



The United Nations University

UNU/IIST

International Institute for Software Technology

From Continuous Specification to Discrete Design

François Siewe and Dang Van Hung

December 1999

UNU/IIST Report No. 182



UNU/IIST and UNU/IIST Reports

UNU/IIST (United Nations University International Institute for Software Technology) is a Research and Training Centre of the United Nations University (UNU). It is based in Macau, and was founded in 1991. It started operations in July 1992. UNU/IIST is jointly funded by the Governor of Macau and the governments of the People's Republic of China and Portugal through a contribution to the UNU Endownment Fund. As well as providing two-thirds of the endownment fund, the Macau authorities also supply UNU/IIST with its office premises and furniture and subsidise fellow accommodation.

The mission of UNU/IIST is to assist developing countries in the application and development of software technology.

UNU/IIST contributes through its programmatic activities:

1. Advanced development projects, in which software techniques supported by tools are applied,
2. Research projects, in which new techniques for software development are investigated,
3. Curriculum development projects, in which courses of software technology for universities in developing countries are developed,
4. University development projects, which complement the curriculum development projects by aiming to strengthen all aspects of computer science teaching in universities in developing countries,
5. Courses, which typically teach advanced software development techniques,
6. Events, in which conferences and workshops are organised or supported by UNU/IIST, and
7. Dissemination, in which UNU/IIST regularly distributes to developing countries information on international progress of software technology.

Fellows, who are young scientists and engineers from developing countries, are invited to actively participate in all these projects. By doing the projects they are trained.

At present, the technical focus of UNU/IIST is on formal methods for software development. UNU/IIST is an internationally recognised center in the area of formal methods. However, no software technique is universally applicable. We are prepared to choose complementary techniques for our projects, if necessary.

UNU/IIST produces a report series. Reports are either Research R, Technical T, Compendia C or Administrative A. They are records of UNU/IIST activities and research and development achievements. Many of the reports are also published in conference proceedings and journals.

Please write to UNU/IIST at P.O. Box 3058, Macau or visit UNU/IIST home page: http://www.iist.unu.edu, if you would like to know more about UNU/IIST and its report series.

Zhou Chaochen, Director — 01.8.1997 – 31.7.2001





The United Nations University

UNU/IIST

International Institute for Software Technology

P.O. Box 3058
Macau

From Continuous Specification to Discrete Design

François Siewe and Dang Van Hung

Abstract

In this report we present a syntactical approach to the design of real-time distributed systems that can handle both continuous time and discrete time models in a uniform logical framework. We approximate continuous state variables by discrete ones and formalize the relationship between them. Given a specification of the requirements of a real-time system as a formula over continuous state variables, we derive a discrete design of a digital controller as formula over discrete state variables that makes the system to satisfy the continuous specification under some assumptions about the behavior of the environment and the relationship between continuous state variables and discrete state variables. We provide rules useful for refining and verifying the correctness of an implementation syntactically.

François Siewe is a Fellow of UNU/IIST (May to December 1999), on leave from the Department of Mathematics and Computer Science, University of Dschang, Cameroon, where he is a lecturer. He got his BSc in Mathematics and his MSc and Doctorate in computer Science from the University of Yaounde I (Cameroon). His research interests include Formal Techniques of Programming, Parallel and Distributed Computing. His e-mail address at UNU/IIST is fs@iist.unu.edu and in Cameroon is fsiewe@uycdc.uninet.cm

Dang Vang Hung is a Research Fellow of UNU/IIST, on leave of absence from tne Institute of Information Technology, Hanoi, Vietnam, where he is a senior researcher. His research interests include formal methods, parallel and distributed computing. His e-mail address is dvh@iist.unu.edu

Copyright © 2000 by UNU/IIST, François Siewe and Dang Van Hung

Contents

1 Introduction **1**

2 Duration Calculus with Iteration **3**
2.1 Syntax 3
2.2 Semantics 4
2.3 Proof System of DC* 6

3 Discrete Interface **8**

4 A Technique for Design and Verification **14**
4.1 Design Process 14
4.2 Refinement and Verification Rules 15
4.3 An example 26

5 Example Water Level Controller **27**

6 Audio Control Protocol **32**

7 Conclusion **45**

1 Introduction

A real-time system can be thought of as a plant in permanent interaction with its environment. However, the behaviour of the plant should satisfy specific critical-time constraints to face environment disturbances. Very often, a slight failure in the reaction of the plant can cause harm or damage. That is why formal methods are needed for specifying and reasoning about real-time system. Various formal approaches for specifying and reasoning about real-time system behaviours have been proposed.

In this report we focus on the Duration Calculus (DC for short) approach, introduced in 1991 by Zhou, Hoare and Ravn [8]. DC is a logic for reasoning about *state durations* which is based on Interval Temporal Logic. The concept of state duration is the essential measurements of the behaviour of real-time systems [13]. A *state* is modeled as a boolean valued function over time (real numbers), having at most finite number of discontinuity points in any finite interval (finite variability). So, the duration of a state over a time-interval is the integral of the state over the time-interval. The approach is powerful enough for specifying and reasoning about the behaviour of real-time systems in continuous time model. But what about their implementation?

A specification of a real-time system is given in continuous time model. But an implementation should be given in discrete time since the state of a digital program changes in discrete time at ticks of a computer clock. A sampling semantics of DC is given in [19] defining a discrete approximation of the semantics of DC formulae.

In this work, we intended to give a syntactical approach for designing a real-time system specified as a DC formula. Such an approach is suitable for both refining a specification of a real-time system towards an implementation and verifying the correctness of the implementation with respect to the specification using proof assistant tools. Our model of real-time hybrid system is a distributed system depicted as in Figure 1. The states of the environment and the plant change continuously according to the laws of physics. We use *sensors* to sample continuous states and *actuators* to control the plant. The *control program* gets at discrete time points the states of the environment via the sensors, computes a command and imposes it to the plant via the actuators. We assume that the computation time is negligible. As there is no global clock in distributed systems, the concept of discreteness is a local property in our model. That is a state variable might be considered as discrete in one component of the system and as continuous in another as the two components do not have the same clock. Therefore we partition the domain $Svar$ of all state variables into two partitions (when there are several processes in the system, according to the view of one process): $cSvar$ of continuous state variables and $dSvar$ of discrete state variable relatively to the control program. Then we consider the relationship between them, defining the approximation of continuous states by discrete ones. Our goal is to derive a 'discrete' specification of the control program from the specification of the continuous behaviour of the plant and the environment. The design is correct if the 'discrete' specification implies the specification of the continuous behaviour of the plant under the assumption of the behaviour of the environment and the relationship between continuous states and discrete states. Hence we can reason about the design and the correctness of an implementation in a uniform manner.

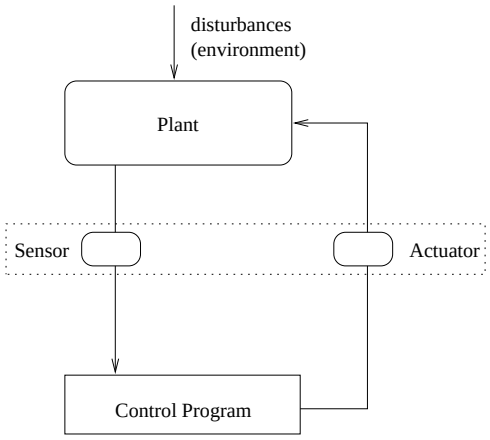


Figure 1: A model of controlled system

Duration Calculus with Iteration (DC^* for short), a logic obtained by extending DC with the iteration operator (*), has been defined in [20] to play the role of interface between DC and *Timed Automata* model [1]. A sound proof system for DC^* has been proposed. The completeness of the proof system is ensured for a restricted class of DC^* formulae called *Simple DC^* (SDC^* for short) formulae*. In [20] it is also shown that a simple DC^* formula denotes exactly a *Timed Regular Expression*. Eugene Asarin et al have proved in [3] that from a timed regular expression one can build a timed automaton that recognises exactly the model of the timed regular expression. Our design technique is based on those results. The requirement of a real-time system is specified as a DC formula over $cSvar$. The design decision is formalised as a DC^* formula over $cSvar$ and the implementation is given as a SDC^* formula over $dSvar$. The design process is formulated as follows. For a specification S of the requirement of a real-time system, at the first step a design decision must be taken as how the requirement of the real-time system is to be met and formalised as a DC^* formula D over $cSvar$ such that $D \Rightarrow S$. Then we find a SDC^* formula I over $dSvar$ such that $A \vdash I \Rightarrow D$ where A stands for the assumption of the behaviour of the environment and the relationship between continuous state and discrete state variables. The discrete SDC^* formula I is the specification of the control program. We provide rules for deriving I and verifying the correctness of an implementation syntactically.

To illustrate our discretization technique, let us consider a classical simple example, Gas Burner, taken from [13]. The critical real-time requirement of the gas burner is that the proportion of time spent in the leak state is not more than one twentieth of the elapsed time, whenever the burner is observed for at least one minute. This requirement can be formalised as a DC formula $S_{gb} \mathrel{\widehat{=}} \Box(\ell \geq 60 \Rightarrow 20 * \int leak \leq \ell)$. We consider the design decision that any occurrence of leak should not last for a period longer than one second, and the distance between two consecutive occurrences of leaks must be at least thirty seconds long. We formalise the design decision as a DC^* formula $D_{gb} \mathrel{\widehat{=}} \Box(\lceil\!\lceil leak \rceil\!\rceil \Rightarrow \ell \leq 1) \wedge \Box(\lceil\!\lceil leak \rceil\!\rceil^\frown \lceil\!\lceil \neg leak \rceil\!\rceil^\frown \lceil\!\lceil leak \rceil\!\rceil \Rightarrow \ell \geq 30)$. It is proved in [8] that $D_{gb} \Rightarrow S_{gb}$. Let us consider a discrete state variable $leak'$ used by the control program to observe the presence or absence of leak via the sensor. Let us assume that a presence of leak can be observed within δ time units ($\delta < 1$), which could follow the fact that leak is stable for

more than δ time units, and the sampling step is at most δ. This implies that $leak'$ approximate $leak$ with the tolerance δ. Let $rocover$ be a discrete state which can stop leak within $1-2\delta$ time units and keeps the Burner in $nonleak$ state as long as it is true. Then a discrete specification of the control program is defined as

$$\begin{array}{lcl} I'_{gb} & \widehat{=} & \Box(\lceil leak' \rceil \Rightarrow \ell \leq 1-\delta) \wedge \\ & & \Box\left(\begin{array}{r} \lceil leak' \rceil ^\frown \lceil \neg leak' \rceil ^\frown \lceil leak' \rceil \Rightarrow \\ \ell > 30+\delta \end{array} \right) \end{array}$$

I'_{gb} in turn is refined by $I_{gb} \widehat{=} PREF(\lceil \neg leak' \rceil ^\frown (\lceil leak' \wedge recover \rceil \wedge 1-2\delta \leq \ell \leq 1-\delta) ^\frown (\lceil recover \rceil \wedge \ell = 31+\delta) ^\frown (\lceil \neg leak' \rceil \vee \ell = 0)^*)$, where for any DC formula F, $PREF(F)$ denote the formula satisfied by all the prefix of any time interval satisfying F. We prove that $I_{gb} \Rightarrow D_{gb}$ under above assumptions about the relationship between $leak$ and $leak'$, $recover$ and $nonleak$, and the assumption about the stability of $leak$. Note that only 'discrete' state variables can occur in I_{gb}, and I_{gb} can be translated to a timed automaton in the obvious way.

The remaining of the paper is organised as follows. In Section 2 we recall the syntax, the semantics and the proof system of DC^*. Section 3 presents our discretization technique. The design technique is presented in Section 4. In Section 5 we design and verify a water level control system. In Section 6 our design technique is used to formalise and verify a communication protocol, the audio control protocol. Section 7 concludes the paper.

2 Duration Calculus with Iteration

In this section we give the syntax, the semantics and the proof system of DC^*. The readers are referred to [20] for more details on the completeness and the decidability of the calculus.

2.1 Syntax

A language for DC* is built starting from the following sets of *symbols*: a set of *constant symbols* $\{a, b, c, \ldots\}$, a set of *individual variables* $\{x, y, z, \ldots\}$, a set of *state variables* $\{P, Q, \ldots\}$, a set of *temporal variables* $\{u, v, \ldots\}$, a set of *function symbols* $\{f, g, \ldots\}$, a set of *relation symbols* $\{R, U, \ldots\}$, and a set of *temporal propositional letters* $\{A, B, \ldots\}$. These sets are required to be pairwise disjoint and disjoint with the set $\{\mathbf{0}, \neg, \vee, ^\frown, ^*, \exists, \int, (,)\}$. Besides, 0 should be one of the constant symbols; $+$ should be a binary function symbol; $=$ and $\leq$ should be binary relation symbols.

Given the sets of symbols, a DC* language definition is essentially that of the sets of *state expressions* S, *terms* t and *formulae* φ of the language. These sets can be defined by the

following BNFs:

$$
\begin{array}{l}
S \mathrel{\widehat{=}} \mathbf{0} \mid P \mid \neg S \mid S \vee S \\
t \mathrel{\widehat{=}} c \mid x \mid u \mid \int S \mid f(t, \ldots, t) \\
\varphi \mathrel{\widehat{=}} A \mid R(t, \ldots, t) \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi^\frown\varphi) \mid (\varphi^*) \mid \exists x \varphi
\end{array}
$$

Terms and formulae that have no occurrences of $^\frown$ (*chop*), nor of temporal variables, or $\int$, are called *rigid*.

