Tạp chí Tin học và Điều khiển học, T.20, S.3 (2004), 205–218

DESIGNING A LIFT CONTROL SYSTEM

PHAM TRAN NHU[1], NGUYEN VAN TRUONG[2]

[1]*Institute of Information Technology,*
[2]*Pedagogical University-Thai Nguyen University*

Abstract. In this paper, we present an application of syntactical approach given in a formal design technique for real-time embedded systems. The technique is the model of discretization at the state level and the approximation of continuous state variables by discrete ones. The lift system presented in this paper shall be monitored and controlled by a computing system that shall respect the components, handle the events, and satisfy the usual procedures and invariants. The Duration Calculus with Iteration is used in the paper to specify requirements of the system.

Tóm tắt. Trong bài báo này chúng tôi trình bày một ứng dụng - hệ thống điều khiển thang máy - theo cách tiếp cận hình thức cho các hệ nhúng. Kỹ thuật thiết kế được dùng là mô hình hoá sự rời rạc và xấp xỉ các biến trạng thái liên tục bởi các biến trạng thái rời rạc. Hệ thống thang máy được giám sát và điều khiển thông qua một hệ thống tính toán nhằm quản lý các bộ phận, điều khiển các sự kiện và làm thoả mãn các thủ tục cùng những bất biến đặc trưng cho hệ thống. Tính Toán khoảng lặp được dùng trong bài viết để đặc tả các yêu cầu của hệ thống.

1. INTRODUCTION

The lift control system belongs among real-time control systems. The system consists of some physical plant, in permanent interaction with its environment, for which a suitable controller has to be constructed such that the controlled plant exhibits the desired time dependent behavior. Many authors have proposed approachs for designing the lift control system (e.g. [2, 10]). However, some approach is just a postulate - it has not yet been widely tested, so a failure in the reaction of the plant may appear. The problem is to use suitable technique for specifying and reasoning about the design of the system.

For any real-time systems in general, and for our lift control system in particular, the continuous model (real numbers) is suitable for specifying the continuous behavior of the states of the environment and those of the plant, which can change at any time according to the laws of physics. However, the state of a digital program changes only at discrete time points at ticks of a computer clock, so the discrete model (natural numbers) should be considered for implementation of the system. Therefore, it is appropriate to combine two models into the same formalism such that the design and its correctness can accurately be reasoned about in an uniform manner.

Using formal methods is a key solution for buiding a correct system. In this paper, we apply Duration Calculus with Iteration (DC^*), a logic obtained by extending Duration Calculus (DC) (cf. e.g. [12]) with the iteration operator (*) [1], to model of our lift control system.

This makes for a logical framework that can handle both continuous time and discrete time models for the system.

The design process can be formalised as follows. Firstly, a state variables model of the system should be defined. The state variables model comprises continuous state variables (modeling the behavior of continuous components) and discrete state variables (modeling the behavior of discrete components). Secondly, the requirement of the system is formalized as a DC formula Req over continuous state variables. A design decision must be established in order to the requirement of the system will be met and refined into a detailed design Des over continuous state variables such that $A \vdash Des \Rightarrow Req$, where A stands for some assumptions about the behavior of the environment and the relationship between continuous state variables. Finally, the discretization step follows. We approximate continuous state variables by discrete ones and formalize the relationship between them based on the general behavior of the sensors and actuators. The control requirement is derived from the detailed design and refined into a *simpleDC** formula $Cont$ over discrete state variables such that $A_c \vdash Cont \Rightarrow Des$, where A_c stands for some assumptions about the behavior of the environment and the relationship between continuous state variables, and relationship between discrete state variables and continuous ones. The discrete formula $Cont$ is the formal specification of the controller.

The remaining of the paper is organized as follows. In Section 2 we give a brief summary of DC^*. The discretization technique is peresented in Section 3. Some refinement and verification rules are given in Section 4. The formal design process of the lift control system, the main part of the paper, is contained in Section 5. Section 6 concludes the paper.

2. DURATION CALCULUS WITH ITERATION

In this section we give a brief summary of DC^*. The readers are referred to [1] for more details on the calculus.

A language for DC^* is built starting from the following sets of *symbols*: a set of *constant symbols* $\{a, b, c, \}$, a set of *individual variables* $\{x, y, z, ...\}$, a set of *state variables* $\{P, Q, ...\}$, a set of *temporal variables* $\{u, v, ...\}$, a set of *function symbols* $\{f, g, ...\}$, a set of *relation symbols* $\{R, U, ...\}$, and a set of *temporal propositional letters* $\{A, B, ...\}$.

A DC^* language definition is essentially that of the sets of *state expressions* S, *terms* t and *formulas* φ of the language. These sets can be defined inductively by the following BNFs:

$S \hat{=} \mathbf{0} | P | \neg S | S \vee S$

$t \hat{=} c | x | u | \int S | f(t, ..., t)$

$\varphi \hat{=} A | R(t, ..., t) | \neg\varphi | (\varphi \vee \varphi) | (\varphi^\frown\varphi) | (\varphi^*) | \exists x\varphi$

A state variable P is interpreted as a function $I(P) : IR^+ \rightarrow \{0, 1\}$ (a state). $I(P)(t) = 1$ means that state P is present at time t, and $I(P)(t) = 0$ means that P is not present at time t. We assume that a state has finite variability in any finite time interval. A state expression is interpreted as a function which is defined by the interpretations for the state variables and boolean operators.

For an arbitrary state expression S, its duration is denoted by $\int S$. Given an interpretation I of the state variables and an interval, duration $\int S$ is interpreted as the accumulated length of time within the interval at which S is present. So for any interval $[t, t']$, the interpretation $I(\int S)([t, t'])$ is defined as $\int_t^{t'} I(S)(t)dt$.

A formula φ is satisfied by an interpretation in an interval $[t, t']$ when it evaluates to true for that interpretation over that time interval. This is written as $I, [t, t'] \models \varphi$.

Given an interpretation I, a formula $\varphi^\frown\varphi'$ is true for $[t, t"]$ if there exists a t' such that $t \leq t' \leq t"$ and φ and φ' are true for $[t, t']$ and $[t', t"]$, respectively.

