int \dot{x} \leq m \Rightarrow x' \leq x_0 + m$$

is a valid formula, because for any interval $[c, d]$,

$$\begin{aligned} & (x = x_0 \wedge \int \dot{x} \leq m \Rightarrow x' \leq x_0 + m) [c, d] \\ & = x(c) = x_0 \wedge \int_c^d \dot{x} \leq m \Rightarrow x(d) \leq x_0 + m \\ & = \text{true} \end{aligned}$$

Definition 2 For a Boolean state P , $[P]$ is the everywhere operator,

$$[P] [c, d] = \forall c \leq t \leq d. P(t)$$

Note that this definition differs from the traditional Duration Calculus where $[P]$ means that state P holds everywhere except possibly at finitely many points.

Definition 3 Let P and Q be two formulae. $P;Q$ is a formula constructed from P and Q by the chop operator $;$. $P;Q$ is satisfied in interval $[c, d]$ iff there exists h ($c \leq h \leq d$) such that P is satisfied in $[c, h]$ and Q is satisfied in $[h, d]$.

Let R be formula. R^* is a formula constructed from R by the star operator. R^* is a formula which is satisfied in interval $[c, d]$, iff there exists a partition h_0, h_1, \dots, h_n , ($c = h_0 \leq h_1 \leq \dots \leq h_{n-1} \leq h_n = d$), such that R is satisfied in all $[h_i, h_{i+1}]$ where $0 \leq i \leq n - 1$.

There are some other operators in the Duration Calculus, but we do not introduce them here, since they are not needed for our case study and can be found in the literature. For the same reason, we do not introduce the full formal proof system of the Duration Calculus. Our design will be primarily supported at a higher level by a number of refinement rules. The soundness of these rules can be proved by the following axioms, where p is a state assertion and A, B are durational formulae.

$$\begin{aligned} (\text{AX1}) \quad & p \wedge (A; B) = (p \wedge A); B \\ (\text{AX2}) \quad & (A \wedge p'); B = A; (p \wedge B) \\ (\text{AX3}) \quad & [p] = \neg((\neg p'); \text{true}) \end{aligned}$$

2.2 Refinement rules

In this section, p, q and r are state assertions, and $[S], [S_1], [S_2]$ and B are durational formulae. The main combinator that our example system uses is the chop operator. The following rule allows us to refine a system into two parts connected by the chop operator.

$$(1) \quad \frac{\begin{array}{l} p \wedge [S_1] \Rightarrow r' \\ r \wedge [S_2] \Rightarrow q' \end{array}}{p \wedge [S_1]; [S_2] \Rightarrow q'}$$

The formulae are of the form $p \wedge \llbracket S \rrbracket \Rightarrow q'$. Its meaning is that for any interval if $\llbracket S \rrbracket$ holds and p holds at the beginning then q holds at the end of the interval. Following the terminology in program logic, we call p and q the pre- and post-conditions respectively. The above rule indicates that to prove that q holds at the end of the interval, it is sufficient to find an assertion r such that it holds after the first phase, and that q holds after the second phase under r as the pre-condition. Notice that when r serves as the post-condition of the first phase, it is primed, and when it serves as the pre-condition of the second phase, it is unprimed. One can see the analogy between this rule and the one for sequential composition in Hoare logic. In a closer setting, a similar rule appeared in Interval Temporal Logic [4].

$$(2) \quad \frac{\begin{array}{l} p \wedge \llbracket S_1 \rrbracket \Rightarrow [q] \wedge r' \\ r \wedge \llbracket S_2 \rrbracket \Rightarrow [q] \end{array}}{p \wedge \llbracket S_1 \rrbracket; \llbracket S_2 \rrbracket \Rightarrow [q]}$$

The second rule is similar in spirit: to prove that $[q]$ holds, we can prove it holds in the first phase under the original pre-condition p and holds in the second phase under the new pre-condition r . By applying this rule repeatedly, we have the following rule about star.

$$(3) \quad \frac{\begin{array}{l} p \Rightarrow r \\ r \wedge \llbracket S \rrbracket \Rightarrow [q] \wedge r' \end{array}}{p \wedge \llbracket S \rrbracket^* \Rightarrow [q]}$$