2.2 Semantics

The linearly ordered field of the real numbers,

$$\langle \mathbf{R}, =_{\mathbf{R}}, 0_{\mathbf{R}}, 1_{\mathbf{R}}, +_{\mathbf{R}}, -_{\mathbf{R}}, \times_{\mathbf{R}}, /_{\mathbf{R}}, \leq_{\mathbf{R}} \rangle ,$$

is the most important component of DC semantics. We denote by $\mathbf{I}$ the set of the bounded intervals over $\mathbf{R}$, $\{[\tau_1, \tau_2] \mid \tau_1, \tau_2 \in \mathbf{R}, \tau_1 \leq_{\mathbf{R}} \tau_2\}$. For a set $A \subseteq \mathbf{R}$, we denote by $\mathbf{I}(A)$ the set $\{[\tau_1, \tau_2] \in \mathbf{I} \mid \tau_1, \tau_2 \in A\}$ of intervals with end-points in A.

Given a DC* language $\mathcal{L}$, a model for $\mathcal{L}$ is an *interpretation* $\mathcal{I}$ of the symbols of $\mathcal{L}$ that satisfies the following conditions: $\mathcal{I}(c), \mathcal{I}(x) \in \mathbf{R}$ for constant symbols c and individual variables x; $\mathcal{I}(f) : \mathbf{R}^n \to \mathbf{R}$ for n-place function symbols f; $\mathcal{I}(v) : \mathbf{I} \to \mathbf{R}$ for temporal variables v; $\mathcal{I}(R) : \mathbf{R}^n \to \{0, 1\}$ for n-place relation symbols R; $\mathcal{I}(P) : \mathbf{R} \to \{0, 1\}$ for state variable P, and $\mathcal{I}(A) : \mathbf{I} \to \{0, 1\}$ for temporal propositional letters A. Besides, $\mathcal{I}(0) = 0_{\mathbf{R}}$, $\mathcal{I}(+) = +_{\mathbf{R}}$, $\mathcal{I}(=)$ is $=_{\mathbf{R}}$, and $\mathcal{I}(\leq)$ is $\leq_{\mathbf{R}}$. The following condition, known as *finite variability of state*, is imposed on interpretations: *For every* $[\tau_1, \tau_2] \in \mathbf{I}$ *such that* $\tau_1 < \tau_2$, *and every state variable* S *there exist* $\tau'_1, \ldots, \tau'_n \in \mathbf{R}$ *such that* $\tau_1 = \tau'_1 < \ldots < \tau'_n = \tau_2$ *and* $\mathcal{I}(S)$ *is constant on the intervals* (τ'_i, τ'_{i+1}), $i = 1, \ldots, n-1$.

For the rest of the paper we omit the index $._{\mathbf{R}}$, that distinguishes operations on reals from the corresponding symbols.

Definition 1 *Given a DC interpretation* $\mathcal{I}$ *for the DC* language* $\mathcal{L}$, *the meaning of state expressions* S *in* $\mathcal{L}$ *under* $\mathcal{I}$, $S_{\mathcal{I}} : \boldsymbol{R} \to \{0, 1\}$, *is defined inductively as follows: for all* $\tau \in \boldsymbol{R}$

$$
\begin{array}{ll}
\mathbf{0}_{\mathcal{I}}(\tau) \mathrel{\widehat{=}} 0 & \\
P_{\mathcal{I}}(\tau) \mathrel{\widehat{=}} \mathcal{I}(P)(\tau) & \textit{for state variables } P \\
(\neg S)_{\mathcal{I}}(\tau) \mathrel{\widehat{=}} 1 - S_{\mathcal{I}}(\tau) & \\
(S_1 \vee S_2)_{\mathcal{I}}(\tau) \mathrel{\widehat{=}} \max((S_1)_{\mathcal{I}}(\tau), (S_2)_{\mathcal{I}}(\tau)) &
\end{array}
$$

Given an interval $[\tau_1, \tau_2] \in \mathbf{I}$*, the meaning of a term* t *in* $\mathcal{L}$ *under* $\mathcal{I}$ *is a number* $\mathcal{I}^{\tau_2}_{\tau_1}(t) \in \mathbf{R}$ *defined inductively as follows:*

$$
\begin{array}{ll}
\mathcal{I}^{\tau_2}_{\tau_1}(c) \mathrel{\widehat{=}} \mathcal{I}(c) & \textit{for constant symbols } c, \\
\mathcal{I}^{\tau_2}_{\tau_1}(x) \mathrel{\widehat{=}} \mathcal{I}(x) & \textit{for individual variables } x, \\
\mathcal{I}^{\tau_2}_{\tau_1}(v) \mathrel{\widehat{=}} \mathcal{I}(v)([\tau_1, \tau_2]) & \textit{for temporal variables } v, \\
\mathcal{I}^{\tau_2}_{\tau_1}(\int S) \mathrel{\widehat{=}} \int_{\tau_1}^{\tau_2} S_{\mathcal{I}}(\tau) d\tau & \textit{for state expressions } S, \\
\mathcal{I}^{\tau_2}_{\tau_1}(f(t_1, \ldots, t_n)) \mathrel{\widehat{=}} \mathcal{I}(f)(\mathcal{I}^{\tau_2}_{\tau_1}(t_1), \ldots, \mathcal{I}^{\tau_2}_{\tau_1}(t_n)) & \textit{for } n\textit{-place function} \\
 & \textit{symbols } f.
\end{array}
$$

The definitions given so far are relevant to the semantics of DC in general. The extension to the semantics that comes with DC* appears in the definition of the $\models$ relation below. Let us recall the traditional relation on interpretations: For interpretations $\mathcal{I}$, and $\mathcal{J}$ of the symbols of the same DC* language $\mathcal{L}$ and for a symbol x in $\mathcal{L}$, we say $\mathcal{I}$ *x-agrees* with $\mathcal{J}$ iff $\mathcal{I}(s) = \mathcal{J}(s)$ for all symbols s in $\mathcal{L}$, but possibly x.

Definition 2 *Given a DC* language* $\mathcal{L}$*, and an interpretation* $\mathcal{I}$ *of the symbols of* $\mathcal{L}$*. The relation* $\mathcal{I}, [\tau_1, \tau_2] \models \varphi$ *for* $[\tau_1, \tau_2] \in \mathbf{I}$ *and formulae* φ *in* $\mathcal{L}$ *is defined by induction on the construction of* φ *as follows:*

$$
\begin{array}{ll}
\mathcal{I}, [\tau_1, \tau_2] \not\models \mathit{false} & \\
\mathcal{I}, [\tau_1, \tau_2] \models A & \textit{iff } \mathcal{I}(A)([\tau_1, \tau_2]) = 1 \textit{ for temporal} \\
 & \textit{propositional letters } A \\
\mathcal{I}, [\tau_1, \tau_2] \models R(t_1, \ldots, t_n) & \textit{iff } \mathcal{I}(R)(\mathcal{I}^{\tau_2}_{\tau_1}(t_1), \ldots, \mathcal{I}^{\tau_2}_{\tau_1}(t_n)) = 1 \\
\mathcal{I}, [\tau_1, \tau_2] \models \neg\varphi & \textit{iff } \mathcal{I}, [\tau_1, \tau_2] \not\models \varphi \\
\mathcal{I}, [\tau_1, \tau_2] \models (\varphi \vee \psi) & \textit{iff either } \mathcal{I}, [\tau_1, \tau_2] \models \varphi \textit{ or } \mathcal{I}, [\tau_1, \tau_2] \models \psi \\
\mathcal{I}, [\tau_1, \tau_2] \models (\varphi^\frown\psi) & \textit{iff } \mathcal{I}, [\tau_1, \tau] \models \varphi \textit{ and } \mathcal{I}, [\tau, \tau_2] \models \psi \\
 & \textit{for some } \tau \in [\tau_1, \tau_2] \\
\mathcal{I}, [\tau_1, \tau_2] \models (\varphi^*) & \textit{iff either } \tau_1 = \tau_2\textit{, or there exist } \tau'_1, \ldots, \tau'_n \in \mathbf{R} \\
 & \textit{such that } \tau_1 = \tau'_1 < \ldots < \tau'_n = \tau_2 \textit{ and} \\
 & \mathcal{I}, [\tau'_i, \tau'_{i+1}] \models \varphi \textit{ for } i = 1, \ldots, n-1 \\
\mathcal{I}, [\tau_1, \tau_2] \models \exists x\varphi & \textit{iff } \mathcal{J}, [\tau_1, \tau_2] \models \varphi \textit{ for some } \mathcal{J} \textit{ that} \\
 & x\textit{-agrees with } \mathcal{I}
\end{array}
$$

Note that the clauses that define the interpretation of constructs other than * in DC* are the same as in DC. This entails that DC* is a conservative extension of DC.

In the rest of the paper for the convenience of reading, we use the following conventions. We use the customary *infix* notation for terms with $+$, and formulae with $\leq$ and $=$ occurring in them. We introduce the boolean connectives $\wedge$, $\Rightarrow$ and $\Leftrightarrow$, the relation symbols $\neq$, $\geq$, $<$ and $>$, and the $\forall$ quantifier as abbreviations in the usual way. We assume that boolean connectives

bind more tightly than $\frown$. Since $\frown$ is associative, we omit parentheses in formulae that contain consecutive occurrences of $\frown$. Besides, we use the following abbreviations, that are generally accepted in Duration Calculus:

$$\begin{array}{ll} \mathbf{1} \mathrel{\widehat{=}} \neg\mathbf{0} & \ell \mathrel{\widehat{=}} \int \mathbf{1} \\ [\![S]\!] \mathrel{\widehat{=}} \int S = \ell \wedge \ell \neq 0 & \Diamond\varphi \mathrel{\widehat{=}} true \frown \varphi \frown true \\ \Box\varphi \mathrel{\widehat{=}} \neg\Diamond\neg\varphi & (\varphi^+) \mathrel{\widehat{=}} \varphi \frown (\varphi^*) \\ \varphi^0 \mathrel{\widehat{=}} \ell = 0 & \varphi^k \mathrel{\widehat{=}} \underbrace{\varphi \frown \ldots \frown \varphi}_{k \text{ times}} \quad \text{for } k > 0 \end{array}$$

2.3 Proof System of DC*

The proof system for DC* consists of a complete Hilbert-style proof system for first order logic (cf. e.g. [25]), axioms and rules for interval logic (cf. e.g. [9]), Duration Calculus axioms and rules ([12]) and axioms about iteration ([20]). We assume that the readers are familiar with Hilbert-style proof systems for first order logic and do not give one here. Here follow the interval logic and DC-specific axioms and rules.

Axioms and rules for Interval Logic

$$\begin{array}{ll} (A1_l) & (\varphi \frown \psi) \wedge \neg(\chi \frown \psi) \Rightarrow (\varphi \wedge \neg\chi \frown \psi) \\ (A1_r) & (\varphi \frown \psi) \wedge \neg(\varphi \frown \chi) \Rightarrow (\varphi \frown \psi \wedge \neg\chi) \\ (A2) & ((\varphi \frown \psi) \frown \chi) \Leftrightarrow (\varphi \frown (\psi \frown \chi)) \\ (R_l) & (\varphi \frown \psi) \Rightarrow \varphi \text{ if } \varphi \text{ is rigid} \\ (R_r) & (\varphi \frown \psi) \Rightarrow \psi \text{ if } \psi \text{ is rigid} \\ (B_l) & (\exists x \varphi \frown \psi) \Rightarrow \exists x(\varphi \frown \psi) \text{ if } x \text{ is not free in } \psi \\ (B_r) & (\varphi \frown \exists x \psi) \Rightarrow \exists x(\varphi \frown \psi) \text{ if } x \text{ is not free in } \varphi \\ (L1_l) & (\ell = x \frown \varphi) \Rightarrow \neg(\ell = x \frown \neg\varphi) \\ (L1_r) & (\varphi \frown \ell = x) \Rightarrow \neg(\neg\varphi \frown \ell = x) \\ (L2) & \ell = x + y \Leftrightarrow (\ell = x \frown \ell = y) \\ (L3_l) & \varphi \Rightarrow (\ell = 0 \frown \varphi) \\ (L3_r) & \varphi \Rightarrow (\varphi \frown \ell = 0) \end{array}$$

$$(N_l) \quad \frac{\varphi}{\neg(\neg\varphi \frown \psi)} \qquad (N_r) \quad \frac{\varphi}{\neg(\psi \frown \neg\varphi)}$$

$$(Mono_l) \quad \frac{\varphi \Rightarrow \psi}{(\varphi \frown \chi) \Rightarrow (\psi \frown \chi)} \qquad (Mono_r) \quad \frac{\varphi \Rightarrow \psi}{(\chi \frown \varphi) \Rightarrow (\chi \frown \psi)}$$

Duration Calculus axioms and rules

$(DC1)$ $\int \mathbf{0} = 0$
$(DC2)$ $\int \mathbf{1} = \ell$
$(DC3)$ $\int S \geq 0$
$(DC4)$ $\int S_1 + \int S_2 = \int (S_1 \vee S_2) + \int (S_1 \wedge S_2)$
$(DC5)$ $(\int S = x^\frown \int S = y) \Rightarrow \int S = x + y$
$(DC6)$ $\int S_1 = \int S_2$ if $S_1 \Leftrightarrow S_2$ in propositional calculus.

$$(IR_1) \quad \frac{[\ell = 0/A]\varphi \ \varphi \Rightarrow [A^\frown [\![S]\!]/A]\varphi \ \varphi \Rightarrow [A^\frown [\![\neg S]\!]/A]\varphi}{[true/A]\varphi}$$

$$(IR_2) \quad \frac{[\ell = 0/A]\varphi \ \varphi \Rightarrow [[\![S]\!]^\frown A/A]\varphi \ \varphi \Rightarrow [[\![\neg S]\!]^\frown A/A]\varphi}{[true/A]\varphi}$$

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

Axioms about iteration

(DC_1^*) $\ell = 0 \Rightarrow \varphi^*$
(DC_2^*) $(\varphi^* {}^\frown \varphi) \Rightarrow \varphi^*$
(DC_3^*) $(\varphi^* \wedge \psi^\frown true) \Rightarrow (\psi \wedge \ell = 0^\frown true) \vee (((\varphi^* \wedge \neg\psi^\frown \varphi) \wedge \psi)^\frown true)$.