We consider the following abbreviations:

$\ell \triangleq \int \mathbf{1}$, $\lceil S \rceil \triangleq (\int S = \ell) \wedge (\ell > 0)$, $\diamond\varphi \triangleq (true^\frown\varphi^\frown true)$, and $\Box\varphi \triangleq \neg\diamond\neg\varphi$. We assume that boolean connectives bind more tightly than $^\frown$. Besides, we use some other symbols as abbreviations in the usual way.

The proof system for DC^* consists of a complete Hilbert-style proof system for first order logic (cf. e.g. [8]), axioms and rules for interval logic, Duration Calculus axioms and rules and axioms about iteration (cf. e.g. [12]). We only recall here some axioms and rules of the proof system of DC^*.

$(DC1)$ $\int \mathbf{0} = 0$

$(DC2)$ $\int \mathbf{1} = \ell$

$(DC3)$ $\int S \geq 0$

$(DC4)$ $\int S_1 + \int S_2 = \int(S_1 \vee S_2) + \int(S_1 \wedge S_2)$

$(DC5)$ $(\int S = x^\frown \int S = y) \Rightarrow \int S = x + y$

$(DC6)$ $\int S_1 = \int S_2$ if $S_1 \Leftrightarrow S_2$ in propositional calculus.

(IR1)
$$\begin{array}{c}[\ell = 0/A]\varphi \quad \varphi \Rightarrow [A^\frown\lceil S\rceil/A]\varphi \\ \\ \dfrac{\varphi \Rightarrow [A^\frown\lceil\neg S\rceil/A]\varphi}{[true/A]\varphi}\end{array}$$

(IR2)
$$\begin{array}{c}[\ell = 0/A]\varphi \quad \varphi \Rightarrow [\lceil S\rceil^\frown A/A]\varphi \\ \\ \dfrac{\varphi \ \Rightarrow [\lceil\neg S\rceil^\frown A/A]\varphi}{[true/A]\varphi}\end{array}$$

(ω)
$$\frac{\forall k < \omega[(\lceil S\rceil \vee \lceil\neg S\rceil)^k/A]\varphi}{[true/A]\varphi}$$

(DC_1^*) $\ell = 0 \Rightarrow \varphi^*$

(DC_2^*) $(\varphi^\frown\varphi^*) \Rightarrow \varphi^*$

(DC_3^*) $(\varphi^* \wedge \varphi'^\frown true) \Rightarrow (\varphi' \wedge \ell = 0^\frown true) \vee (((\varphi^* \wedge \neg\varphi'^\frown\varphi) \wedge \varphi')^\frown true)$

The proof system of DC^* is complete for sentences where iteration is allowed only for a restricted class of formulas called *simple*.

Definition 1. Simple DC^* formulas are defined inductively by the following BNF

$$\varphi \triangleq \ell = 0 | \lceil S\rceil | a \leq \ell | \ell \leq a | (\varphi \vee \varphi) | (\varphi \wedge \varphi) | (\varphi^\frown\varphi) | \varphi^*$$

Definition 2. Given a simple DC^* formula φ, we define a simple DC^* formula $PREF(\varphi)$ as follows.

1. $PREF(\lceil S \rceil) \triangleq \lceil S \rceil^*$
2. $PREF(a \le \ell) \triangleq \ell \ge 0$
3. $PREF(\ell \le a) \triangleq \ell \le a$
4. $PREF(\varphi \vee \varphi') \triangleq PREF(\varphi) \vee PREF(\varphi')$
5. $PREF(\varphi \wedge \varphi') \triangleq PREF(\varphi) \wedge PREF(\varphi')$
6. $PREF(\varphi^\frown \varphi') \triangleq PREF(\varphi) \vee (\varphi^\frown PREF(\varphi'))$
7. $PREF(\varphi^*) \triangleq \varphi^{*\frown} PREF(\varphi)$

Intuitively, $PREF(D)$ is a simple formula that holds for all prefixes of an interval that validates D. It follows immediately from the definition that

Proposition 1. $\varphi \Rightarrow \neg(\neg PREF(\varphi)^\frown true)$.

The class of simple DC^* formulas plays an important role in our design process presented in section 5. The following section presents the discretization technique.

3. DISCRETE INTERFACE

A model of real-time control systems is depicted in figure 1. The *plant* denotes the continuous componets of the system. The *controller* is a discrete component denoting a control program executed by a computer. The *sensors* sample the states of the plant. The *actuators* receive commands from the controller and control the plant accordingly. The sensors and the actuators constitute the continuous-to-discrete and discrete-to-continuous interfaces respectively.

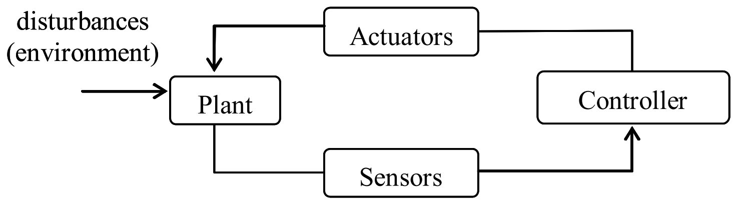

Figure 1. A model of controlled system

In the following part of this section we defined three concepts for formalising the relationship between continuous state variables and discrete ones.

Definition 3. (Stability) Given a state variable s and a positive real number δ, we say s is $\delta - stable$ iff the following formula is satisfied by any interval

$$\delta - stable(s) \triangleq \Box(\lceil \neg s \rceil^\frown \lceil s \rceil^\frown \lceil \neg s \rceil \Rightarrow \lceil \neg s \rceil^\frown (\lceil s \rceil \wedge \ell > \delta)^\frown \lceil \neg S \rceil)$$

The *stability* means that a state should not change quickly in order to be observable at discrete time.

Definition 4. (Control state) Given two state variables r and s, and a non-negative real number δ, we say r $\delta - controls$ s iff the following formula is satisfied by any interval

$$r \rhd_\delta s \overset{\wedge}{=} \Box(\lceil r \rceil \wedge \ell > \delta \Rightarrow (\ell \leq \delta)^\frown \lceil s \rceil)$$

The concept of *control state* is used for formalising the behaviour of actuators. Let r be a state variable modeling a program command, and s a state of the plant. Then the relation $r \rhd_\delta s$ means that whenever the controller issues the command r, the plant gets into state s within at most δ time units. So the maximum response time is δ time units.