The following rule gives a method for proving $p \wedge \llbracket S \rrbracket \Rightarrow [q]$. The first premise says that if $p \wedge \llbracket S \rrbracket$ holds over an interval, then B holds over any sub-interval with the same beginning point. The second premise indicates that if $p \wedge B$ holds over an interval, then q holds at the end. Therefore, for any interval $[c, d]$ over which $p \wedge \llbracket S \rrbracket$ is satisfied, it follows that there exists a formula B such that B holds over $[c, t]$ for any $t \leq d$ and consequently q holds at t (end point of $[c, t]$).

$$(4) \quad \frac{\begin{array}{l} p \wedge \llbracket S \rrbracket \Rightarrow \neg((\neg B); \text{true}) \\ p \wedge B \Rightarrow q' \end{array}}{p \wedge \llbracket S \rrbracket \Rightarrow [q]}$$

As a special case, we have the following

$$(5) \quad \frac{\begin{array}{l} p \wedge \llbracket S \rrbracket \Rightarrow [r] \\ p \wedge [r] \Rightarrow q' \end{array}}{p \wedge \llbracket S \rrbracket \Rightarrow [q]}$$

Lastly, we need a rule concerning what are called freeze variables. In contrast to state variables, freeze variables do not occur in implementation but only in specifications. They are widely used

in program logic, and we find them useful in the durational framework too. As a simple example, we want to specify a component S which does not increase the value of x . We can express this as

$$x = x_0 \wedge \llbracket S \rrbracket \Rightarrow \lceil x \leq x_0 \rceil$$

where x_0 is used to ‘freeze’ the initial value of x . From this, we can deduce

$$x \leq 0 \wedge x = x_0 \wedge \llbracket S \rrbracket \Rightarrow \lceil x \leq 0 \rceil$$

Since the conclusion does not involve x_0 , we expect

$$x \leq 0 \wedge \llbracket S \rrbracket \Rightarrow \lceil x \leq 0 \rceil$$

to hold without the pre-condition $x = x_0$. The following rule captures the more general situation

$$(6) \quad \frac{p \wedge X = X_0 \wedge \llbracket S \rrbracket \Rightarrow B}{p \wedge \llbracket S \rrbracket \Rightarrow B}$$

where X is a list of state functions, and X_0 is a list of same number of freeze variables which do not occur freely in B .

Having introduced the refinement rules, we now turn to specifying the chemical concentration control system.

3 Specification

Let the concerned physical state variables be as follows:

- m_c : mass of the chemical in the tank
- m_s : mass of the liquid in the tank
- m_{c1} : mass of dissolved chemical in the tank
- α : concentration level,
- α_* : concentration level if all chemical was dissolved,

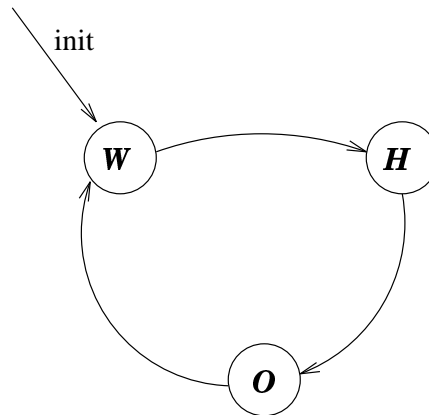
- q_c : rate of supply of the chemical
- q_s : rate of supply of the liquid
- Q : rate of outflow
- x : solution depth

They are all continuous functions from non-negative reals to non-negative reals. The dynamics of the system are described by the following equations:

$$(7) \quad \begin{cases} \alpha = \frac{m_{c1}}{m_s + m_{c1}} \\ \alpha_* = \frac{m_c}{m_s + m_c} \\ \dot{\alpha} = 5(\alpha_* - \alpha) - \frac{\alpha(1-\alpha)}{m_s} q_s \\ \dot{m}_c = q_c - \alpha Q \\ \dot{m}_s = q_s - (1 - \alpha)Q \\ x = k_1(m_c + 3m_s) \end{cases}$$

The third equation characterises the change of α : $5(\alpha_* - \alpha)$ represents the effect from the dissolving process while $\frac{\alpha(1-\alpha)}{m_s} q_s$ represents the effect from adding the liquid. The last equation relates the depth of the solution to the quantities of the chemical and the liquid in the tank.