A restriction is made on the application of first order logic rules and axioms that involve substitution: $[t/x]\varphi$ is defined if no variable in t becomes bound due to the substitution, and either t is rigid or $^\frown$ does not occur in φ.

In order to simplify proofs in DC, the following theorem, known as *deduction theorem*, has been established [13].

Theorem 1 *Let ? be a set of DC formulae and ϕ and ψ, DC formulae.*

$$?, \phi \vdash \psi \ \textit{implies} \ ? \vdash \Box\phi \Rightarrow \psi$$

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

Definition 3 *A Simple DC^* formula is defined inductively by:*

$$\varphi \mathrel{\widehat{=}} \ell = 0 \mid [\![S]\!] \mid a \leq \ell \mid \ell \leq a \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi^\frown \varphi) \mid \varphi^*$$

For simplifying the presentation in the next section, we introduce the notion $PREF(D)$ for any simple formula D. Intuitively, $PREF(D)$ is a simple formula that holds for all prefixes of an interval that validates D.

Definition 4 *Given a simple DC* formula φ, we define a simple DC* formula $PREF(\varphi)$ as follows.*

1. $PREF(\llbracket S \rrbracket) \mathrel{\widehat{=}} \llbracket S \rrbracket^*$
2. $PREF(a \leq \int \mathbf{1}) \mathrel{\widehat{=}} \int \mathbf{1} \geq 0$
3. $PREF(\int \mathbf{1} \leq a) \mathrel{\widehat{=}} \int \mathbf{1} \leq a$
4. $PREF(\varphi \vee \varphi') \mathrel{\widehat{=}} PREF(\varphi) \vee PREF(\varphi')$
5. $PREF(\varphi \wedge \varphi') \mathrel{\widehat{=}} PREF(\varphi) \wedge PREF(\varphi')$
6. $PREF(\varphi^\frown\varphi') \mathrel{\widehat{=}} PREF(\varphi) \vee (\varphi^\frown PREF(\varphi'))$
7. $PREF(\varphi^*) \mathrel{\widehat{=}} \varphi^* {}^\frown PREF(\varphi)$

It follows immediately from the definition that

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

The class of simple DC^* formulae plays an important role in our design technique presented in section 4. The following section presents our discretization technique.

3 Discrete Interface

In this section we formalise the relationship between continuous state variables and discrete ones. Our concept of discreteness is more general. A distributed system may comprise several digital components running independently at the ticks of local clocks. So, the discreteness of a state variable is relative to a given clock. Indeed, if P and P' are two processes of the system running asynchronously under different clocks then a discrete variable relatively to P (i.e changes only at ticks of the clock of P) is considered as continuous in P'. Figure 2 illustrates our idea. The state s is discrete according to the process P, and continuous according to process P'. But the state r is continuous for both processes P and P'.

We use discrete state variables in digital components of the system to approximate continuous ones. Our concept of approximation means any change of the continuous state variable is shifted in the discrete state variable. Therefore the continuous state should not change very fast compare to the speed of the sensor and the controller. So, we require some environment states to be stable enough to be controllable by a digital program. In general, there is no way to control the stability of environment states, viz an occurrence of a state should last for at least some times. In our approach we only make the assumption about it. The assumption on the

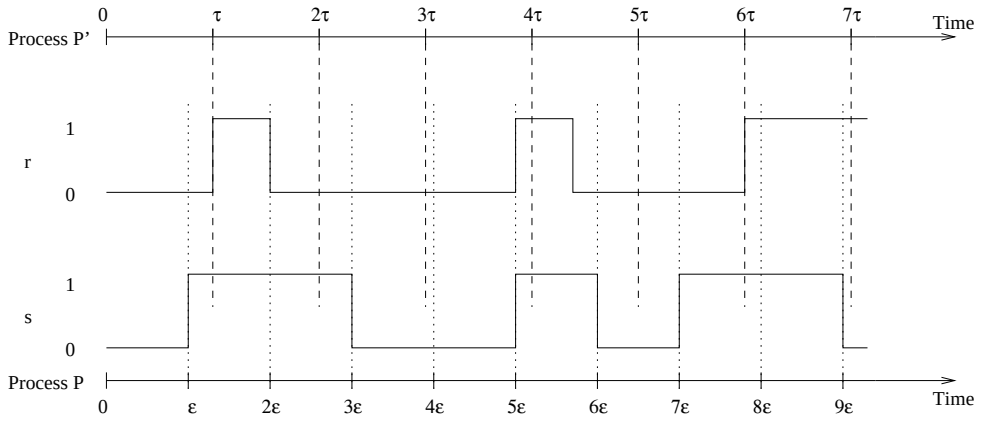
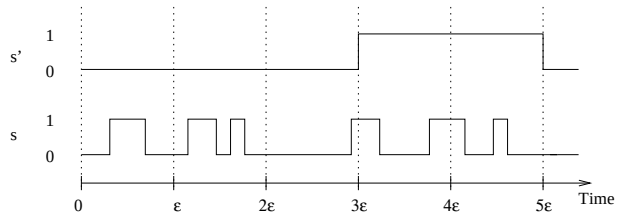


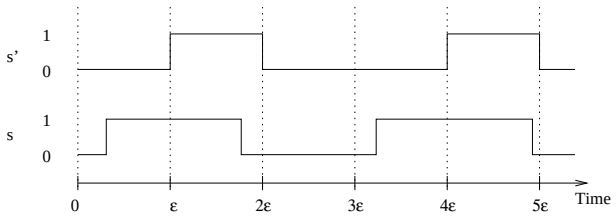
Figure 2: The state s is discrete according to P, and continuous according to P'. The state r is continuous for both P and P'

stability of continuous state variables is capital in our discretization technique as illustrated in Figure 3. In Figure 3(a), the continuous state s changes very fast. So, some changes in s are not observed in discrete time. Therefore its discrete approximation s' is very loose. Figure 3(b) shows that if the presence and absence of a state s are stable for at least the sampling step then any change of s is observable in discrete time.



(a)

s changes so fast. So, some changes in s are not observed in discrete time.



(b)

s and $\neg s$ are ϵ-stable. So, any changes in s is observed in discrete time.

Figure 3: Stability and discretization (s' is a sampling of s)

We also model a control program command as a discrete state variable which is *true* (*false*) when the command is on (off respectively). The following definitions formalise the relationship between continuous states and (relative) discrete states. We first formalise the concept of *stability* as illustrated in Figure 4.

Definition 5 (Stability property) *Given a state variable s and a positive real number δ, we say s is δ-stable iff the following formula is satisfied by any interval*

$$\delta\text{-}stable(s) \mathrel{\widehat{=}} \Box([\![\neg s]\!]^\frown[\![s]\!]^\frown[\![\neg s]\!]) \Rightarrow [\![\neg s]\!]^\frown([\![s]\!] \wedge \ell > \delta)^\frown[\![\neg s]\!]$$

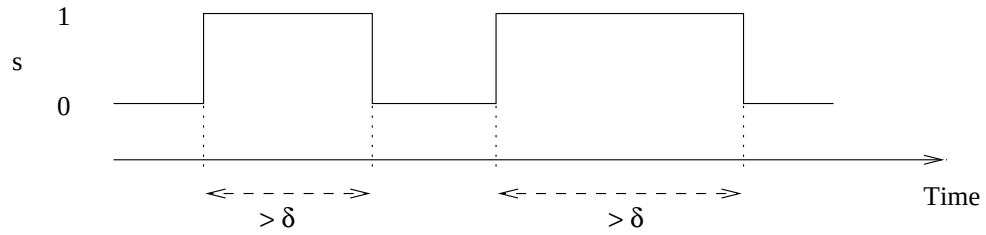


Figure 4: The state s is δ-stable

The formula δ-stable(s) says that s must last for more than δ time units whenever it becomes *true* (Figure 4). Such property is required for state variables to be controllable. Moreover the concept of stability can equally be used to capture useful properties of real-time systems.

Example The property that the distance between two consecutive occurrences of a state s must be greater than δ time units is formalised by δ-stable($\neg s$).

We also consider the concept of *control* for formalising the relationship between state variables. The intuition is that when the controller issues a command, the effect of the actuator may take some time to change the state of the plant. This concept is illustrated in Figure 5. We use white space to represent an undetermined behaviour of a state in a time-interval.

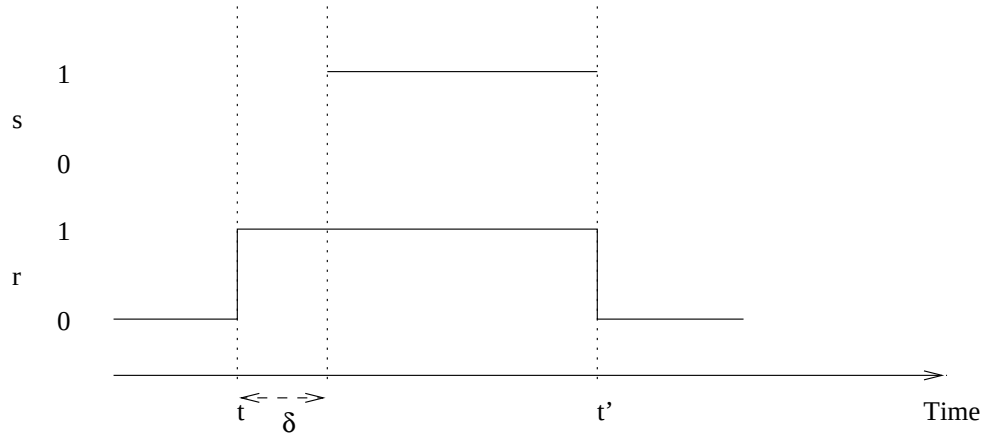


Figure 5: The state r δ-controls the state s

Definition 6 (Control property) *Given two state variables r and s, and a non-negative real number δ, we say r δ-controls s iff the following formula is satisfied by any interval*

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

This formula asserts that whenever r lasts for a period longer than δ time units, s must hold and last until at least r changes (Figure 5). The concept of control is useful for modelling the

behaviour of actuators. We also use this concept for reasoning about the time-bounded response property.

Example

- If $r \rhd_\delta \neg s$ then any occurrence of r stable enough 'stops' s within δ time units.
- If $s \rhd_\delta \neg s$ then any occurrence of s does not last for a period longer than δ time units.

Note that Definition 6 says nothing about the behaviour of s when r occurs but disappears after a period less than or equal to δ time units. This make our concept of control not strong enough for reasoning about occurrences of events as they might last for a very short time. So we introduce the concept of *strong control* suitable for modelling and reasoning about occurrences of events. We also formalise an event by a state variable. Figure 6 is an illustration of this concept.

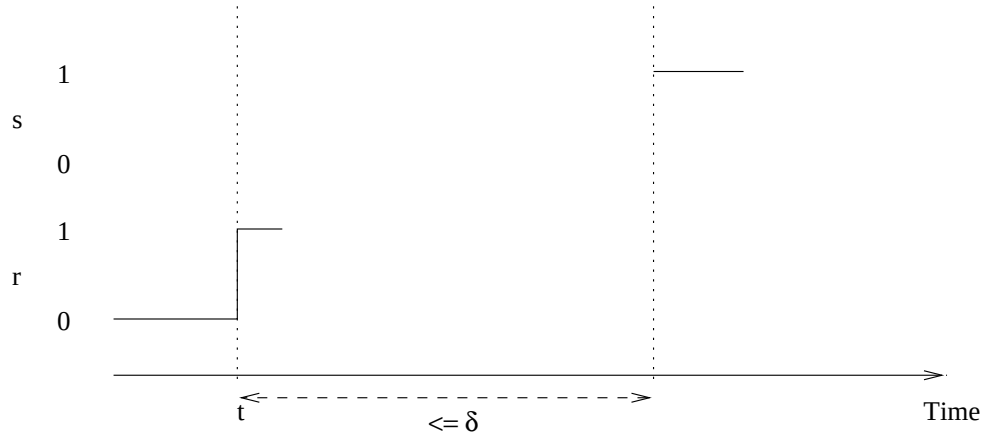


Figure 6: The state r strongly δ-controls the state s

Definition 7 (Strong Control property) *Given two state variables r and s, and a non-negative real number δ, we say r* strongly δ-controls *s iff the following formula is satisfied by any interval*

$$r \trianglerighteq_\delta s \mathrel{\widehat{=}} \Box((\llbracket r \rrbracket^\frown true) \wedge \ell > \delta \Rightarrow (\ell \leq \delta)^\frown \llbracket s \rrbracket^\frown true)$$

This formula says that whenever r holds, s must hold within δ time units (time-bounded response property). It does not need that the event r lasts for some times, say δ time units, neither that s holds as long as r does, as required in Definition 6. However the two concepts of control are related by the following theorem.

Theorem 2 *Let r and s be two state variables. If δ-stable(r) and $r \rhd_\delta s$ then $\llbracket \neg r \rrbracket^\frown true \Rightarrow r \trianglerighteq_\delta s$.*

Proof (Theorem 2).