Definition 5. (Observation state) Given two state variables r and s, and a non-negative real number δ, we say r $\delta -$ *observes* s iff the following formula is satisfied by any interval.

$$r \overset{\rightarrow}{\approx}_\delta s \overset{\wedge}{=} (s \rhd_\delta r) \wedge (\neg s \rhd_\delta \neg r)$$

The concept of *observation state* can be used for formalizing the behavior of the sensors. Let r be a state variable modeling a discrete program variable, and s a state of the environment. Then the relation $r \overset{\rightarrow}{\approx}_\delta s$ means that any change (stable enough) in s is observed by the controller within δ time units. So the sampling step is δ time units. Note that the definition say nothing about unstable change of s.

We will assume that environment state variables are stable enough to be observable by the controller, otherwise there is no way to observe them in discrete time.

For formalising the discrete interface, for any continuous state variable s, we consider a discrete state variable s_c used by the control program to observe s via the sensors. The relationship between s and its sampling s_c is formalised by $s_c \overset{\rightarrow}{\approx}_\delta s$ for some non-negative real number δ. Similarly, for any state t of the plant we consider a command t_c, a discrete state for requesting (via the actuators) the plant getting into state t. The ralationship between t and t_c is formalised by $t_c \rhd_\tau t$ for some non-negative real number τ.

4. REFINEMENT AND VERIFICATION RULES

Some rules given in this section are useful for both the refinement and the verification. The proofs of some rules and more details are given in [7].

Transitivity rules

$$\text{Rule 1} \quad \frac{(r \rhd_\delta s)(s \rhd_\tau t)}{r \rhd_{(\delta+\tau)} t} \qquad\qquad \text{Rule 2} \quad \frac{(r \overset{\rightarrow}{\approx}_\delta s)(s \overset{\rightarrow}{\approx}_\tau t)}{r \overset{\rightarrow}{\approx}_{(\delta+\tau)} s}$$

These rules say that the accuracy is deteriorated through sequential samplings of a state. They are helpful for the design of distributed systems comprising many sensors, as well as how to use the sensors efficiently.

Observation rules

Rule 3

$$\frac{(r \overset{\rightarrow}{\approx}_\delta s)}{(\lceil s \rceil \wedge \ell > \delta)^\frown(\lceil \neg s \rceil \wedge \ell > \delta) \Rightarrow \ell \leq \delta^\frown \lceil r \rceil^\frown \lceil \neg r \rceil^\frown true}$$

Rule 3 allows to capture the change of state from 0 to 1 or from 1 to 0 by observation.

State Distance

Rule 4a
$$\frac{(r \overrightarrow{\approx}_\delta s)\ \delta - stable(s)}{(\delta+\tau) - stable(r) \Rightarrow \tau - stable(s)}$$

Rule 4b
$$\frac{(r \overrightarrow{\approx}_\delta s)\ \delta - stable(r)}{(\delta+\tau) - stable(s) \Rightarrow \tau - stable(r)}$$

These rules define a necessary condition for the stability of a continuous state, which is the stability of its sampling and vice-versa. It is useful for refinement.

State Occurrence

Rule 5a
$$\frac{(s \rhd_\tau t)}{\Box(\lceil t \rceil \Rightarrow \ell \le \tau) \Rightarrow \Box(\lceil s \rceil \Rightarrow \ell \le \delta + \tau)}$$

Rule 5b
$$\frac{(r \overrightarrow{\approx}_\delta s)}{\Box(\lceil r \rceil \Rightarrow \ell \le \tau) \Rightarrow \Box(\lceil s \rceil \Rightarrow \ell \le \delta + \tau)}$$

These rules are helpful for both refinement and verification. It define how fast the control program should be to satisfy a time constraint about the occurrence of the state.

Duration of state

Rule 6 $PREF(\lceil r \rceil^{*\frown}(\lceil \neg r \rceil^\frown(\lceil r \rceil \land \ell > \delta))^{*\frown}\lceil \neg r \rceil) \Rightarrow \delta - stable(r)$

Rule 7 $PREF(\lceil \neg r \rceil^{*\frown}((\lceil r \rceil \land \ell \le \delta)^\frown\lceil \neg r \rceil)^{*\frown}(\lceil r \rceil \land \ell \le \delta)) \Rightarrow \Box(\lceil r \rceil \Rightarrow \ell \le \delta)$

Invariant Rule for loop

$$\varphi \Rightarrow \neg(true^\frown \neg\alpha)\quad \ell = 0 \Rightarrow \alpha$$

$$\varphi \Rightarrow \neg(\neg\beta^\frown true)\quad \ell = 0 \Rightarrow \beta$$

Rule 8
$$\frac{\alpha^\frown\varphi^{*\frown}\beta \Rightarrow \chi\quad \varphi \Rightarrow \Box\chi}{\varphi^* \Rightarrow \Box\chi}$$

Invariant Rule for sequential concatenation

$$\psi \Rightarrow \Box\chi\quad \varphi \Rightarrow \Box\chi\quad \alpha^\frown\beta \Rightarrow \chi$$

Rule 9
$$\frac{\psi \Rightarrow \neg(\neg\beta^\frown true)\ \varphi \Rightarrow \neg(true^\frown\neg\alpha)}{\varphi^\frown\psi \Rightarrow \Box\chi}$$

Trivial parallel composition

Rule 10
$$\frac{A \Rightarrow \Box\psi\ B \Rightarrow \Box\varphi}{A \land B \Rightarrow \Box(\psi \land \varphi)}$$

Monotonicity

Rule 11a $\dfrac{r \rhd_\tau s\ \tau \le \delta}{r \rhd_\delta s}$ Rule 11b If $r \Rightarrow s$ then $r \rhd_0 s$

Rule 11c $\dfrac{(r \rhd_\delta s)(t \rhd_\delta u)}{(r \land t) \rhd_\delta (s \land u)}$ Rule 11d $\dfrac{(r \overrightarrow{\approx}_\delta s)(t \overrightarrow{\approx}_\delta u)}{(r \land t) \overrightarrow{\approx}_\delta (s \land u)}$

5. DESIGN A SINGLE LIFT CONTROL SYSTEM

5.1 Problem domain description

The logical control of a lift system studied in this paper consists of a simple, single lift system. It allows movement of a single lift cage between a finite number of floors. The starting and stopping of the lift [cage] and the opening and closing of floor doors are made by the pressing of floor call, door close and cage send buttons.