The execution of the system is divided into three phases. The first one is the waiting phase $[[W]]$. At the end of it the user requests a desired concentration level α_I (α_I is then a discrete variable). Upon receiving such a request, the system enters the handling phase $[[H]]$ in which appropriate amounts of the chemical and the liquid are added to the tank. Afterwards, in phase $[[O]]$, the outlet is opened and some solution is sent to the next workplace. This can be illustrated by the following diagram.



Formally, the system is defined by

$$sys = init \wedge ([\mathcal{W}]; [\mathcal{H}]; [\mathcal{O}])^*$$

where *init* is the initial condition and $[\mathcal{W}]$, $[\mathcal{H}]$ and $[\mathcal{O}]$ are durational formulae characterising the corresponding phases. Phases $[\mathcal{W}]$ and $[\mathcal{O}]$ involve mainly activities of the user. To achieve the control objectives, we can rely on some assumptions. In practice, the requested concentration level is within a limited range. In this case, α_I lies between 60% and 75%. The rate of outflow Q is limited, and in particular, we assume $Q \leq 5(m_c + m_s)$. In every opening phase, the change in the solution depth is limited to between h and $\frac{4}{3}h$. In summary, the assumptions are as follows

$$\begin{aligned} ass = & [60\% \leq \alpha_I \leq 75\% \wedge Q \leq 5(m_c + m_s)] \\ & \wedge ([\mathcal{O}] \Rightarrow [\text{unchanged}(\alpha_I) \wedge q_s = q_c = 0 \wedge h \leq \int \dot{x} \leq \frac{4}{3}h]) \\ & \wedge ([\mathcal{W}] \Rightarrow [Q = q_s = q_c = 0]) \end{aligned}$$

Formula $[\text{unchanged}(\alpha_I)]$ holds if and only if α_I is not changed in the interval. It can be defined precisely, but such details are not important for the purpose of this paper.

We next examine the requirements. The main objective is that the concentration level of the solution flowing out of the tank should be close enough to the requested one. In particular, it is required that the difference between α and α_I should not be more than $0.1\%\alpha_I$. Formally, this is expressed as

$$req_1 \stackrel{\text{def}}{=} Q > 0 \Rightarrow |\alpha - \alpha_I| \leq 0.1\%\alpha_I$$

To ensure that the solution is sufficiently uniform, it is stipulated that $\alpha_* - \alpha$ should not be more than 2%. Let

$$req_2 \stackrel{\text{def}}{=} \alpha_* - \alpha \leq 2\%$$

There are some other relatively easier to satisfy requirements: The chemical can only be put into the tank when $x \geq h^* = \frac{7}{6}h$, and the solution depth x should be restricted to a range between $H = 3h$ and h . Also, obviously, in the handling phase the value of α_I should not be changed.

To summarise, the requirements are listed below

$$\begin{aligned} req = & [Q > 0 \Rightarrow |\alpha - \alpha_I| \leq 0.1\%\alpha_I] & req_1 \\ & \wedge [\alpha_* - \alpha \leq 2\%] & req_2 \\ & \wedge [q_c > 0 \Rightarrow x \geq \frac{7}{6}h] \\ & \wedge [h \leq x \leq H] \\ & \wedge ([\mathcal{H}] \Rightarrow [\text{unchanged}(\alpha_I)]) \end{aligned}$$

Our design task is to refine $\llbracket \mathcal{H} \rrbracket$ into specifications which can be easily implemented and in the meantime ensure that the system satisfies the overall requirements. Correctness of the design amounts to proving

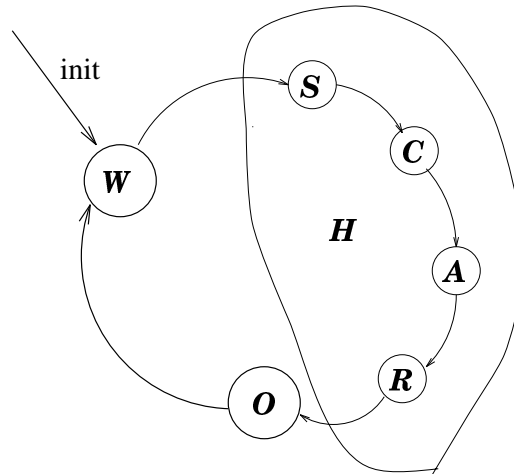
$$Th(DY) \vdash ass \wedge sys \Rightarrow req$$

Here $Th(DY)$ before the \vdash sign indicates the theory about the dynamics of the system can be used in the proof.