1	$\Box(\lceil r \rceil \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \lceil s \rceil)$	assumption
2	$\Box(\lceil \neg r \rceil^\frown \lceil r \rceil^\frown \lceil \neg r \rceil \Rightarrow \lceil \neg r \rceil^\frown (\lceil r \rceil \wedge \ell > \delta)^\frown \lceil \neg r \rceil)$	stability of r
3	$true \Rightarrow (\lceil \neg r \rceil \vee \lceil r \rceil)^*$	finite variability
4	$\lceil \neg r \rceil^\frown true \Rightarrow \lceil \neg r \rceil^\frown (\lceil \neg r \rceil \vee \lceil r \rceil)^*$	by 3, DC^*
5	$\lceil \neg r \rceil^\frown true \Rightarrow \lceil \neg r \rceil^\frown (\lceil \neg r \rceil \vee (\lceil r \rceil \wedge \ell > \delta))^{*\frown}$ $((\lceil \rceil \vee \lceil r \rceil) \wedge \ell \leq \delta)$	by 2, 4, DC^*
6	$\lceil \neg r \rceil^\frown true \Rightarrow \Box((\lceil r \rceil^\frown true) \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \lceil s \rceil^\frown true)$	by 1, 5, DC^*

□

Finally we define the concept of *observation* as depicted in Figure 7.

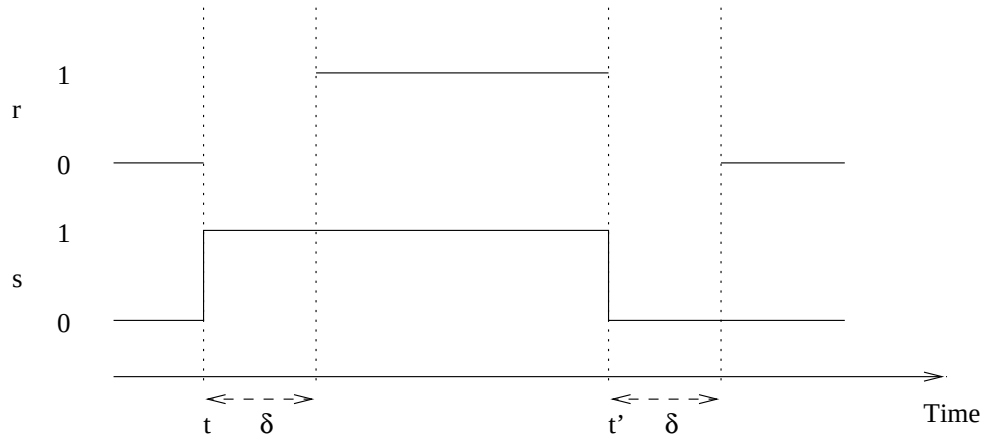


Figure 7: The state r δ-observes the state s

Definition 8 (Observation property) *Given two state variables r and s, and a non-negative real number δ, we say r δ-observes s iff the following formula is satisfied by any interval*

$$r \overrightarrow{\approx}_\delta s \mathrel{\widehat{=}} (s \rhd_\delta r) \wedge (\neg s \rhd_\delta \neg r)$$

This formula says that any change in s, stable for at least δ time units, occurs in r within δ time units, and r keeps the new value until at least s changes again. For two consecutive discontinuity time points t and t' ($t' > t + \delta$) of state s (Figure 7), we do not know how r behaves in between t and $t + \delta$ nor how it behaves in between t' and $t' + \delta$ (indicated by a white space in the figure). But we do know that r keeps the value of s from $t + \delta$ to t' at least.

The concept of observation can be used to capture several properties. Following are some examples.

Formalising the discrete approximation of state

The observation property together with the stability property defined above can be used to formalise the approximation of continuous state variables by discrete ones. Indeed, if r δ-observes s, if s and $\neg s$ are δ-stable, and if r (or $\neg r$) is $\frac{\delta}{2}$-stable then any change in s occurs in r within at most δ time units, and r does not change otherwise. In this case we say that r is a sampling of s with sampling step less than or equal to δ. Figure 8 is an illustration of this property.

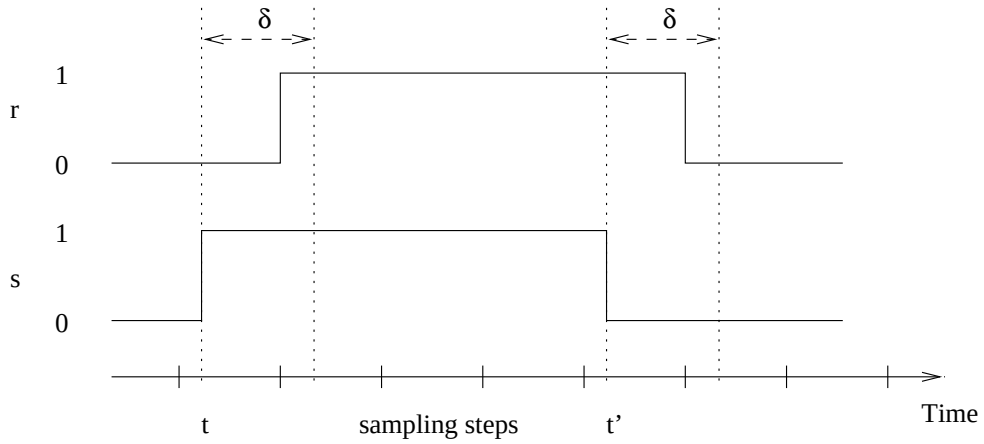


Figure 8: The state r is a sampling of state s

Modelling communication at physical layer

We consider a model of communication at the physical layer in which a sender and a receiver are connected via a bus. The clocks of the sender and the receiver are running at different rates. We model by a state variable s the signal sent by the sender (Figure 9). We refer to the clock of the sender as a reference time. The receiver receives signals by digitising. Let r be a state variable modelling the signal obtained by the receiver. Without loss of generality, assume that the delay between the sender and the receiver is 0. Physically, due to some disturbance, it takes some amount of time to the signal on the bus to change from high to low or vice-versa [23, 24]. So the value of r is chosen arbitrarily among 0 and 1 if the digitised value is neither 0 nor 1. Therefore r is unreliable when the analog signal changes from high to low or from low to high. This is represented in Figure 9 by a white space in the corresponding time-intervals. Let us assume that when s changes, r is unreliable for at most δ cycles of the sender's clock. Then the relationship between the receiver's signal r and the sender's signal s can be formalised by $r \overrightarrow{\approx}_{\delta} s$. However the signal sent and the signal received are different because the sender and the received are not synchronised. Communication protocols are developed to deal with the problem. In section 6, we use our approach to specify and verify the well-known Audio Control Protocol [7]

for asynchronous communication between two digital hardware devices.

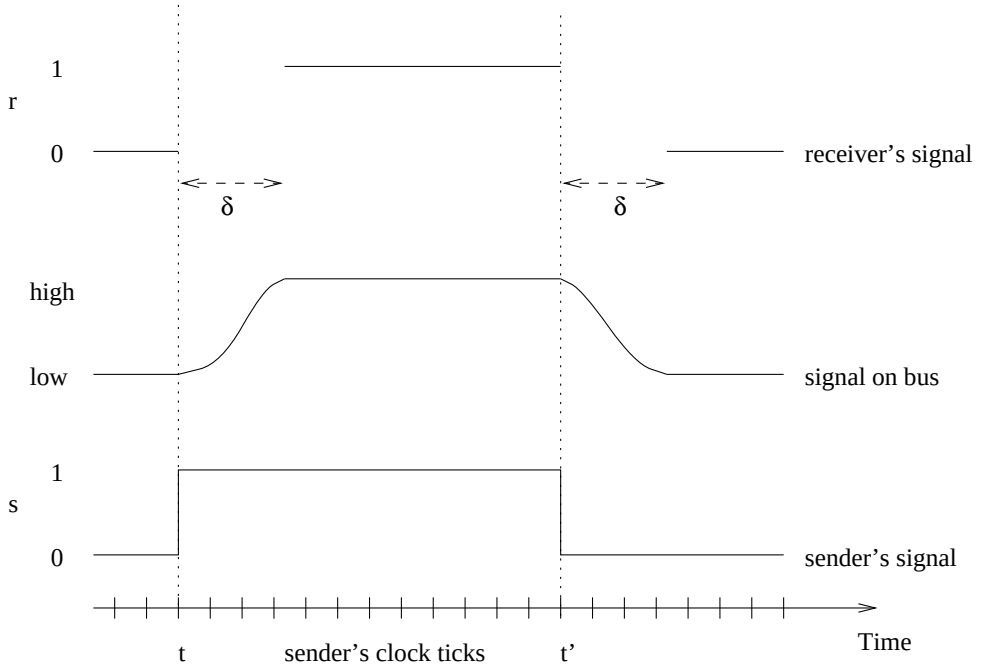


Figure 9: A model of communication at physical layer

In this section we have presented our discretization technique and useful definitions for formalising the stability of continuous states and the relationship between continuous states and discrete ones. In the next section we use those concepts for formulating the design process and reasoning about it.

4 A Technique for Design and Verification

In this section we present our method for designing real-time systems in DC. The design process is formulated in Section 4.1. In Section 4.2 we give some rules useful for the refinement and the verification. In Section 4.3 we prove that the design proposed in Section 1 for the Gas burner is correct w.r.t. the specification of the system.

4.1 Design Process

In this section we formulate the design process. The subtle step in the design process is the passage from the continuous specification to a discrete design.

We consider a discrete state variable w.r.t. a local clock as just a special case of continuous state variables whose discontinuity points coincide with ticks of the clock. So we can reason about the specification and the design in a uniform framework. For each controlled state variable r we consider a sampling state variable r' used by the control program to observe r at discrete time points. We make some assumptions about the behaviour of the sensors establishing the relationship between r and r'. The general form of the relationship between a state r and its sampling r' is $r' \overrightarrow{\approx}_\delta r$ for some non-negative real number δ. That is r' is an approximation (in the sense of Definition 8) of r in discrete time.

We also formalise the behaviour of actuators. We consider for each state t of the plant a program command t'' for requesting the plant getting into state t. We assume that when a command is issued its effect may take some delay to change the state of the plant. The assumptions about the behaviour of actuators are formalised using the concept of control introduced in Section 3. The general form of those assumptions is $t'' \rhd_\delta t$ or $t'' \trianglerighteq_\delta t$ for some non-negative real number δ.

The assumptions about the behaviour of the sensors and the actuators define the relationship between the continuous state variables and the discrete state variables relatively to the control program. As mentioned in the introduction, we consider as model of the detailed design the class of simple DC^* formulae over 'discrete' state variables. Moreover we can reason about the relationship between the stability, the speed and the controllability depending on individual case. For example, given some assumptions about the stability of states we can determine how fast the sensors, the controller and the actuators should be to satisfy the real-time requirement of the system.

4.2 Refinement and Verification Rules

In this section, we give some rules useful for reasoning about the design and its correctness. The following rules can be used to refine a specification toward and implementation as SDC^* formula and verify the correctness of the design. In those rules r, s, t and u are state variables, α, β, φ, ψ, and χ are formulae and δ and τ are non-negative real numbers. In the following proofs PC stands for Predicate Calculus and *Def.* for Definition.

Monotonic rules

Rule 1 If $r \overrightarrow{\approx}_\delta s$ then $\neg r \overrightarrow{\approx}_\delta \neg s$

Rule 2 If $(r \rhd_\tau s)$ and $(\tau \leq \delta)$ then $r \rhd_\delta s$

Rule 3 If $(r \trianglerighteq_\tau s)$ and $(\tau \leq \delta)$ then $r \trianglerighteq_\delta s$

Rule 4 If $(r \overrightarrow{\approx}_\tau s)$ and $(\tau \leq \delta)$ then $r \overrightarrow{\approx}_\delta s$

Rule 5 If $r \Rightarrow s$ then $r \rhd_0 s$

Rule 6 If $r \Rightarrow s$ then $r \trianglerighteq_0 s$

Rule 1 says that if a state r δ-observes a state s then $\neg r$ δ-observes $\neg s$. Its proof is straightforward from Definition 8. Rule 2 asserts that if a state r τ-controls a state s then r δ-controls s for any non-negative real number δ greater than or equal to τ. Therefore the smallest non-negative real number τ such that $r \triangleright_\tau s$ represents the maximum delay for an occurrence of r longer enough to start s. The meaning of rules 3 and 4 are defined in the similar way. Following is the proof of Rule 2. Rule 3 can be proved in the similar way. The proof of Rule 4 is straightforward from Definition 8 and Rule 2.

Proof (Rule 2).

1	$r \triangleright_\tau s$	assumption
2	$\delta \geq \tau$	assumption
3	$\Box(\llbracket r \rrbracket \wedge \ell > \tau \Rightarrow (\ell \leq \tau)^\frown \llbracket s \rrbracket)$	by 1, Def. 6
4	$\ell \leq \tau \Rightarrow \ell \leq \delta$	by 2, DC^*
5	$\ell > \delta \Rightarrow \ell > \tau$	by 2, DC^*
6	$(\ell \leq \tau)^\frown \llbracket s \rrbracket \Rightarrow (\ell \leq \delta)^\frown \llbracket s \rrbracket$	by 4, $Mono_l$
7	$\Box(\llbracket r \rrbracket \wedge \ell > \delta \Rightarrow (\ell \leq \tau)^\frown \llbracket s \rrbracket)$	by 3, 5, PC
8	$\Box(\llbracket r \rrbracket \wedge \ell > \delta \Rightarrow (\ell \leq \delta)^\frown \llbracket s \rrbracket)$	by 6, 7, PC
9	$r \triangleright_\delta s$	by 8, Def. 6

□

Rule 5 specifies the link between the implication over state variables and our concept of control of states. If r implies s then r 0-controls s. The meaning of Rule 6 is similar to that of Rule 5. The proof is easy and is not given here.

Converse monotonic rule

Rule 7 If δ-stable$(s) \wedge (\tau \leq \delta)$ then τ-stable(s)

Rule 7 asserts that if a state variable s is δ-stable then s is τ-stable for any positive real number τ less than or equal to δ. So the minimum distance between two consecutive occurrences of $\neg s$ is the greatest positive real number δ such that s is δ-stable. The proof is also obvious.