Components: The lift system has the following immediate components: a lift *cage* with send buttons, one for each floor; a *motor*; $N+1$ *floors*, each with a *floor door*, a *call button* and a *close button*; *sensors* and *actuators*; a *controller*.

We identify floors by natural numbers, numbered 0 to N, and assume that the lift can carry any number of clients!

The system state is made up from the above components with their attributes.

Attributes: The system and its components have the following attributes.

+ The fift cage is either stopped at floor j for j lying between 0 and N inclusive, or is moving up (or down) between floors i and $i+1$ (i and $i-1$), for i lying between 0 and $N-1(N$ and $1)$.

+ A floor door is either open or closed.

+ The motor is either running up (or down) or is stopped.

+ The motor, when running, runs at a constant speed-which causes the lift cage to move between immediately neighbouring floors in t_m time units.

Events: We consider only the following events.

+ A send button is pressed for floor k, for $k = 0, ..., N$.

+ A call button on floor k is pressed, for $k = 0, ..., N$.

+ A close button is pressed for the door at floor k, for $k = 0, ..., N$.

+ The opening (and closing) of floor doors.

+ The starting and stopping of the motor-implying the same for the cage.

For the sake of simplicity we do not identify explicitly two journeys of the lift cage: upward one and downward one.

Procedure: A lift journey is procedurally described.

+ *Servicing* a floor k means that a send button is pressed for floor k, or a call button on floor k is pressed, or the lift cage is running upwardly or downwardly (towards floor k).

+ There is a *request* on floor j, means that a call or a send button at floor j is pressed, iff there does not exist any services of floors and the floor door is closed; or a close button at floor j is pressed when the floor door is open. This implies that the lift system services floors succesively. This dogma makes our design simple.

Invariants: The above plus the invariants fully describe expectations.

+ There are at least two floors (a component invariant).

+ The cage has exactly one send button for each floor (a component invariant).

+ Pressing a call button at floor i or pressing a send button for floor i causes the lift to service that floor within t_s time units (a procedural, functional invariant).

+ A floor door may only be open if the lift cage is at that floor (a component safety invariant).
+ The floor door is open for at least t_0 time units and at most $t_{\max}$ time units (a procedural, functional invariant).

The lift system presented in this paper shall be monitored and controlled by a controller that shall respect the components, handle the events, and satisfy the usual procedures and invariants enumerated above.

5.2. Formalizing the requirements of the system

We introduce the following continuous state variables: variable c_i holds if the call button on floor i is pressed, variable s_i holds if the send button for floor i is pressed, variable d_i holds if the door at floor i is open, and variable f_i holds if the lift is at floor i, for i ranges over interval $[0, ..., N]$; variable *motor* hodls if motor is on (and this makes the lift cage move); variable $close_i$ holds if the close button on floor i is pressed (at the time when the door at floor i is open). We do not model lift positions between floors.

The requirements of the system are defined by

$$Req \triangleq \Box(SafetyReq \wedge FunctReq)$$

The safety property for the lift control system is: for every floor, the door must only be opened if the lift is at that floor. This is equivalent to stating that "if the lift is not at floor i, then door i must be closed".

$$SafetyReq \triangleq \lceil d_i \rceil \Rightarrow \lceil f_i \rceil$$

The function requirement is the following conjunction

$$FunctReq \triangleq F_1 \wedge F_2 \wedge F_3$$

Pressing a send button causes the lift to service the corresponding floor within t_s time units.

$$F_1 \triangleq \lceil s_i \rceil ^\frown true \Rightarrow \ell \le t_s \vee (\ell \le t_s ^\frown \lceil d_i \rceil ^\frown true)$$

This requirement states that for every observation interval for which s_i hodls initially, i.e. the send button for the $i'th$ floor is pressed, either the interval is shorter than or equal to t_s or it may be diveded into three subintervals where the first lasts at most t_s, in the second the door at floor i is opened, and a final subinterval which is unconstrained.

A similar condition must hold when pressing a call button: pressing a call button causes the lift to service the corresponding floor within t_s time units.

$$F_2 \triangleq \lceil c_i \rceil ^\frown true \Rightarrow \ell \le t_s \vee (\ell \le t_s ^\frown \lceil d_i \rceil ^\frown true)$$

The system must guarantee that when a floor is serviced, the door is open for at least t_0 time units and at most $t_{\max}$ time units.

$$F_3 \triangleq (\lceil \neg d_i \rceil ^\frown \lceil d_i \rceil ^\frown \lceil \neg d_i \rceil \Rightarrow \ell \ge t_0) \wedge (\lceil d_i \rceil \Rightarrow \ell \le t_{\max})$$

Having defined the requirements, we now present a design decision which implements the requirements.

5.3. Design decision

We define the design decision by the predicate *Des*

$$Des \mathrel{\hat{=}} \Box(D1 \wedge D2 \wedge D3 \wedge D4 \wedge D5 \wedge D6 \wedge D7)$$

The following formula is derived directly from the assumptions of the behaviour of the system as described in section 5.1

$$\begin{aligned}
&D1 \mathrel{\hat{=}} ((\ell = a \wedge \lceil s_i \vee c_i \rceil)^\frown(\ell = b \wedge (\lceil \neg d_i \rceil^\frown \lceil d_i \rceil))\\
&\Rightarrow (\ell = a)^\frown(\ell \leq b \wedge \lceil \neg(s_i \vee c_i \vee s_j \vee c_j)\rceil)^\frown true) \wedge \lceil \neg(close_i \wedge \neg d_i)\rceil\\
&\wedge \lceil \neg(c_i \wedge d_i)\rceil \wedge \lceil \neg(c_i \wedge s_i)\rceil \wedge \lceil \neg(s_i \wedge d_i)\rceil \wedge \lceil \neg(c_i \wedge d_j)\rceil \wedge \lceil \neg(c_i \wedge c_j)\rceil\\
&\wedge \lceil \neg(c_i \wedge s_j)\rceil \wedge \lceil \neg(s_i \wedge s_j)\rceil \wedge \lceil \neg(s_i \wedge d_j)\rceil
\end{aligned}$$