To avoid getting into too many details, we concentrate on req_1 and req_2 . In a strictly top-down development, one would derive the specification of $\llbracket \mathcal{W} \rrbracket; \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket$ in the next step. As it turns out, the specification of $\llbracket \mathcal{W} \rrbracket; \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket$ is quite simple and the correctness of the overall system can be easily proved with the help of Rule 3. However at this stage it is difficult to postulate the specification. Therefore, we switch to the bottom-up approach and proceed with the design of $\llbracket \mathcal{H} \rrbracket$ instead.

4 Design of $\llbracket \mathcal{H} \rrbracket$

As a particular design decision, the handling phase is decomposed into four sub-phases: sampling, calculating, adding and resolving.



$$\llbracket \mathcal{H} \rrbracket = \llbracket \mathcal{S} \rrbracket; \llbracket \mathcal{C} \rrbracket; \llbracket \mathcal{A} \rrbracket; \llbracket \mathcal{R} \rrbracket$$

In the calculating phase, the control program computes the appropriate chemical and liquid increments to be added into the tank in the next phase. Clearly, the increments depend on the quantities of the chemical and the liquid in the tank. Therefore, the calculating phase is preceded by a sampling phase in which the related data are collected.

Theorem 1 *At any time point,*

$$\begin{cases} m_s &= \frac{x(1-\alpha_*)}{k_1(3-2\alpha_*)} \\ m_c &= \frac{x\alpha_*}{k_1(3-2\alpha_*)} \end{cases}$$

The depth x can be measured, but α_* cannot. However, α_* and α are reasonably close when *req₂* is satisfied. Therefore, concentration level α is sampled instead and its value is used together with the sampled value of x to calculate the approximated values of m_s and m_c . Let x^0 and α^0 be freeze variables denoting the sampled values. The approximated values are

$$(8) \quad \begin{cases} m_s^{\textcircled{a}} &\stackrel{\text{def}}{=} \frac{x^0(1-\alpha^0)}{k_1(3-2\alpha^0)} \\ m_c^{\textcircled{a}} &\stackrel{\text{def}}{=} \frac{x^0\alpha^0}{k_1(3-2\alpha^0)} \end{cases}$$

Let the chemical increment and the liquid increment be denoted by program variables Δm_c and Δm_s . Recall that α_I denotes the next desired concentration. We shall choose Δm_c and Δm_s that satisfy

$$(9) \quad \alpha_I = \frac{m_c^{\textcircled{a}} + \Delta m_c}{m_c^{\textcircled{a}} + \Delta m_c + m_s^{\textcircled{a}} + \Delta m_s}$$

Obviously, there exist many pairs of such Δm_c and Δm_s . A control program which chooses the exact values of Δm_c and Δm_s will be constructed. But at this level, we do not consider the details of the control program. Instead, now we only postulate the specifications of the sub-phases and prove that if these specifications are satisfied by future implementations then the overall system will satisfy *req₁* and *req₂*.

Recall the requirement that the chemical can only be added when the depth is above $\frac{7}{6}h$. Therefore, if the depth is lower than $\frac{7}{6}h$, only the liquid can be added, and in order for the chemical to be added, the depth after all the liquid is poured in must be at least $\frac{7}{6}h$. Let formula F2 be

$$(10) \quad k_1(m_c + 3(m_s + \Delta m_s)) \geq \frac{7}{6}h$$

To ensure req_2 ,

$$(11) \quad \alpha_* - \alpha \leq 2\%$$

must be satisfied at the beginning. To this end, we postulate the proof obligation for the first two sub-phases

$$\begin{aligned} \text{Proof Obligation 1} \quad & x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \wedge \llbracket \mathcal{S} \rrbracket; \llbracket \mathcal{C} \rrbracket \\ & \Rightarrow (\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F0 \wedge F1)' \wedge F2' \wedge m_c' = m_c^0 \wedge m_s' = m_s^0 \wedge req_2 \wedge [\text{unchanged}(\alpha_I)] \end{aligned}$$