Transitivity rules

Rule 8 $$\frac{(r \triangleright_\delta s) \quad (s \triangleright_\tau t)}{r \triangleright_{(\delta+\tau)} t}$$

Rule 9 $$\frac{(r \trianglerighteq_\delta s) \quad (s \trianglerighteq_\tau t)}{r \trianglerighteq_{(\delta+\tau)} t}$$

Rule 10 $\dfrac{(r \overrightarrow{\approx}_{\delta} s) \quad (s \overrightarrow{\approx}_{\tau} t)}{r \overrightarrow{\approx}_{(\delta+\tau)} t}$

These rules say that the accuracy is deteriorated through sequential samplings of a state. They are helpful for the design, namely of distributed systems comprising many sensors, as how to use the sensors efficiently. We only give the proof of Rule 8. Rule 9 can be proved in the similar way. The proof of Rule 10 is straightforward from Definition 8 and Rule 8.

Proof (Rule 8).

1	$r \rhd_{\delta} s$	assumption
2	$s \rhd_{\tau} t$	assumption
3	$\Box(\llbracket r \rrbracket \wedge \ell > \delta \Rightarrow (\ell \leq \delta)^\frown \llbracket s \rrbracket)$	by 1, Def. 6
4	$\Box(\llbracket s \rrbracket \wedge \ell > \tau \Rightarrow (\ell \leq \tau)^\frown \llbracket t \rrbracket)$	by 2, Def. 6
5	$\Box(\llbracket r \rrbracket \wedge \ell > \delta + \tau \Rightarrow (\ell \leq \delta)^\frown (\llbracket s \rrbracket \wedge \ell > \tau))$	by 3, DC^*
6	$\Box(\llbracket r \rrbracket \wedge \ell > \delta + \tau \Rightarrow (\ell \leq \delta)^\frown (\ell \leq \tau)^\frown \llbracket t \rrbracket)$	by 4, 5, DC^*
7	$\Box(\llbracket r \rrbracket \wedge \ell > \delta + \tau \Rightarrow (\ell \leq \delta + \tau)^\frown \llbracket t \rrbracket)$	by 6, DC^*
8	$r \rhd_{(\delta+\tau)} t$	by 7, Def. 6

□

Control rules

Rule 11 $\dfrac{(r \rhd_{\delta} s) \quad (t \rhd_{\delta} u)}{(r \wedge t) \rhd_{\delta} (s \wedge u)}$

Rule 12 $\dfrac{(r \rhd_{\delta} t) \quad \Box(\ell \leq \delta - \tau^\frown \llbracket s \rrbracket \Rightarrow \llbracket r \rrbracket)}{s \rhd_{\tau} t}$

Rule 11 specifies the parallel composition of control properties. If r δ-controls s and t δ-controls u then $r \wedge t$ δ-controls $s \wedge u$. This rule can be proved as follows.

Proof (Rule 11).

1	$r \rhd_{\delta} s$	assumption
2	$t \rhd_{\delta} u$	assumption
3	$(r \wedge t) \rhd_0 r$	Rule 5, PC
4	$(r \wedge t) \rhd_0 t$	Rule 5, PC
5	$(r \wedge t) \rhd_{\delta} s$	by 1, 3, Rule 8
6	$(r \wedge t) \rhd_{\delta} u$	by 2, 4, Rule 8
7	$\Box(\llbracket r \wedge t \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \llbracket s \rrbracket)$	by 5, Def. 6
8	$\Box(\llbracket r \wedge t \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \llbracket u \rrbracket)$	by 6, Def. 6
9	$\Box(\llbracket r \wedge t \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown (\llbracket s \rrbracket \wedge \llbracket u \rrbracket))$	by 7, 8, PC, DC^*
10	$\Box(\llbracket r \wedge t \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \llbracket s \wedge u \rrbracket))$	by 9, DC^*
11	$(r \wedge t) \rhd_{\delta} (s \wedge u)$	by 10, Def. 6

□

Rule 12 says that a predictable state is suitable to be controlled. Indeed an occurrence of s is always predicted by an occurrence of r a while before. This gives a way to control s by observing r and taking the appropriate decision before s occurs. The proof of this rule is given below.

Proof (Rule 12).

1	$r \rhd_\delta t$	assumption
2	$\Box(\ell \leq \delta - \tau \frown [\![s]\!] \Rightarrow [\![r]\!])$	assumption
3	$\Box([\![r]\!] \Rightarrow ((\ell \leq \delta) \vee \ell \leq \delta \frown [\![t]\!]))$	by 1, Def. 6
4	$\Box((\ell \leq \delta - \tau) \frown [\![s]\!] \Rightarrow ((\ell \leq \delta) \vee \ell \leq \delta \frown [\![t]\!]))$	by 2, 3, DC^*
5	$\Box([\![s]\!] \Rightarrow ((\ell \leq \tau) \vee \ell \leq \tau \frown [\![t]\!]))$	by 4, DC^*
6	$s \rhd_\tau t$	by 5, Def. 6

□

Observation rules

Rule 13 $$\frac{r \rhd_\delta s}{\Box([\![r]\!] \wedge [\![\neg s]\!] \Rightarrow \ell \leq \delta)}$$

Rule 14 $$\frac{r \overrightarrow{\approx}_\delta s \quad \delta - stable(\neg s)}{\Box(([\![r]\!] \wedge \ell > \delta) \Rightarrow \ell \leq \delta \frown [\![s]\!] \frown \ell \leq \delta)}$$

Rule 15 $$\frac{\tau - stable(\neg s) \quad \delta \geq \tau \quad b > \tau}{\begin{array}{l}[\![s]\!] \frown \ell \leq \delta \frown ([\![\neg s]\!] \wedge \ell \geq b) \Rightarrow \\ [\![s]\!] \frown ([\![\neg s]\!] \wedge \ell = \tau) \frown \ell \leq \delta - \tau \frown ([\![\neg s]\!] \wedge \ell \geq b - \tau)\end{array}}$$

Rule 16 $$\frac{r \overrightarrow{\approx}_\delta s \quad \frac{\delta}{2} - stable(r) \quad \frac{\delta}{2} - stable(\neg r)}{\Box(([\![s]\!] \wedge \ell > \delta) \frown ([\![\neg s]\!] \wedge \ell > \delta) \Rightarrow \ell \leq \delta \frown [\![r]\!] \frown [\![\neg r]\!])}$$

The rule 13 says that if a state r δ-observes a state s then r approximates s with a delay less than or equal to δ time units. This rule is proved as follows.

Proof (Rule 13).

1	$r \rhd_\delta s$	assumption
2	$\Box([\![r]\!] \wedge \ell > \delta \Rightarrow \ell \leq \delta \frown [\![s]\!])$	by 1, Def. 6
3	$\Box([\![r]\!] \Rightarrow (\ell \leq \delta \vee (\ell \leq \delta \frown [\![s]\!])))$	by 2, PC
4	$\Box([\![r]\!] \wedge [\![\neg s]\!] \Rightarrow ((\ell \leq \delta \vee (\ell \leq \delta \frown [\![s]\!])) \wedge [\![\neg s]\!]))$	by 3, PC
5	$\Box([\![r]\!] \wedge [\![\neg s]\!] \Rightarrow (\ell \leq \delta \wedge [\![\neg s]\!]) \vee ((\ell \leq \delta \frown [\![s]\!]) \wedge [\![\neg s]\!]))$	by 4, PC
6	$\Box([\![r]\!] \wedge [\![\neg s]\!] \Rightarrow (\ell \leq \delta \wedge [\![\neg s]\!]))$	by 5, DC^*
7	$\Box([\![r]\!] \wedge [\![\neg s]\!] \Rightarrow \ell \leq \delta)$	by 6, PC

□

Rule 14 says that if r δ-observes s and $\neg s$ is δ-stable then any period longer than δ time units satisfying r satisfies $\ell \leq \delta^\frown[\![s]\!]^\frown\ell \leq \delta$. The rule is proved as follows.

Proof (Rule 14).

1	$\delta - stable(\neg s)$	assumption
2	$r \overrightarrow{\approx}_\delta s$	assumption
3	$\Box([\![\neg s]\!] \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown[\![\neg r]\!])$	by 2, Def. 8, Def. 6
4	$([\![r]\!] \wedge \ell > \delta) \Rightarrow ([\![s]\!] \vee [\![\neg s]\!])^*$	finite variability
5	$([\![r]\!] \wedge \ell > \delta) \Rightarrow ([\![]\!] \vee [\![s]\!] \vee [\![\neg s]\!])^\frown$ $([\![s]\!] \vee ([\![\neg s]\!] \wedge \ell > \delta))^*{}^\frown([\![]\!] \vee [\![s]\!] \vee [\![\neg s]\!])$	by 1, 4, DC^*
6	$([\![r]\!] \wedge \ell > \delta) \Rightarrow (([\![]\!] \vee [\![s]\!] \vee [\![\neg s]\!]) \wedge \ell \leq \delta)^\frown$ $[\![s]\!]^*{}^\frown(([\![]\!] \vee [\![s]\!] \vee [\![\neg s]\!]) \wedge \ell \leq \delta)$	by 3, 5, DC^*
7	$([\![r]\!] \wedge \ell > \delta) \Rightarrow (\ell \leq \delta^\frown[\![s]\!]^*{}^\frown\ell \leq \delta)^\frown\ell > \delta$	by 6, DC^*, PC
8	$([\![r]\!] \wedge \ell > \delta) \Rightarrow \ell \leq \delta^\frown[\![s]\!]^\frown\ell \leq \delta$	by 1, 2, 7, DC^*
9	$\Box([\![r]\!] \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown[\![s]\!]^\frown\ell \leq \delta)$	by 8, DC^*

□

Rule 15 allows to capture the change of state (edge) from 0 to 1 or from 1 to 0. Its proof is straightforward.

Proof (Rule 15).

1	$\tau - stable(\neg s)$	assumption
2	$(\delta \geq \tau) \wedge (b > \tau)$	assumption
3	$[\![s]\!]^\frown\ell \leq \delta^\frown([\![\neg s]\!] \wedge \ell \geq b)$	
	$\Rightarrow [\![s]\!]^\frown(([\![\neg s]\!] \vee [\![s]\!])^* \wedge \ell \leq \delta)^\frown([\![\neg s]\!] \wedge \ell \geq b)$	finite variability
	$\Rightarrow [\![s]\!]^\frown((([\![\neg s]\!] \wedge \ell > \tau) \vee [\![s]\!])^* \wedge \ell \leq \delta)^\frown$ $([\![\neg s]\!] \wedge \ell \geq b)$	by 1, DC^*
	$\Rightarrow [\![s]\!]^\frown([\![\neg s]\!] \wedge \ell = \tau)^\frown(([\![\neg s]\!] \vee [\![s]\!])^*$ $\wedge \ell \leq \delta - \tau)^\frown([\![\neg s]\!] \wedge \ell \geq b - \tau)$	by 2, DC^*
	$\Rightarrow [\![s]\!]^\frown([\![\neg s]\!] \wedge \ell = \tau)^\frown\ell \leq \delta - \tau^\frown$ $([\![\neg s]\!] \wedge \ell \geq b - \tau)$	DC^*

□

Rule 16 says that if r δ-observes s then any change in s is observable by r provided s and $\neg s$ are δ-stable. In other words, without assumption on the stability of continuous states, there is no way to observe them in discrete time. The rule is proved as follows.

Proof (Rule 16).

1	$r \overrightarrow{\approx}_\delta s$	assumption
2	$\frac{\delta}{2}-stable(r)$	assumption
3	$\frac{\delta}{2}-stable(\neg r)$	assumption
4	$\Box((\llbracket s \rrbracket \wedge \ell > \delta) \Rightarrow \ell \leq \delta^\frown \llbracket r \rrbracket)$	by 1, Def. 8
5	$\Box((\llbracket \neg s \rrbracket \wedge \ell > \delta) \Rightarrow \ell \leq \delta^\frown \llbracket \neg r \rrbracket)$	by 1, Def. 8
6	$\Box((\llbracket s \rrbracket \wedge \ell > \delta)^\frown(\llbracket \neg s \rrbracket \wedge \ell > \delta) \Rightarrow$ $\ell \leq \delta^\frown \llbracket r \rrbracket^\frown(\llbracket \neg s \rrbracket \wedge \ell > \delta))$	by 4, $Mono_l$
7	$\Box((\llbracket s \rrbracket \wedge \ell > \delta)^\frown(\llbracket \neg s \rrbracket \wedge \ell > \delta) \Rightarrow$ $\ell \leq \delta^\frown \llbracket r \rrbracket^\frown \ell \leq \delta^\frown \llbracket \neg r \rrbracket)$	by 6, 7, DC^*
8	$\Box((\llbracket s \rrbracket \wedge \ell > \delta)^\frown(\llbracket \neg s \rrbracket \wedge \ell > \delta) \Rightarrow \ell \leq \delta^\frown \llbracket r \rrbracket^\frown \llbracket \neg r \rrbracket)$	by 2, 3, 7, DC^*

□

State Distance

Rule 17 $$\frac{r \overrightarrow{\approx}_\delta s \quad 2\delta - stable(r)}{(\delta+\tau)-stable(s) \Rightarrow \tau - stable(r)}$$

Rule 17 is useful for refinement. In general we cannot guarantee the stability of the states of the environment, we only make an assumption about it. This rule defines a necessary condition between the stability of a continuous state and the stability of its sampling. For example, the design decision D_{gb} of the gas burner requires $Leak$ to be 30-stable. So it is necessary that for any implementation I_{gb} of the system the formula $I_{gb} \Rightarrow (30-\delta)$-stable$(\neg Leak')$ is satisfied.

The rule is proved as follows.

Proof (Rule 17).