If for every interval for which a send button for floor i is pressed initially, and the lift is at floor j and $j \neq i$, then the interval may be divided into three subintervals where the first lasts at most θ time units, in the second the motor is on, and an unconstrained final subinterval; in the condition of $i = j$, then the door at floor i must be opened within θ time units, where θ stands for a response time of the system. A similar condition must hold when pressing a call button at floor i.

$$\begin{aligned}
D2 &\mathrel{\hat{=}} \lceil (s_i \vee c_i) \wedge f_j \rceil^\frown true \Rightarrow (\ell \leq \theta)^\frown \lceil motor \rceil^\frown true\\
D3 &\mathrel{\hat{=}} \lceil (s_i \vee c_i) \wedge f_i \rceil^\frown true \Rightarrow (\ell \leq \theta)^\frown \lceil d_i \rceil^\frown true
\end{aligned}$$

If a send button for floor i is pressed while lift is at floor j, the lift may reach the destination floor and then the motor is off and the door at the floor is opened within θ time units. A similar condition must hold when pressing a call button at floor i.

$$\begin{aligned}
&D4 \mathrel{\hat{=}} \lceil (s_i \vee c_i) \wedge f_j \rceil^\frown true\\
&\Rightarrow (\ell \leq \theta)^\frown \ell = |i - j| t_m^\frown \lceil (\ell \leq \theta^\frown \lceil d_i \rceil) \wedge \lceil f_i \rceil \rceil^\frown true) \wedge \lceil (s_i \vee c_i) \wedge f_j \rceil^\frown true\\
&\Rightarrow (\ell \leq \theta)^\frown \ell = |i - j| t_m^\frown \lceil (\ell \leq \theta^\frown \lceil \neg motor \rceil) \wedge \lceil f_i \rceil \rceil^\frown true)
\end{aligned}$$

If the close button at floor i is pressed it may make the door at the floor open within θ time units.

$$D5 \mathrel{\hat{=}} \lceil close_i \rceil^\frown true \Rightarrow (\ell \leq \theta)^\frown \lceil \neg d_i \rceil^\frown true$$

Two following formulas will help satisfy the requirement of the maximum and minimum time units for which a door is open.

$$\begin{aligned}
D6 &\mathrel{\hat{=}} \lceil \neg d_i \rceil^\frown \lceil d_i \rceil^\frown true \Rightarrow \lceil \neg d_i \rceil^\frown \ell < t_0^\frown \lceil \neg close_i \rceil^\frown true\\
D7 &\mathrel{\hat{=}} (\lceil d_i \rceil \wedge t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \Rightarrow \lceil d_i \rceil^\frown \lceil \neg d_i \rceil^\frown true
\end{aligned}$$

Inittially the lift is idle at the ground floor with the doors close, motor stops, and no requests for the lift.

$$Init \mathrel{\hat{=}} \lceil \neg motor \wedge f_0 \wedge \neg d_i \wedge \neg s_i \wedge \neg c_i \wedge \neg close_i \rceil^\frown true \vee \lceil\rceil$$

The maximum time it may take to service a floor corresponds to the time it takes to move across $N+1$ floors and the response time it takes to open doors.

$$A1 \triangleq (t_s \geq (n+1)t_m + 2\theta)$$

The following formula is derived directly from the attribute of the motor as described in section 5.1

$$A2 \triangleq \lceil (s_i \vee c_i) \wedge f_j \rceil ^\frown true \Rightarrow \ell \leq \theta ^\frown \ell = |i-j| t_m ^\frown \lceil f_j \rceil ^\frown true$$

We assume that the response time θ is small enough (compare to the speed of the motor) in order to the lift, having reached the destination floor, can be at the floor within at least θ time units while the motor is still running.

$$A3 \triangleq \lceil \neg f_i \rceil ^\frown \lceil f_i \wedge motor \rceil ^\frown true \Rightarrow \lceil \neg f_i \rceil ^\frown \ell \leq \theta \wedge \lceil f_i \rceil ^\frown true$$
$$A4 \triangleq (t_0 + \theta + 1 \leq t_{\max})$$

Let $A \triangleq Init \wedge A1 \wedge A2 \wedge A3 \wedge A4$

The following theorem says that the design Des implies the specification Req, under the assumption A.

Theorem 5.3.1. $A \vdash Des \Rightarrow Req$

Proof. See Appendix.

We will find a discrete specification for the controller as follows.

5.4. Discrete design

For any continuous state variable s, let s_c be the discrete state variable used by the controller to observe s via the sensors. Then the relationships between continuous state variables and discrete state ones are formalised as following formulas: $f_{ic} \overrightarrow{\approx}_\delta f_i, c_{ic} \overrightarrow{\approx}_\delta c_i, d_{ic} \overrightarrow{\approx}_\delta d_i, s_{ic} \overrightarrow{\approx}_\delta s_i$, and $close_{ic} \overrightarrow{\approx}_\delta close_i$, where δ is the sampling step.

Let $Dopen_i, Dclose_i, Mon$, and $Moff$ be discrete state variables, which hold when the controller requests the actuators to open the door at floor i, close the door at floor i, start the motor, and stop the motor, respectively. The relationship between them and the continuous state variables d_i and $motor$ are expressed by $Dopen_i \rhd_\tau d_i$, $Dclose_i \rhd_\tau d_i$, $Mon \rhd_\tau motor$, and $Moff \rhd_\tau \neg motor$, where τ stands for the response time of the plant via the actuators. Besides, we also introduce a symbol ℓ as one described above, but its value can be calculated only by some computer clock.