where m_c^0 and m_s^0 are freeze variables. Define formulae F4, F5 and F6 by

$$(12) \quad m_s \leq m_s^{\textcircled{a}} \leq m_{c1} + m_s \wedge m_{c1} \leq m_c^{\textcircled{a}} \leq m_c$$

$$(13) \quad m_c^{\textcircled{a}} + m_s^{\textcircled{a}} \leq m_c + m_s$$

$$(14) \quad \frac{m_c - m_c^{\textcircled{a}}}{m_c} \leq 0.00068$$

$$\begin{aligned} \text{Theorem 2} \quad & x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge \frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246 \\ & \wedge ((\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F0 \wedge F1) \wedge m_c = m_c^0 \wedge m_s = m_s^0)' \\ & \Rightarrow (\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F1 \wedge F4 \wedge F5 \wedge F6)' \end{aligned}$$

In the adding phase, the chemical and the liquid are added into the tank according to the values calculated in the last phase, and in such a way that req_2 is satisfied if $F3$ holds at the beginning. The outlet is closed in this period.

$$\begin{aligned} \llbracket \mathcal{A} \rrbracket & \stackrel{\text{def}}{=} F2 \Rightarrow \\ & \int q_s = \Delta m_s \wedge \int q_c = \Delta m_c \wedge [Q = 0 \wedge \text{unchanged}(\alpha_I) \wedge (q_c > 0 \Rightarrow x \geq \frac{7}{6}h)] \wedge (F3 \Rightarrow req_2) \end{aligned}$$

Condition $F2$ is assumed, for without it $\llbracket \mathcal{A} \rrbracket$ may not be possible to implement. The chemical is then left to dissolve for one time unit or more.

$$\llbracket \mathcal{R} \rrbracket = [Q = q_s = q_c = 0 \wedge \text{unchanged}(\alpha_I)] \wedge l \geq 1$$

The following theorem says that after the chemical is added, the value of α_* is greater than that of α_I .

Theorem 3 $(\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F1 \wedge F4) \wedge \llbracket \mathcal{A} \rrbracket \Rightarrow \alpha'_* \geq \alpha'_I$

It follows from the range of α_I , the value of α_* at the end of the adding phase is no less than 0.6. This fact is then used as the pre-condition of the next phase.

Theorem 4 $F3 \wedge \alpha_* \geq 0.6 \wedge \llbracket \mathcal{R} \rrbracket \Rightarrow (\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246)' \wedge req_2$

Theorem 5 $(\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F1 \wedge F5 \wedge F6) \wedge \llbracket \mathcal{A} \rrbracket \Rightarrow (\frac{\alpha_* - \alpha_I}{\alpha_*} \leq 0.00068)'$

Theorem 6 $\llbracket \mathcal{O} \rrbracket \Rightarrow \int \frac{Q}{m_c + m_s} \leq \frac{4}{3}$

Theorem 7 $\frac{\alpha_* - \alpha_I}{\alpha_*} \leq 0.00068 \wedge \frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246 \wedge \int \frac{Q}{m_c + m_s} \leq \frac{4}{3} \wedge [q_s = q_c = 0 \wedge \text{unchanged}(\alpha_I)]$
 $\Rightarrow (|\alpha - \alpha_I| \leq 0.1\% \alpha_I)'$

Theorem 8 $\frac{\alpha_* - \alpha_I}{\alpha_*} \leq 0.00068 \wedge \frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246 \wedge \llbracket \mathcal{O} \rrbracket \Rightarrow req_1$

Theorem 9 $\alpha_* - \alpha \leq c \wedge [q_s = q_c = 0] \Rightarrow (\alpha_* - \alpha \leq c)'$
 $\frac{\alpha_* - \alpha}{\alpha_*} \leq c \wedge [q_s = q_c = 0] \Rightarrow (\frac{\alpha_* - \alpha}{\alpha_*} \leq c)'$

The above theorems are primarily about each sub-phases. Using the refinement rules, we obtain the following theorem about the composition of the whole handling phase with the opening phase.