1	$r \overrightarrow{\approx}_\delta s$	assumption
2	$2\delta - stable(r)$	assumption
3	$(\llbracket \neg s \rrbracket \wedge \ell \leq \delta)^\frown(\llbracket s \rrbracket \wedge \ell > \delta+\tau)^\frown(\llbracket \neg s \rrbracket \wedge \ell \leq \delta)$ $\Rightarrow (\llbracket \neg s \rrbracket \wedge \ell \leq \delta)^\frown \ell \leq \delta^\frown(\llbracket r \rrbracket \wedge \ell > \tau)^\frown(\llbracket \neg s \rrbracket \wedge \ell \leq \delta)$	by 1, DC^*
	$\Rightarrow (\llbracket \neg r \rrbracket \vee \llbracket r \rrbracket)^\frown(\ell = 0 \vee \llbracket \neg r \rrbracket)^\frown(\llbracket r \rrbracket \wedge \ell > \tau)^\frown$ $(\llbracket \neg r \rrbracket \vee \llbracket r \rrbracket)^\frown(\ell = 0 \vee \llbracket r \rrbracket)$	by 2, DC^*
4	$\Box(\llbracket \neg s \rrbracket^\frown \llbracket s \rrbracket^\frown \llbracket \neg s \rrbracket \Rightarrow \llbracket \neg s \rrbracket^\frown(\llbracket s \rrbracket \wedge \ell > \delta+\tau)^\frown \llbracket \neg s \rrbracket)$ $\Rightarrow \Box(\llbracket \neg r \rrbracket^\frown \llbracket r \rrbracket^\frown \llbracket \neg r \rrbracket \Rightarrow \llbracket \neg r \rrbracket^\frown(\llbracket r \rrbracket \wedge \ell > \tau)^\frown \llbracket \neg r \rrbracket)$	by 3, DC^*

□

State Occurrence

Rule 18 $s \rhd_\delta \neg s \Leftrightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta)$

Rule 19 $s \trianglerighteq_\delta \neg s \Leftrightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta)$

Rule 20 $$\frac{s \rhd_\delta t}{\Box(\llbracket t \rrbracket \Rightarrow \ell \leq \tau) \Rightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta + \tau)}$$

Rule 21 $$\frac{r \overrightarrow{\approx}_\delta s}{\Box(\llbracket r \rrbracket \Rightarrow \ell \leq \tau) \Rightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta + \tau)}$$

The rules 18 and 19 are useful for verifying that a state does not last more than some amount of time. Their proofs are also similar.

Proof (Rule 18).

$s \rhd_\delta \neg s$	$\Leftrightarrow \Box(\llbracket s \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \llbracket \neg s \rrbracket)$	Def. 6
	$\Leftrightarrow \Box(\llbracket s \rrbracket \Rightarrow (\ell \leq \delta \vee (\ell \leq \delta^\frown \llbracket \neg s \rrbracket)))$	DC^*
	$\Leftrightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta)$	DC^*

□

The rules 20 and 21 are helpful for both refinement and verification. For example, if we can prove that $I_{gb} \Rightarrow \Box(\llbracket Leak' \rrbracket \Rightarrow \ell \leq 1 - \delta)$ then we can deduce by these rules that $I_{gb} \Rightarrow \Box(\llbracket Leak \rrbracket \Rightarrow \ell \leq 1)$. These rules also define how fast the control program should be to satisfy a time constraint about the occurrence of state. We only give the proof of Rule 20. The proof of Rule 21 is straightforward from Definition 8 and Rule 20.

Proof (Rule 20).

1	$s \rhd_\delta t$	assumption
2	$\Box(\llbracket s \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown \llbracket t \rrbracket)$	by 1, Def. 6
3	$\Box(\llbracket t \rrbracket \Rightarrow \ell \leq \tau) \Rightarrow \Box(\llbracket s \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown(\llbracket t \rrbracket \wedge \ell \leq \tau))$	by 2, DC^*
4	$\Box(\llbracket s \rrbracket \wedge \ell > \delta \Rightarrow \ell \leq \delta^\frown(\llbracket t \rrbracket \wedge \ell \leq \tau)) \Rightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta + \tau)$	DC^*
5	$\Box(\llbracket t \rrbracket \Rightarrow \ell \leq \tau) \Rightarrow \Box(\llbracket s \rrbracket \Rightarrow \ell \leq \delta + \tau)$	by 3, 4, PC

□

The following Rules are useful for proving safety properties.

Invariant Rule for loop

Rule 23 $$\frac{\begin{array}{lll} \varphi \Rightarrow \neg(\neg\beta^\frown true) & \ell = 0 \Rightarrow \alpha & \varphi \Rightarrow \Box\chi \\ \varphi \Rightarrow \neg(true^\frown\neg\alpha) & \ell = 0 \Rightarrow \beta & \alpha^\frown\varphi^*{}^\frown\beta \Rightarrow \chi \end{array}}{\varphi^* \Rightarrow \Box\chi}$$

Proof (Rule 23).

1	$\varphi \Rightarrow \neg(true^\frown\neg\alpha)$	assumption
2	$\varphi \Rightarrow \neg(true^\frown\neg(\alpha^\frown\ell = 0))$	by 1, DC^*
3	$\ell = 0 \Rightarrow \varphi^*$	by DC_1^*
4	$\varphi \Rightarrow \neg(true^\frown\neg(\alpha^\frown\varphi^*))$	by 2, 3, $Mono_r$
5	$\ell = 0 \Rightarrow \alpha$	assumption
6	$\ell = 0 \Rightarrow (\ell = 0^\frown\ell = 0)$	$L2$
7	$\ell = 0 \Rightarrow (\alpha^\frown\varphi^*))$	by 5, 6, DC_1^*, $Mono_l$, $Mono_r$
8	$(true^\frown\neg(\alpha^\frown\varphi^*)) \Rightarrow \neg\ell = 0$	by 7, DC^*
9	$\neg((true^\frown\neg(\alpha^\frown\varphi^*)) \wedge \ell = 0^\frown true)$	by 8, N_l
10	$(\varphi^* \wedge (true^\frown\neg(\alpha^\frown\varphi^*))) \Rightarrow$ $((true^\frown\neg(\alpha^\frown\varphi^*)) \wedge \ell = 0^\frown true)\vee$ $((\varphi^* \wedge \neg(true^\frown\neg(\alpha^\frown\varphi^*))^\frown\varphi)\wedge$ $(true^\frown\neg(\alpha^\frown\varphi^*))^\frown true)$	by DC_3^*
11	$(\varphi^* \wedge \neg(true^\frown\neg(\alpha^\frown\varphi^*))^\frown\varphi)\wedge$ $(true^\frown\neg(\alpha^\frown\varphi^*)) \Rightarrow$ $(true^\frown(\alpha^\frown\varphi^*{}^\frown\varphi) \wedge \neg(\alpha^\frown\varphi^*))\vee$ $(\varphi \wedge (true^\frown\neg(\alpha^\frown\varphi^*))))$	DC^*
12	$(\alpha^\frown\varphi^*{}^\frown\varphi) \Rightarrow (\alpha^\frown\varphi^*)$	by DC_2^*, $Mono_r$
13	$\neg((\alpha^\frown\varphi^*{}^\frown\varphi) \wedge \neg(\alpha^\frown\varphi^*))$ $\vee(\varphi \wedge (true^\frown\neg(\alpha^\frown\varphi^*)))$	by 4, $Mono_r$, 12
14	$\neg(true^\frown((\alpha^\frown\varphi^*{}^\frown\varphi) \wedge \neg(\alpha^\frown\varphi^*))$ $\vee(\varphi \wedge (true^\frown\neg(\alpha^\frown\varphi^*))))$	by 13, N_r
15	$\neg((\varphi^* \wedge \neg(true^\frown\neg(\alpha^\frown\varphi^*))^\frown\varphi)\wedge$ $(true^\frown\neg(\alpha^\frown\varphi^*)))$	by 11, 14
16	$\varphi^* \Rightarrow \neg(true^\frown\neg(\alpha^\frown\varphi^*)))$	by 9, 10, 15, $Mono_l$
17	$(\alpha^\frown\varphi^*) \wedge (\neg(\alpha^\frown\varphi^*{}^\frown\beta)^\frown true) \Rightarrow$ $(\alpha \wedge (\neg(\alpha^\frown\varphi^*{}^\frown\beta)^\frown true)^\frown true)\vee$ $(\alpha^\frown(\varphi^* \wedge (\neg(\varphi^*{}^\frown\beta)^\frown true)))$	DC^*
18	$\ell = 0 \Rightarrow \beta$	assumption
19	$\ell = 0 \Rightarrow ((\varphi^*{}^\frown\beta)^\frown true)$	by 6, DC_1^*, 18, $Mono_l$, $Mono_r$
20	$\alpha \Rightarrow (\alpha^\frown\ell = 0)$	by $L3_r$
21	$\alpha \Rightarrow ((\alpha^\frown\varphi^*{}^\frown\beta)^\frown true)$	by 19, 20, $Mono_r$
22	$(\alpha^\frown\varphi^*) \wedge (\neg(\alpha^\frown\varphi^*{}^\frown\beta)^\frown true) \Rightarrow$ $(\alpha^\frown(\varphi^* \wedge (\neg(\varphi^*{}^\frown\beta)^\frown true)))$	by 17, 21, $Mono_l$
23	$\varphi^* \wedge (\neg(\varphi^*{}^\frown\beta)^\frown true) \Rightarrow$ $(\varphi^* \wedge (\neg(\varphi^*{}^\frown\beta)^\frown true)^\frown true)$	by $Mono_r$
24	$\varphi^* \wedge (\neg(\varphi^*{}^\frown\beta)^\frown true) \Rightarrow$ $(\ell = 0 \wedge (\neg(\varphi^*{}^\frown\beta)^\frown true)^\frown true)\vee$ $(((\varphi^* \wedge \neg(\neg(\varphi^*{}^\frown\beta)^\frown true))^\frown\varphi)\wedge$ $(\neg(\varphi^*{}^\frown\beta)^\frown true)^\frown true)$	by DC_3^*

25	$\ell = 0 \Rightarrow (\varphi^* \frown \beta)$	by 6, DC_1^*, 18, $Mono_l$, $Mono_r$
26	$(\neg(\varphi^* \frown \beta) \frown true) \Rightarrow \ell \neq 0$	by 25, $Mono_l$
27	$((\varphi^* \wedge \neg(\neg(\varphi^* \frown \beta) \frown true)) \frown \varphi)$ $\wedge(\neg(\varphi^* \frown \beta) \frown true) \Rightarrow$ $(\varphi^* \frown \varphi \wedge (\neg\beta \frown true))$	DC^*
28	$\varphi \Rightarrow \neg(\neg\beta \frown true)$	assumption
29	$\neg(\varphi^* \frown \varphi \wedge (\neg\beta \frown true))$	by 28, N_r
30	$\neg(((\varphi^* \wedge \neg(\neg(\varphi^* \frown \beta) \frown true)) \frown \varphi) \wedge$ $(\neg(\varphi^* \frown \beta) \frown true) \frown true)$	by 29, N_l
31	$\varphi^* \Rightarrow \neg(\neg(\varphi^* \frown \beta) \frown true)$	by 23, 24, 26, 30
32	$\neg(\alpha \frown (\varphi^* \wedge (\neg(\varphi^* \frown \beta) \frown true)))$	by 31, N_r
33	$(\alpha \frown \varphi^*) \Rightarrow \neg(\neg(\alpha \frown (\varphi^* \frown \beta)) \frown true)$	by 22, 32
34	$\varphi^* \Rightarrow \neg((true \frown \neg(\alpha \frown (\varphi^* \frown \beta))) \frown true)$	by 16, 33, $Mono_l$, $Mono_r$
35	$\alpha \frown \varphi^* \frown \beta \Rightarrow \chi$	assumption
36	$\Box(\alpha \frown \varphi^* \frown \beta) \Rightarrow \Box\chi$	by 35, DC^*
37	$\varphi^* \Rightarrow \Box\chi$	by 34, 36, PC

□

Invariant Rule for sequential concatenation

Rule 24
$$\frac{\varphi \Rightarrow \Box\chi \quad \varphi \Rightarrow \neg(true \frown \neg\alpha) \quad \psi \Rightarrow \Box\chi \quad \psi \Rightarrow \neg(\neg\beta \frown true) \quad \alpha \frown \beta \Rightarrow \chi}{\varphi \frown \psi \Rightarrow \Box\chi}$$

Proof (Rule 24).

1	$\varphi \Rightarrow \Box\chi$	assumption
2	$\psi \Rightarrow \Box\chi$	assumption
3	$\varphi \Rightarrow \neg(true \frown \neg\alpha)$	assumption
4	$\psi \Rightarrow \neg(\neg\beta \frown true)$	assumption
5	$\alpha \frown \beta \Rightarrow \chi$	assumption
6	$\varphi \frown \psi \Rightarrow \Box\chi \frown \Box\chi$	by 1, 2, DC^*
7	$\varphi \frown \psi \Rightarrow \neg(true \frown \neg\alpha) \frown \neg(\neg\beta \frown true)$	by 3, 4, DC^*
8	$\neg(true \frown \neg\alpha) \frown \neg(\neg\beta \frown true) \Rightarrow true \frown \alpha \frown \beta \frown true$	DC^*
9	$\varphi \frown \psi \Rightarrow true \frown \alpha \frown \beta \frown true$	by 7, 8, PC
10	$\varphi \frown \psi \Rightarrow \Box\chi$	by 3, 4, 6, 9, DC^*

□

Trivial parallel composition

Rule 25
$$\frac{A \Rightarrow \Box\psi \quad B \Rightarrow \Box\varphi}{A \wedge B \Rightarrow \Box(\psi \wedge \varphi)}$$

Proof (Rule 25).