Let

$$\varphi 1 \triangleq (\lceil (c_{ic} \vee s_{ic}) \wedge f_{jc} \wedge Mon \rceil \wedge \ell = \tau) ^\frown (|i-j| t_m \leq \ell \leq \delta + |i-j| t_m) ^\frown \lceil Moff \rceil$$
$$\varphi 2 \triangleq (\lceil (c_{ic} \vee s_{ic}) \wedge f_{jc} \wedge Mon \rceil \wedge \ell = \tau) ^\frown (|i-j| t_m \leq \ell \leq \delta + |i-j| t_m) ^\frown \lceil Dopen_i \rceil$$
$$\varphi 3 \triangleq (\lceil (c_{ic} \vee s_{ic}) \wedge f_{ic} \wedge Dopen_i \rceil \wedge (\ell = \tau)$$
$$\phi 1 \triangleq (\ell \geq t_0 - \tau) ^\frown \lceil close_{ic} \wedge Dclose_i \rceil$$
$$\phi 2 \triangleq (t_{\max} - \theta - 1 \leq \ell \leq t_{\max} - \theta) ^\frown \lceil Dclose_i \rceil$$

$$\mu 1 \triangleq \neg(\lceil (c_{ic} \vee s_{ic}) \wedge \neg f_{ic} \rceil ^\frown (\ell \leq \theta + |i-j|t_m) ^\frown \lceil c_{ic} \vee s_{ic} \vee c_{jc} \vee s_{jc} \rceil)$$
$$\mu 2 \triangleq \lceil \neg(c_{ic} \wedge d_{jc}) \rceil \wedge \lceil \neg(c_{ic} \wedge c_{jc}) \rceil \wedge \lceil \neg(c_{ic} \wedge s_{jc}) \rceil \wedge \lceil \neg(s_{ic} \wedge s_{jc}) \rceil \wedge$$
$$\lceil \neg(s_{ic} \wedge d_{jc}) \rceil \wedge \lceil \neg(close_{ic} \wedge \neg d_{ic}) \rceil \wedge \lceil \neg(c_{ic} \wedge d_{ic}) \rceil \wedge \lceil \neg(c_{ic} \wedge s_{ic}) \rceil \wedge \lceil \neg(s_{ic} \wedge d_{ic}) \rceil$$
$$\varphi \triangleq \lceil \neg c_{ic} \wedge \neg s_{ic} \rceil^{* \frown} ((\varphi 1 \wedge \varphi 2) \vee \varphi 3 ^\frown \phi 1 \vee \phi 2)^*$$
$$\varphi \triangleq \lceil \neg c_{ic} \wedge \neg s_{ic} \rceil^* \vee \lceil \neg c_{ic} \wedge \neg s_{ic} \rceil ^\frown ((\varphi 1 \wedge \varphi 2) \vee \varphi 3 ^\frown \phi 1 \vee \phi 2)^*$$

A discrete design for the controller is difined as following formula

$$Cont \triangleq (\varphi^{* \frown} \phi \vee PREF(\varphi^*)) \wedge \mu 1 \wedge \mu 2$$

Let

$$A_c \triangleq A \wedge (f_{ic} \overrightarrow{\approx}_\delta f_i) \wedge (c_{ic} \overrightarrow{\approx}_\delta c_i) \wedge (d_{ic} \overrightarrow{\approx}_\delta d_i) \wedge (s_{ic} \overrightarrow{\approx}_\delta s_i) \wedge (close_{ic} \overrightarrow{\approx}_\delta close_i) \wedge$$
$$(Dopen_i \rhd_\tau d_i) \wedge (Dclose_i \rhd_\tau \neg d_i) \wedge (Mon \rhd_\tau motor) \wedge (Moff \rhd_\tau \neg\ motor) \wedge$$
$$(\theta \geq \tau + \delta) \wedge (\lceil Dopen_i \rceil \Rightarrow \ell \leq \theta) \wedge (\lceil Dclose_i \rceil \Rightarrow \ell \leq \theta)$$

The following theorem shows the correctness of the discrete design, under the assumption A_c.

Theorem 5.4.1. $A_c \vdash Cont \Rightarrow Des$

Proof. See Appendix.

6. CONCLUSION

We have designed a lift control system by using the technique of modelling discretization at the state level and the approximating continuous state variables by discrete ones. We consider DC^* as specification language to reason about the design of the system. DC^* has been used successfully in many case studies, see e.g. [1, 5, 7], except for our system described above. Using the technique will make our system design move closer to real world compares to one given in [10], and it is useful for programmers to implement it in some programming language. Besides, it is not difficult to prove the correctness of the design of the system by using the technique.

In general, we just have considered the system with simple requirements and the dogmatic assumption. In our future work, we will use the technique to design more complex and practical systems.

Acknowledgment. The authors would like to thank Mr. Dang Van Hung for his useful references.

REFERENCES

[1] Dang Van Hung, Dimitar P. Guelev, Completeness and Decidability of a Fragment of Duration Calculus with Iteration, *Technical Report 163*, UNU/IIST, P.O.Box 3058, Macau, 1999.

[2] Derek N. Dyck, Peter E. Caines, The Logical Control of an Elevator,*IEEE Transactions on Automatic Control* **40** (1995) 480–486.

[3] Doron A.Peled, *Software Reliability Methods*, Springer-Verlag, USA, 2001.

[4] Edward A. Lee, Embedded Software, *Advances In Computers* **56** (2002) 55–90.

[5] Fancois Siewe, Dang Van Hung, Formal Design Technique For Real-Time Embedded Systems, *Technical Report, Science Conference on the Occasion of 25th foundation year*, Institute of Information Technology, Hanoi, Vietnam, 2001.

[6] Francois Siewe, Dang Van Hung, Deriving Real-Time Programs from Duration Calculus Specifications, *Technical Report 222,* UNU/IIST, P.O. Box 3058, Macau, 2000.

[7] Fanõois Siewe, Dang Van Hung, From Continuous Specification to Discrete Design, *Technical Report 182,* UNU/IIST, P.O. Box 3058, Macau, 2002.

[8] J. Shoenfield, *Mathematical Logic,* Addison-Wesley, Massachusetts, 1967.

[9] Jan Vytopil, *Formal Techniques in Real-Time And Fault-Tolerant Systems,* Kluwer Academic Publishers, USA, 1993.

[10] Kirsten Mark Hansen, Angers Peter Ravn, and Hans Rischel, Designing Verified Real-time Systems, *Technical Report,* Dept. of Computer Science, Technical Uni. of Denmark, EuroMicro '92, Paris, 1992.