Theorem 10 $x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \wedge (\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246) \wedge \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket$
 $\Rightarrow req_1 \wedge req_2 \wedge (\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246)'$

Proof:

$$\begin{aligned}
& x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right) \wedge \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
= & \{ \text{definition of } \llbracket \mathcal{H} \rrbracket \} \\
& x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \wedge \llbracket \mathcal{S} \rrbracket; \llbracket \mathcal{C} \rrbracket; \llbracket \mathcal{A} \rrbracket; \llbracket \mathcal{R} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{Proof Obligation 1} \} \\
& (x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right) \\
& \wedge (\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F0 \wedge F1 \wedge F2)' \wedge m'_c = m_c^0 \wedge m'_s = m_s^0 \wedge req_1 \wedge req_2); \llbracket \mathcal{A} \rrbracket; \llbracket \mathcal{R} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{Theorem 2} \} \\
& (req_1 \wedge req_2 \wedge (\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F1 \wedge F4 \wedge F5 \wedge F6)'); \llbracket \mathcal{A} \rrbracket; \llbracket \mathcal{R} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{Theorem 3, Theorem 5, Rule 1, Rule 2} \} \\
& (req_1 \wedge req_2 \wedge (\alpha_* \geq \alpha_I \wedge \frac{\alpha_* - \alpha_I}{\alpha_*} \leq 0.00068)'); \llbracket \mathcal{R} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{Theorem 4, Rule 1, Rule 2} \} \\
& (req_1 \wedge req_2 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246 \wedge \frac{\alpha_* - \alpha_I}{\alpha_*} \leq 0.00068\right)'); \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{Theorem 8, Theorem 9, Rule 1, Rule 2} \} \\
& req_1 \wedge req_2 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right)'
\end{aligned}$$

Theorem 11 $F3 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right) \wedge \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket \Rightarrow req_1 \wedge req_2 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right)'$

Proof: it follows from Theorem 10 and Rule 6 (the freeze variable rule).

Let $init \stackrel{\text{def}}{=} F3 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right)$.

Theorem 12 $init \wedge \llbracket \mathcal{W} \rrbracket; \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket \Rightarrow req_1 \wedge req_2 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right)'$

Proof:

$$\begin{aligned}
& init \wedge \llbracket \mathcal{W} \rrbracket; \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{assumption about } \llbracket \mathcal{W} \rrbracket \} \\
& req_1 \wedge req_2 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right)'; \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket \\
\Rightarrow & \{ \text{Theorem 11, rule 1, rule 2} \} \\
& req_1 \wedge req_2 \wedge \left(\frac{\alpha_* - \alpha}{\alpha_*} \leq 0.0002246\right)'
\end{aligned}$$

Applying refinement rule 3, we finally conclude that

Theorem 13 $init \wedge (\llbracket \mathcal{W} \rrbracket; \llbracket \mathcal{H} \rrbracket; \llbracket \mathcal{O} \rrbracket)^* \Rightarrow req_1 \wedge req_2$

5 Refinement of $\llbracket \mathcal{A} \rrbracket$

In the specification of $\llbracket \mathcal{A} \rrbracket$, it is stipulated that if $F3$ holds at the beginning then req_2 holds. However, being an appropriate specification statement at a high level, it does not give any information about how to achieve this. Clearly, adding the chemical increases the difference between α_* and α , and therefore the chemical should be added in a sufficiently gentle way. The addition of the liquid has a somewhat more complex impact: it follows from DY that $\alpha_* - \alpha$ increases if $m_c m_{c1} > m_s^2$ and decreases if $m_c m_{c1} < m_s^2$. The following theorem gives a sufficient condition to refine the specification of $\llbracket \mathcal{A} \rrbracket$ taking into account of the combined effect of adding the chemical and the liquid.

Theorem 14
$$F3 \wedge [Q = 0 \wedge \exists k_c, k_s. k_c \geq \frac{m_s}{(m_c + m_s)^2} \wedge k_s \geq (\frac{m_{c1}}{(m_{c1} + m_s)^2} - \frac{m_c}{(m_c + m_s)^2}) \wedge k_c q_c + k_s q_s \leq 0.1] \\ \Rightarrow req_2$$

The specification of $\llbracket \mathcal{A} \rrbracket$ can be refined into

$$F2 \Rightarrow \int q_s = \Delta m_s \wedge \int q_c = \Delta m_c \wedge [Q = 0 \wedge \text{unchanged}(\alpha_I) \wedge (q_c > 0 \Rightarrow x \geq \frac{7}{6}h)] \\ \wedge (F3 \Rightarrow [\exists k_c, k_s. k_c \geq \frac{m_s}{(m_c + m_s)^2} \wedge k_s \geq (\frac{m_{c1}}{(m_{c1} + m_s)^2} - \frac{m_c}{(m_c + m_s)^2}) \wedge k_c q_c + k_s q_s \leq 0.1])$$

As a typical lower level specification, it contains a lot of details. Based on it, it is easy to design a control algorithm which ultimately decides the values of q_c and q_s .