1	$A \Rightarrow \Box\psi$	assumption
2	$B \Rightarrow \Box\varphi$	assumption
3	$A \wedge B \Rightarrow \Box\psi \wedge \Box\varphi$	by 1, 2, PC
4	$A \wedge B \Rightarrow \Box(\psi \wedge \varphi)$	by 3, DC^*

□

Synchronisation

Rule 26 $((\varphi \wedge [\![\neg r]\!]^\frown[\![r]\!]) \wedge ((\psi \wedge [\![\neg r]\!]^\frown[\![r]\!]) \Rightarrow (\varphi \wedge \psi \wedge [\![\neg r]\!]^\frown[\![r]\!]$

Rule 27 $([\![r]\!]^\frown(\varphi \wedge [\![\neg r]\!])) \wedge ([\![r]\!]^\frown(\psi \wedge [\![\neg r]\!])) \Rightarrow [\![r]\!]^\frown(\varphi \wedge \psi \wedge [\![\neg r]\!])$

The proofs of these rules are straightforward from the proof system of DC^*.

Duration of state

Rule 28 $(\ell = 0 \vee [\![r]\!] \vee [\![\neg r]\!])^\frown((\ell = 0 \vee [\![\neg r]\!])^\frown([\![r]\!] \wedge \ell > \delta)^\frown(\ell = 0 \vee [\![\neg r]\!]))^*{}^\frown(\ell = 0 \vee [\![r]\!] \vee [\![\neg r]\!]) \Rightarrow \delta\text{-stable}(r)$

Rule 29 $(\ell = 0 \vee [\![\neg r]\!])^\frown(([\![r]\!] \wedge \ell \leq \delta)^\frown[\![\neg r]\!])^*{}^\frown(\ell = 0 \vee ([\![r]\!] \wedge \ell \leq \delta)) \Rightarrow \Box([\![r]\!] \Rightarrow \ell \leq \delta)$

In other words, these rules can be formulated as:

$$PREF([\![r]\!]^* {}^\frown([\![\neg r]\!]^\frown([\![r]\!] \wedge \ell > \delta))^* {}^\frown[\![\neg r]\!]) \Rightarrow \delta - \text{stable}(r)$$

$$PREF([\![\neg r]\!]^* {}^\frown([\![r]\!] \wedge \ell \leq \delta)^\frown[\![\neg r]\!])^* {}^\frown ([\![\neg r]\!] \wedge \ell \leq \delta)) \Rightarrow \Box([\![r]\!] \Rightarrow \ell \leq \delta)$$

Rule 28 is useful for verifying that a SDC^* formula satisfies a time constraint about the distance of state. For example, we can use this rule to prove that $I_{gb} \Rightarrow (30 - \delta)-\text{stable}(\neg Leak')$. For simplifying the proof, we consider the following abbreviations.

$$\alpha \mathrel{\widehat{=}} \ell = 0 \vee [\![\neg r]\!] \vee [\![r]\!]$$

$$\varphi \mathrel{\widehat{=}} (\ell = 0 \vee [\![\neg r]\!])^\frown([\![r]\!] \wedge \ell > \delta)^\frown(\ell = 0 \vee [\![\neg r]\!])$$

$$\chi \mathrel{\widehat{=}} [\![\neg r]\!]^\frown[\![r]\!]^\frown[\![\neg r]\!] \Rightarrow \ell > \delta$$

The proof can be summarised as follows. We first prove that $\varphi^* \Rightarrow \Box\chi$ using Rule 23. Then we prove that $\alpha^\frown\varphi^*{}^\frown\alpha \Rightarrow \Box\chi$ using Rule 24 twice.

Proof (Rule 28).

1	$\alpha \Rightarrow \chi$	DC^*
2	$\varphi \Rightarrow \chi$	DC^*
3	$\varphi^* \wedge \chi \Rightarrow (\varphi^* \wedge \chi)^\frown true$	DC^*
4	$(\varphi^* \wedge \chi)^\frown true \Rightarrow ((\ell = 0 \wedge \chi)^\frown true) \vee$ $((((\varphi^* \wedge \chi)^\frown\varphi) \wedge \chi)^\frown true)$	by DC_3^*
5	$\ell = 0 \Rightarrow \chi$	DC^*
6	$(\varphi^* \wedge \chi)^\frown true \Rightarrow ((((\varphi^* \wedge \chi)^\frown\varphi) \wedge \chi)^\frown true)$	by 4, 5, $Mono_r$
7	$(\varphi^* \wedge \chi)^\frown\varphi \Rightarrow \chi$	by 2, 3, DC^*
8	$\varphi^* \Rightarrow \chi$	by 6, 7, $Mono_r$
9	$\alpha^\frown\varphi^* \Rightarrow \chi$	by 1, 8, DC^*
10	$(\alpha^\frown\varphi^*)^\frown\beta \Rightarrow \chi$	by 1, 9, DC^*
11	$\ell = 0 \Rightarrow \alpha$	DC^*
12	$\varphi \Rightarrow \neg(true^\frown\neg\alpha)$	DC^*
13	$\varphi \Rightarrow \neg(\neg\alpha^\frown true)$	DC^*
14	$\varphi^* \Rightarrow \Box\chi$	by 11, 12, 13, 14, Rule 23
15	$\alpha \Rightarrow \Box\chi$	DC^*
16	$\alpha \Rightarrow \neg(true^\frown\neg\alpha)$	DC^*
17	$\alpha^\frown\alpha \Rightarrow \chi$	DC^*
18	$\alpha^\frown\varphi^* \Rightarrow \Box\chi$	by 13, 14, 15, 16, 18, Rule 24
19	$\alpha^\frown\varphi^* \Rightarrow \neg(true^\frown\neg\alpha)$	DC^*
20	$\alpha \Rightarrow \neg(\neg\alpha^\frown true)$	DC^*
21	$(\alpha^\frown\varphi^*)^\frown\alpha \Rightarrow \Box\chi$	by 15, 17, 18, 19, 20, Rule 24

□

Rule 29 is useful for verifying that a SDC^* formula satisfies a time constraint about the occurrence of state. For example, we can use this rule to prove that $I_{gb} \Rightarrow \Box([\![Leak']\!] \Rightarrow \ell \leq 1 - \delta)$.

We now give the proof of Rule 29. For sake of simplicity we consider the following abbreviations.

$$\alpha \mathrel{\widehat{=}} \ell = 0 \vee [\![\neg r]\!]$$

$$\beta \mathrel{\widehat{=}} \ell = 0 \vee ([\![r]\!] \wedge \ell \leq \delta)$$

$$\varphi \mathrel{\widehat{=}} ([\![r]\!] \wedge \ell \leq \delta)^\frown[\![\neg r]\!]$$

$$\chi \mathrel{\widehat{=}} [\![r]\!] \Rightarrow \ell \leq \delta$$

We first prove that $\varphi^* \Rightarrow \Box\chi$ using Rule 23. Then we prove that $\alpha^\frown\varphi^*{}^\frown\beta \Rightarrow \Box\chi$ using Rule 24 twice.

Proof (Rule 29).

1	$\alpha \Rightarrow \chi$	DC^*
2	$\beta \Rightarrow \chi$	DC^*
3	$\varphi \Rightarrow \chi$	DC^*
4	$\varphi^* \wedge \chi \Rightarrow (\varphi^* \wedge \chi)^\frown true$	DC^*
5	$(\varphi^* \wedge \chi)^\frown true \Rightarrow ((\ell = 0 \wedge \chi)^\frown true) \vee$ $((((\varphi^* \wedge \chi)^\frown\varphi) \wedge \chi)^\frown true)$	by DC^*_3
6	$\ell = 0 \Rightarrow \chi$	DC^*
7	$(\varphi^* \wedge \chi)^\frown true \Rightarrow ((((\varphi^* \wedge \chi)^\frown\varphi) \wedge \chi)^\frown true)$	by 5, 6, $Mono_r$
8	$(\varphi^* \wedge \chi)^\frown\varphi \Rightarrow \chi$	by 3, 4, DC^*
9	$\varphi^* \Rightarrow \chi$	by 7, 8, $Mono_r$
10	$\alpha^\frown\varphi^* \Rightarrow \chi$	by 1, 9, DC^*
11	$(\alpha^\frown\varphi^*)^\frown\beta \Rightarrow \chi$	by 2, 10, DC^*
12	$\ell = 0 \Rightarrow \alpha \wedge \beta$	DC^*
13	$\varphi \Rightarrow \neg(true^\frown\neg\alpha)$	DC^*
14	$\varphi \Rightarrow \neg(\neg\beta^\frown true)$	DC^*
15	$\varphi^* \Rightarrow \Box\chi$	by 11, 12, 13, 14, Rule 23
16	$\alpha \Rightarrow \Box\chi$	DC^*
17	$\alpha \Rightarrow \neg(true^\frown\neg\alpha)$	DC^*
18	$\alpha^\frown\beta \Rightarrow \chi$	DC^*
19	$\alpha^\frown\varphi^* \Rightarrow \Box\chi$	by 14, 15, 16, 17, 18, Rule 24
20	$\alpha^\frown\varphi^* \Rightarrow \neg(true^\frown\neg\alpha)$	DC^*
21	$\beta \Rightarrow \neg(\neg\beta^\frown true)$	DC^*
22	$\beta \Rightarrow \Box\chi$	DC^*
23	$(\alpha^\frown\varphi^*)^\frown\beta \Rightarrow \Box\chi$	by 18, 19, 20, 21, 22, Rule 24

□

4.3 An example

We consider our example in the introduction, Gas burner, and formalise the assumptions about the behaviour of the sensor and the stability of $Leak$ as

$$\begin{aligned} A_{gb} \;\widehat{=}\; & (Leak' \overset{\rightarrow}{\approx}_\delta Leak) \wedge 2\delta\text{-stable}(Leak) \wedge \\ & \delta < 1/2 \wedge recover \rhd_{(1-2\delta)} \neg Leak \wedge \\ & \delta\text{-stable}(Leak') \wedge \delta\text{-stable}(\neg Leak') \end{aligned}$$

where $Leak'$ is the sampling of $Leak$ via the sensor. So it is reasonable that $Leak'$ should stable for one sampling step. Here we prove that the implementation I_{gb} satisfies the design decision D_{gb} under assumption A_{gb}, provided $\delta < 1$.

Theorem 3 $(\delta < 1) \wedge A_{gb} \vdash I_{gb} \Rightarrow D_{gb}$

Proof ().

1	$\delta < 1$	{assumption}
2	A_{gb}	{assumption}
3	$I_{gb} \Rightarrow$ $PREF(\llbracket \neg leak' \rrbracket^\frown$ $((\llbracket leak' \rrbracket \wedge 1 - 2\delta \leq \ell \leq 1 - \delta) \wedge$ $(\ell \leq 1 - 2\delta^\frown \llbracket \neg Leak \rrbracket))^\frown$ $(\llbracket \neg leak \rrbracket \wedge \ell = 31 + \delta)^\frown$ $(\llbracket \neg leak' \rrbracket \vee \ell = 0)^*)$	{DC*}
4	$I_{gb} \Rightarrow$ $PREF(\llbracket \neg leak' \rrbracket^\frown$ $(\llbracket leak' \rrbracket \wedge 1 - 2\delta \leq \ell \leq 1 - \delta)^\frown$ $(\llbracket \neg leak' \rrbracket \wedge \ell \geq 30 + \delta)^\frown$ $(\llbracket \neg leak' \rrbracket \vee \ell = 0)^*)$	{DC*, A_{gb}}
5	$I_{gb} \Rightarrow \Box(\llbracket Leak' \rrbracket \Rightarrow \ell \leq 1 - \delta)$	{by 4,Rule 29, DC^*, PC}
6	$\Box(\llbracket Leak' \rrbracket \Rightarrow \ell \leq 1 - \delta) \Rightarrow \Box(\llbracket Leak \rrbracket \Rightarrow \ell \leq 1)$	{by 1, 2, Rule 21}
7	$I_{gb} \Rightarrow \Box(\llbracket Leak \rrbracket \Rightarrow \ell \leq 1)$	{by 5, 6, PC}
8	$I_{gb} \Rightarrow D_{gb}$	{by 2, 7, PC, arithmetic}

□

5 Example Water Level Controller

We consider a water level controller that opens and closes a valve regulating the outflow of water from a container. The container has an input vent through which water flows at the constant rate λ. When the valve is open, the water level decreases at rate λ. The controller is supposed to keep the water level in between the critical low level ω_{cl} and high level ω_{ch} units. Let ω denoting the water level at any time. We suppose that the valve is either open or closed. We model the state of the valve by a state variable *Open* which is true when the valve is open and false when it is closed.