[11] Michael Schiebe, *Saskia Pferrer, Real-Time Systems Engineering And Applications,* Kluwer Academic Publishers, USA, 1992.

[12] Michael R. Hansen, Zhou Chaochen, Duration Calculus: Logical Foundations, *Formal Aspects of Computing* **9** (1997) 283–330.

[13] Rajesh Kumar Gupta, *Co-synthesis of Hardware And Software For Digital Embedded Systems*, Kluwer Academic Publishers, USA, 1995.

[14] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, Design of Embedded Systems: Formal Models, Validation, and Synthesis, *Proceedings of the IEEE,* **85** (1997).

Appendix

A Proof of Theorem 5.3.1

We shall prove that $A \vdash Des \Rightarrow Req$.
Observation:

+ If $T \vdash A \Rightarrow B$ and $T \vdash C \Rightarrow D$ then $T \vdash A \wedge C \Rightarrow B \wedge D$
+ $A \vdash \Box B \Rightarrow C \Leftrightarrow A, B \vdash C$ (Deduction theorem).
+ $\Box(\phi \Rightarrow \varphi) \Rightarrow (\Box\phi \Rightarrow \Box\varphi)$

Therefore, we can complete the above theorem if we can prove four following theorems.

Theorem 5.3.1.a $A, D1 \wedge D2 \wedge D3 \wedge D4 \vdash \lceil d_i \rceil \Rightarrow \lceil f_i \rceil$
For every interval at which d_i holds, there always exists a smallest interval that contains that one and it makes only c_i, or only s_i hold initially. Assume that c_i holds initially, we will give the brief proof of the theorem as follows.

1. $\lceil c_i \rceil ^\frown \lceil \neg d_i \rceil ^\frown \lceil d_i \rceil$ $\{assumption\}$

2. $\lceil c_i \rceil ^\frown true$ $\{1, DC^*\}$

3. $\lceil c_i \rceil ^\frown true \Rightarrow (\lceil c_i \wedge f_i \rceil ^\frown true) \wedge (\lceil c_i \wedge \neg f_i \rceil ^\frown true) \vee (\lceil c_i \wedge f_i \rceil ^\frown true) \wedge (\lceil c_i \wedge \neg f_i \rceil ^\frown true)$ $\{D1, DC{*}\}$

4. $(\lceil c_i \wedge f_i \rceil^\frown true) \wedge (\lceil c_i \wedge \neg f_i \rceil^\frown true)$
$\Rightarrow (\ell \leq \theta)^\frown(\ell = |i-j|t_m)^\frown\lceil(\ell \leq \theta^\frown\lceil d_i \rceil \wedge \lceil f_i \rceil)\rceil^\frown true \wedge tt \quad \{D2, D3, D4, A\}$
$\Rightarrow (\ell \leq \theta)^\frown(\ell = |i-j|t_m)^\frown\lceil(\ell \leq \theta^\frown\lceil d_i \rceil) \wedge \lceil f_i \rceil\rceil^\frown true \quad \{DC^*\}$
5. $(\lceil c_i \wedge f_i \rceil^\frown true) \wedge (\lceil c_i \wedge \neg f_i \rceil^\frown true)$
$\Rightarrow (\ell \leq \theta^\frown\lceil d_i \rceil) \wedge \lceil f_i \rceil^\frown true \quad \{D2, D4, D3, A3, DC^*\}$
6. $\lceil c_i \rceil^\frown true \Rightarrow (\ell \leq \theta^\frown\lceil d_i \rceil) \wedge \lceil f_i \rceil^\frown true \vee$
$(|i-j|t_m) \leq \ell \leq \theta + |i-j|t_m^\frown\lceil(\ell \leq \theta^\frown\lceil d_i \rceil) \wedge \lceil f_i \rceil\rceil^\frown true \quad \{2, 3, 4, 5\}$
7. $\lceil d_i \rceil \Rightarrow \lceil f_i \rceil \quad \{1, 2, 6, DC^*\}$
We can prove the theorem if si holds in the same way.

Theorem 5.3.1.b $A, D1 \wedge D2 \wedge D3 \wedge D4 \wedge \lceil s_i \rceil^\frown true \vdash \ell \leq t_s \vee (\ell \leq t_s^\frown\lceil d_i \rceil^\frown true)$
For every observation interval for which s_i holds initially and the interval is longer than t_s, we present the brief proof of the theorem as follows.
1. $\lceil s_i \rceil^\frown true \quad \{hypothesis\}$
2. $\lceil s_i \rceil^\frown true \Rightarrow (\lceil s_i \wedge f_i \rceil^\frown true) \vee (\lceil s_i \wedge \neg f_i \rceil^\frown true) \quad \{DC^*\}$
3. $\lceil s_i \wedge f_i \rceil^\frown true \Rightarrow \ell \leq \theta^\frown\lceil d_i \rceil^\frown true \quad \{D3\}$
4. $\lceil (s_i \vee c_i) \wedge f_i \rceil^\frown true \Rightarrow (\ell \leq \theta)^\frown\lceil motor \rceil^\frown true \quad \{D2\}$
$\Rightarrow (\ell \leq \theta)^\frown(\ell = |i-j|t_m)^\frown\ell \leq \theta^\frown\lceil d_i \rceil^\frown true \quad \{A2, D4\}$
$\Rightarrow (\ell \leq t_s)^\frown\lceil d_i \rceil^\frown true \quad \{A1, Arithmetic\}$
5. $(\ell \leq \theta)^\frown\lceil d_i \rceil^\frown true \vee (\ell \leq t_s^\frown\lceil d_i \rceil^\frown true) \quad \{1, 2, 3, 4, DC^*\}$
6. $\ell \leq t_s^\frown\lceil d_i \rceil^\frown true \quad \{5, A1, DC^*\}$

Theorem 5.3.1.c $A, D1 \wedge D2 \wedge D3 \wedge D4 \wedge \lceil c_i \rceil^\frown true \vdash \ell \leq t_s \vee (\ell \leq t_s^\frown\lceil d_i \rceil^\frown true)$
The proof of this theorem can be induced from that of theorem 5.3.1.b.