6 Development of $\llbracket \mathcal{S} \rrbracket; \llbracket \mathcal{C} \rrbracket$

Proof Obligation 1 specifies the behaviours of the composition of the sampling and the calculating phases. We now decompose the specification further. Let \bar{x} and $\bar{\alpha}$ be two program variables for storing the sampled values of x and α . At the end of the sampling phase, \bar{x} and $\bar{\alpha}$ are given the initial values of x and α . Hence, let

$$\llbracket \mathcal{S} \rrbracket \stackrel{\text{def}}{=} x = x^0 \wedge \alpha = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \\ \Rightarrow (\bar{x} = x^0 \wedge \bar{\alpha} = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0)' \wedge req_2 \wedge [\text{unchanged}(\alpha_I)]$$

The calculating process then computes the amount of increments based on the sampled values.

$$\llbracket \mathcal{C} \rrbracket \stackrel{\text{def}}{=} \bar{x} = x^0 \wedge \bar{\alpha} = \alpha^0 \wedge m_c = m_c^0 \wedge m_s = m_s^0 \wedge F3 \\ \Rightarrow ((\exists m_s^{\text{@}}, m_c^{\text{@}}. F0 \wedge F1) \wedge F2)' \wedge m_c' = m_c^0 \wedge m_s' = m_s^0 \wedge req_2 \wedge [\text{unchanged}(\alpha_I)]$$

Given the above definitions of $\llbracket \mathcal{S} \rrbracket$ and $\llbracket \mathcal{C} \rrbracket$, it is trivial to show that Proof Obligation 1 indeed holds.

We do not develop the sampling program any further whereas propose an algorithm for the calculating phase. The input-output behaviours of the algorithm will be proved using the traditional program logic and the results are incorporated into the durational framework based on the following rule. Suppose P is a program and $\llbracket P \rrbracket$ is its durational semantics, then

$$(15) \quad \frac{p \{P\} q}{p \wedge \llbracket P \rrbracket \Rightarrow q'}$$

Although the formal verification is centered around req_1 and req_2 in this paper, the algorithm also guarantees that the solution depth will stay between h and H by deciding the increments appropriately. Recall the assumption that in any opening phase the solution depth can only be lowered by $\frac{4}{3}h$, therefore the depth will not fall below h if after each adding phase, the depth is at least $\frac{7}{3}h$. Moreover, to limit the depth below H and leave more room for adjustment in the next round, clearly the new depth should be as low as possible. The algorithm works as follows. First, calculate the amount of the chemical and the liquid if the solution depth is $\frac{7}{3}h$ with concentration level α_I . If the amount of the chemical (or the liquid) is less than the estimated amount of the chemical (or the liquid) already in the tank, then Δm_c (or Δm_s) is taken to be 0. If the amount of the liquid is more than that estimated in the tank, but the difference is not enough to raise the depth over $\frac{7}{6}h$ (recall that the chemical can only be added after that), then Δm_s will be chosen so that after the liquid is added, the depth is $\frac{7}{6}h$. Once one increment is decided, the other increment can be calculated from F1.

The algorithm is simple, but the reason why it works depends on some facts about the dynamics of the system and therefore its verification is not entirely trivial. The algorithm is presented below together with some assertions outlining its verification.