Theorem 5.3.1.d $A, D7 \wedge D6 \wedge D5 \vdash F3$
1. $\lceil \neg d_i \rceil^\frown\lceil d_i \rceil^\frown true \Rightarrow \lceil \neg d_i \rceil^\frown\ell < t_0^\frown\lceil \neg close_i \rceil^\frown true \quad \{D6\} \Rightarrow \lceil \neg d_i \rceil^\frown(\ell < t_0^\frown\ell \leq \theta) \wedge \lceil d_i \rceil^\frown true \quad \{D5\}$
2. $(\lceil \neg d_i \rceil^\frown\lceil d_i \rceil^\frown\lceil \neg d_i \rceil) \wedge (\ell \leq t_0) \Rightarrow \quad ff\{1, DC^*\}$
3. $\lceil d_i \rceil \Rightarrow \ell \leq t_{\max} \quad \{D7andDC^*\}$
4. $F3 \quad \{A, 2, 3, obs.\text{and}DC^*\}$

B. Proof of Theorem 5.4.1

We shall prove that $A_c \vdash Cont \Rightarrow Des$
Observation:

$$(\varphi \vee \phi^*)^* \Leftrightarrow (\varphi^{*\frown}\phi^*)^*$$

$$A \vdash B \Rightarrow C \Leftrightarrow A, B \vdash C$$

Because of the space litmit, we don' t give the complete proof of the theorem and we just prove the following theorem.
Theorem $A_c \vdash Cont \Rightarrow D7$

1. $\varphi^{*\frown}\phi$ {hypothesis}
2. $(\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^{*\frown}((\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2)^*)^{*\frown}(\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^* \vee \lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^\frown$
$((\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2)^*)^* \quad \{1, DC^*\}$
3. $(\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil \vee ((\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2))^{*\frown}(\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^* \vee \lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^\frown$
$((\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2)^*)^* \quad \{2, obs^*\}$
4. $(\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2$
$\Rightarrow \lceil (c_{ic} \vee s_{ic}) \wedge f_{jc} \wedge Mon\rceil \wedge \ell = \tau^\frown(|i-j|t_m \leq \ell \leq \delta + |i-j|t_m)^\frown(\lceil Dopen_i\rceil)$
$\vee \lceil (c_{ic} \vee s_{ic}) \wedge f_{ic} \wedge Dopen_i\rceil \wedge \ell = \tau^\frown((\ell \geq t_0 - \tau)^\frown\lceil close_{ic} \wedge Dclose_i\rceil)$
$\vee (t_{\max} - \theta - 1 \leq \ell \leq t_{\max} - \theta)^\frown\lceil Dclose_i\rceil \quad \{DC^*\}$
5. $\lceil Dopen_i\rceil \Rightarrow \ell \leq \tau^\frown\lceil d_i\rceil \quad \{A_c, Def.4, A_c\}$
6. $\lceil Dclose_i\rceil \Rightarrow \ell \leq \tau^\frown\lceil \neg d_i\rceil \quad \{A_c, Def.4, A_c\}$
7. $\lceil d_i\rceil \wedge (t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \quad \{ass.\}$
8. $(\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2$
$\Rightarrow (\lceil (c_{ic} \vee s_{ic}) \wedge f_{jc} \wedge Mon\rceil \wedge \ell = \tau^\frown(|i-j|t_m \leq \ell \leq \delta + |i-j|t_m)^\frown \ell \leq \tau^\frown(t_{\max} - 1 \leq \ell \leq t_{\max}) \wedge (\lceil d_i\rceil) \vee (\ell \leq \tau^\frown(t_{\max} - 1 \leq \ell \leq t_{\max}) \wedge \lceil d_i\rceil)^\frown\lceil \neg d_i\rceil \wedge (\ell \leq \delta) \quad \{4, 5, 6, 7, A_c, Rule5a, Arithmetic\}$
9. $(\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2$
$\Rightarrow (\lceil d_i\rceil \wedge (t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \Rightarrow (\ell \geq t_{\max} - 1 \wedge \ell \leq t_{\max})^\frown\lceil \neg d_i\rceil^\frown true) \quad \{8, DC^*\}$
10. $\ell = 0 \Rightarrow \Box D7 \quad \{DC^*\}$
11. $(\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil \vee \lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^* \Rightarrow \quad D7\{10, A_c, DC^*\}$
12. $A_c, ((\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2)^* \wedge \mu 1 \wedge \mu 2, (\lceil d_i\rceil \wedge t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \vdash$
$\lceil d_i\rceil^\frown\lceil \neg d_i\rceil^\frown true \quad \{9, 10, 11, DC^*, obs.\}$
13. $A_c, (\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil \vee \lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^*) \wedge \mu 1 \wedge \mu 2, (\lceil d_i\rceil \wedge t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \vdash$
$\lceil d_i\rceil^\frown\lceil \neg d_i\rceil^\frown true \quad \{11, DC^*, obs.\}$
14. $A_c, (\varphi^{*\frown}\phi) \wedge \mu 1 \wedge \mu 2, (\lceil d_i\rceil \wedge t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \vdash$
$\lceil d_i\rceil^\frown\lceil \neg d_i\rceil^\frown true \quad \{1, 2, 3, 10, 12, 13, obs., Rule9\}$
15. $PREF(\varphi^*) \quad \{hypothesis\}$
16. $(\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^{*\frown}((\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2)^*)^{*\frown}((\ell = 0 \vee \lceil \neg c_{ic} \wedge \neg s_{ic}\rceil) \vee$
$\lceil \neg c_{ic} \wedge \neg s_{ic}\rceil^{*\frown}(\varphi 1 \wedge \varphi 2) \vee \varphi 3^\frown \phi 1 \vee \phi 2)^* \quad \{13, Def.of PREF()\}$
17. $A_c, PREF(\varphi^*) \wedge \mu 1 \wedge \mu 2, (\lceil d_i\rceil \wedge t_{\max} - 1 \leq \ell \leq t_{\max})^\frown true \vdash$
$\lceil d_i\rceil^\frown\lceil \neg d_i\rceil^\frown true \quad \{10, 12, 13, DC^*, obs.\}$
18. $A_c \vdash Cont \Rightarrow D7 \quad \{14, 17, obs.\}$

Nhận bài ngày 11 - 8 - 2003