$$\{\bar{x} = x^0 \wedge \bar{\alpha} = \alpha^0 \wedge x^0 = k_1(m_c^0 + 3m_s^0)\}$$

procedure \mathcal{C}

var: $m_s^{\textcircled{a}}, m'_s, m_c^{\textcircled{a}}, m'_c$;

$$m'_s := \frac{7h(1-\alpha_I)}{3k_1(3-2\alpha_I)}; m'_c := \frac{7h\alpha_I}{3k_1(3-2\alpha_I)};$$

$$m_s^{\textcircled{a}} := \frac{\bar{x}(1-\bar{\alpha})}{k_1(3-2\alpha_I)}; m_c^{\textcircled{a}} := \frac{\bar{x}\bar{\alpha}}{k_1(3-2\alpha_I)};$$

$$\Delta m_s := m'_s - m_s^{\textcircled{a}}; \Delta m_c := m'_c - m_c^{\textcircled{a}};$$

$$\{F0 \wedge \bar{x} = k_1(m_c^0 + 3m_s^0)\}$$

if $\Delta m_s < 0 \rightarrow \{x^0 \geq \frac{7}{6}h\} \Delta m_s := 0; \Delta m_c := \frac{\alpha_I(m_s^{\textcircled{a}} + m_c^{\textcircled{a}}) - m_c^{\textcircled{a}}}{1 - \alpha_I}$

$$\parallel \Delta m_c < 0 \rightarrow \{x^0 \geq \frac{7}{6}h\} \Delta m_c := 0; \Delta m_s := \frac{m_c^{\textcircled{a}} - \alpha_I(m_s^{\textcircled{a}} + m_c^{\textcircled{a}})}{\alpha_I}$$

$$\parallel 0 < \Delta m_s < \frac{\frac{7}{6}h - \bar{x}}{3k_1} \rightarrow \Delta m_s := \frac{\frac{7}{6}h - \bar{x}}{3k_1}; \Delta m_c := \frac{\alpha_I(m_s^{\textcircled{a}} + m_c^{\textcircled{a}})}{1 - \alpha_I} - m_c^{\textcircled{a}}$$

$$\parallel \text{else} \rightarrow \text{skip}$$

fi

$$\{(\exists m_s^{\textcircled{a}}, m_c^{\textcircled{a}}. F0 \wedge F1) \wedge k_1(m_c^0 + 3(m_s^0 + \Delta m_s)) \geq \frac{7}{6}h\}$$

We can use $x^0 = k_1(m_c^0 + 3m_s^0)$ as part of the pre-condition, because it is a theorem following from the dynamics of the system. This program only achieves the input-output behaviour of the $\llbracket \mathcal{C} \rrbracket$. Having constructed the control algorithm, the implementation of phase $\llbracket \mathcal{C} \rrbracket$ is easy: just run the program while keeping all the valves closed. Formally, we have the following theorem.

Theorem 15 $\llbracket \text{procedure } \mathcal{C} \rrbracket \wedge [Q = q_s = q_c = 0] \Rightarrow \llbracket \mathcal{C} \rrbracket$

Proof: it follows from the verification of procedure \mathcal{C} and Rule 7.

7 Discussion

A chemical concentration control system is designed in this paper using a variant of the Duration Calculus. The Duration Calculus is a promising tool for real-time and hybrid systems. It is particularly suitable for this case study because some properties are directly associated with intervals and can therefore be naturally expressed.

We find the case study worthwhile. Firstly, the system has non-trivial dynamics and control structures, and consequently the reasoning needed in the design is involved. One of the authors of this paper is from control engineering and the system was first designed without using formal methods. There were serious questions about the correctness of the system. Without a formal notation, the informal arguments were unclear at many places and, without proper structures, many details cannot be covered at reasonable length. The formal design has greatly improved our confidence in the system, and has clarified many points. Secondly, this case study has helped us to identify a number of rules which are useful in the design. We believe these design rules are of general uses and more rules should be studied in order to facilitate the practical applications of the Duration Calculus.

In this case study, we proposed a design and showed that it is correct with respect to the requirements. There are many other interesting properties in control engineering such as stability¹ and robustness which we have not touched. As topics for future study, we shall examine which of these control engineering concepts can be formalised.

Acknowledgements We thank Zhou Chaochen, Chris George and Dines Bjørner for encouragement and helpful comments.

References

- [1] M. Engel, M. Kubica, J. Madey, D.J. Parnas, A.P. Ravn and A.J. van Schouwen: A Formal

¹Stability is not really a difficulty for our system.

- 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.
- [2] 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.
- [3] B. Moszkowski: A Temporal Logic for Multilevel Reasoning about Hardware. *IEEE Computer*, 18,2,pp.10-19,1985.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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. Vytzil (Editors), pp. 738-755, Sept. 1994.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.