We consider the following state variables $s \mathrel{\widehat{=}} \omega \geq \omega_{cl}$ which holds when the water level is above the critical low level, and $r \mathrel{\widehat{=}} \omega \leq \omega_{ch}$ which holds when the water level is below the critical high level. Initially the water level is ω_0. Then the requirement of the water level monitoring system is formalised as

$$S_{wl} \mathrel{\widehat{=}} \Box(\llbracket s \rrbracket \wedge \llbracket r \rrbracket).$$

We use sensors to observe the water level in the tank and actuators to open or close the valve. Suppose it takes δ time units to observe the water level through the sensors. Suppose it takes

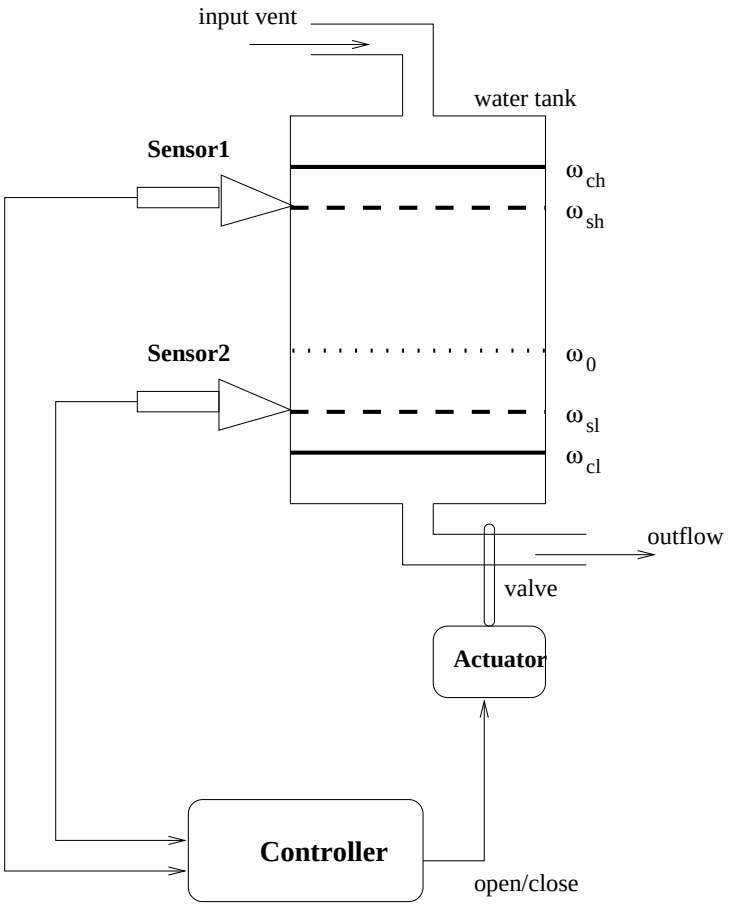


Figure 10: Water tank monitoring system

the actuators τ time units to open or close the valve whenever it receives the command from the controller. These assumption about the behaviours of the sensors and the actuators are formalised in our approach as follows.

$$A_{wl} \mathrel{\widehat{=}} (s' \overrightarrow{\approx}_{\delta} s) \wedge (r' \overrightarrow{\approx}_{\delta} r) \wedge (op \rhd_{\tau} Open) \wedge (close \rhd_{\tau} \neg Open)$$

where s' is a sampling of s through the sensors, r' a sampling of r, op a command to open the valve and $close$ a command to close the valve. Remember that the sensors and the actuators serve as interface between the controller and its physical environment. The relationship between the inputs (states s' and r') and outputs (commands op and $close$) of the controller is formalised as $I_f \mathrel{\widehat{=}} (\neg s' \rhd_0 close) \wedge (\neg r' \rhd_0 op)$. That is the controller issues as soon as possible a command for closing or opening the valve whenever it observes the water reached the critical low or high level respectively. The following theorem says that how fast the controller could react, the valve might not be closed or open before $\delta + \tau$ time units from the instant the water reached the critical level.

Theorem 4 $A_{wl} \wedge I_f \vdash (\neg s \rhd_{\delta+\tau} \neg Open) \wedge (\neg r \rhd_{\delta+\tau} Open)$

The proof of the theorem is easy and we do not give it here. However this result is useful when defining a control strategy.

Instead of observing the water at critical levels, we will consider a safety low level ω_{sl} and high level ω_{sh} such that the valve should be closed (open) within $\theta \mathrel{\widehat{=}} \delta + \tau$ time units when the water level is below the safety low level ω_{sl} (above the safety high level ω_{sh} respectively). Our idea is illustrated in Figure 10.

Let $\tilde{s} \mathrel{\widehat{=}} \omega < \omega_{sl} \wedge Open$ and $\tilde{r} \mathrel{\widehat{=}} \omega > \omega_{sh} \wedge \neg Open$. The state variable $\tilde{r}$ holds if the valve is closed and the water is above the safety high level. The state variable $\tilde{s}$ holds if the valve is open and the water is below the safety low level. Then we consider the design decision that the valve should not be kept closed (open) for more than θ time units whenever the water is above the safety high level (below the safety low level respectively). This design decision is formalised as

$$D_{wl} \mathrel{\widehat{=}} \Box(\lceil \tilde{r} \rceil \Rightarrow \ell \leq \theta) \wedge \Box(\lceil \tilde{s} \rceil \Rightarrow \ell \leq \theta).$$

The following theorem says that the design decision D_{wl} is correct w.r.t. the specification S_{wl} provided $\omega_{sl} \leq \omega_0 \leq \omega_{sh}$ and $\omega_{sl} - \theta\lambda \geq \omega_{cl}$ and $\omega_{sh} + \theta\lambda \leq \omega_{ch}$.

Theorem 5 *If* $\omega_{sl} \leq \omega_0 \leq \omega_{sh}$ *and* $\omega_{sl} - \theta\lambda \geq \omega_{cl}$ *and* $\omega_{sh} + \theta\lambda \leq \omega_{ch}$ *then* $D_{wl} \Rightarrow S_{wl}$

Before giving the proof of theorem 5 we first prove the following two lemmas. Lemma 1 says that if the state $\tilde{r}$ does not last for a period longer than θ time units then r always holds provided $\omega_0 \leq \omega_{sh}$ and $\omega_{sh} + \theta\lambda \leq \omega_{ch}$.

Lemma 1 *If* $\omega_0 \leq \omega_{sh}$ *and* $\omega_{sh} + \theta\lambda \leq \omega_{ch}$ *then* $\Box(\lceil \tilde{r} \rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil r \rceil$

Proof (Lemma 1).

1	$\omega_0 \leq \omega_{sh}$	assumption
2	$\omega_{sh} + \theta\lambda \leq \omega_{ch}$	assumption
3	$\Box(\lceil\tilde{r}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil\omega - \omega_{sh} \leq \theta\lambda\rceil$	by 1, 2, DC^*, arithmetic
4	$\Box(\lceil\tilde{r}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil\omega \leq \omega_{sh} + \theta\lambda\rceil$	by 3, arithmetic
5	$\Box(\lceil\tilde{r}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil\omega \leq \omega_{ch}\rceil$	by 2, 4, arithmetic

□

Lemma 2 says that if the state $\tilde{s}$ does not last for a period longer than θ time units then s always holds provided $\omega_{sl} \leq \omega_0$ and $\omega_{sl} - \theta\lambda \geq \omega_{cl}$.

Lemma 2 *If* $\omega_{sl} \leq \omega_0$ *and* $\omega_{sl} - \theta\lambda \geq \omega_{cl}$ *then* $\Box(\lceil\tilde{s}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil s\rceil$

Proof (Lemma 2).

1	$\omega_{sl} \leq \omega_0$	assumption
2	$\omega_{sl} - \theta\lambda \geq \omega_{cl}$	assumption
3	$\Box(\lceil\tilde{s}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil\omega_{sl} - \omega \leq \theta\lambda\rceil$	by 1, 2, DC^*, arithmetic
4	$\Box(\lceil\tilde{s}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil\omega \geq \omega_{sl} - \theta\lambda\rceil$	by 3, arithmetic
5	$\Box(\lceil\tilde{s}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil\omega \geq \omega_{cl}\rceil$	by 2, 4, arithmetic

□

We now give the proof of Theorem 5.

Proof (Theorem 5).

1	$\omega_0 \leq \omega_{sh} \wedge \omega_{sh} + \theta\lambda \leq \omega_{ch}$	assumption
2	$\omega_{sl} \leq \omega_0 \wedge \omega_{sl} - \theta\lambda \geq \omega_{cl}$	assumption
3	$\Box(\lceil\tilde{r}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil r\rceil$	by 1, Lemma 1
4	$\Box(\lceil\tilde{s}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil s\rceil$	by 2, Lemma 2
5	$\Box(\lceil\tilde{r}\rceil \Rightarrow \ell \leq \theta) \wedge \Box(\lceil\tilde{s}\rceil \Rightarrow \ell \leq \theta) \Rightarrow \Box\lceil r\rceil \wedge \Box\lceil s\rceil$	by 3, 4
6	$D_{wl} \Rightarrow S_{wl}$	by 5, DC^*

□

The next step of the design process is to find a discrete specification of the control program. We sample the state variables $\tilde{r}$ and $\tilde{s}$ through sensors. To be observable, we require those states to be δ-stable. The assumptions about the behaviour of the environment, the behaviours of the sensors and the actuators are then formalised as

$$\tilde{A}_{wl} \mathrel{\widehat{=}} (\tilde{r}' \overrightarrow{\approx}_\delta \tilde{r}) \wedge (\tilde{s}' \overrightarrow{\approx}_\delta \tilde{s}) \wedge \delta\text{-stable}(\tilde{r}) \wedge \delta\text{-stable}(\tilde{s}) \wedge \delta\text{-stable}(\neg\tilde{r}) \wedge \delta\text{-stable}(\neg\tilde{s})$$

We first refine D_{wl} into a DC^* formula

$$D'_{wl} \mathrel{\widehat{=}} \Box(\lceil\!\lceil \tilde{r}' \rceil\!\rceil \Rightarrow \ell \leq \theta - \delta) \wedge \Box(\lceil\!\lceil \tilde{s}' \rceil\!\rceil \Rightarrow \ell \leq \theta - \delta)$$

over discrete states $\tilde{r}'$ and $\tilde{s}'$. Then we prove the correctness of this refinement step.

Lemma 3 $\tilde{A}_{wl} \vdash D'_{wl} \Rightarrow D_{wl}$

Proof (Lemma 3).

1	$\tilde{A}_{wl}$	assumption
2	$\Box(\lceil\!\lceil \tilde{r}' \rceil\!\rceil \Rightarrow \ell \leq \theta - \delta) \Rightarrow \Box(\lceil\!\lceil \tilde{r} \rceil\!\rceil \Rightarrow \ell \leq \theta)$	by 1, Rule 21
3	$\Box(\lceil\!\lceil \tilde{s}' \rceil\!\rceil \Rightarrow \ell \leq \theta - \delta) \Rightarrow \Box(\lceil\!\lceil \tilde{s} \rceil\!\rceil \Rightarrow \ell \leq \theta)$	by 1, Rule 21
4	$D'_{wl} \Rightarrow D_{wl}$	by 2, 3, PC

□

Furthermore we refine D'_{wl} into a SDC^* formula

$$I_{wl} \mathrel{\widehat{=}} ((\lceil\!\lceil \neg\tilde{r}' \rceil\!\rceil \wedge \lceil\!\lceil \neg\tilde{s}' \rceil\!\rceil)^\frown(\ell = 0 \vee (\lceil\!\lceil \tilde{r}' \rceil\!\rceil \wedge \ell \leq \theta - \delta))^\frown(\lceil\!\lceil \neg\tilde{r}' \rceil\!\rceil \wedge \lceil\!\lceil \neg\tilde{s}' \rceil\!\rceil)^\frown(\ell = 0 \vee (\lceil\!\lceil \tilde{s}' \rceil\!\rceil \wedge \ell \leq \theta - \delta)))^*$$

Then we establish the correctness of the refinement step.

Lemma 4 $I_{wl} \Rightarrow D'_{wl}$

Proof (Lemma 4).

1	$I_{wl} \Rightarrow \Box(\lceil\!\lceil \tilde{r}' \rceil\!\rceil \Rightarrow \ell \leq \theta - \delta)$	by Rule 29, DC^*
2	$I_{wl} \Rightarrow \Box(\lceil\!\lceil \tilde{s}' \rceil\!\rceil \Rightarrow \ell \leq \theta - \delta)$	by Rule 29, DC^*
3	$I_{wl} \Rightarrow D'_{wl}$	by 1, 2, PC

□

Finitely we get a correct implementation of the system w.r.t. its specification. The following theorem says that the discrete implementation I_{wl} satisfies the specification of the water monitoring system S_{wl} under assumptions $\tilde{A}_{wl}$ provided $\omega_{sl} \leq \omega_0 \leq \omega_{sh}$ and $\omega_{sl} - \theta\lambda \geq \omega_{cl}$ and $\omega_{sh} + \theta\lambda \leq \omega_{ch}$.

Theorem 6 $(\omega_{sl} \leq \omega_0) \wedge (\omega_0 \leq \omega_{sh}) \wedge (\omega_{sl} - \theta\lambda \geq \omega_{cl}) \wedge (\omega_{sh} + \theta\lambda \leq \omega_{ch}) \wedge \tilde{A}_{wl} \vdash I_{wl} \Rightarrow S_{wl}$

Proof (Theorem 6).

1	$(\omega_{sl} \leq \omega_0) \wedge (\omega_0 \leq \omega_{sh}) \wedge (\omega_{sl} - \theta\lambda \geq \omega_{cl}) \wedge (\omega_{sh} + \theta\lambda \leq \omega_{ch})$	assumption
2	A_{wl}	assumption
3	$I_{wl} \Rightarrow D'_{wl}$	by Lemma 4
4	$D'_{wl} \Rightarrow D_{wl}$	by 2, Lemma 3
5	$D_{wl} \Rightarrow S_{wl}$	by 1, Theorem 5
6	$I_{wl} \Rightarrow S_{wl}$	by 3, 4, 5, PC

□

6 Audio Control Protocol

In this section, we use our approach to specify and verify the correctness of the audio control protocol. The protocol uses the well-known Machester encoding of bit strings. Each bit is encoded as a cell consisting of $4Q$ cycles of the receiver clock. Bit 1 is encoded as an upgoing edge at the middle of the cell, and bit 0 is encoded as a downgoing edge at the middle of the cell. If the same bit is sent twice in a row then an additional edge is required, which is placed exactly in between the corresponding two cells (see Figure 11).

The receiver scans only upgoing edges using the Philips decoding. It has to decode the signal without seeing the downgoing edges. This is always possible except that a message ending with '10' cannot be distinguished from the same message ending with '1'. The problem is solved by requiring that all messages either end with '00' or have an odd length. In addition, all messages should begin with '1' for the receiver to detect the beginning of the messages.

The protocol has been studied in [21] using a digitized extension of DC. A special formula int has been introduced to formalise the digitisation of signal. Moreover the receiver's clock has been considered as reference time.

Our approach is more intuitive for many reasons. We do not consider any extension of the original DC. We only consider an approximation of continuous states by discrete ones in a natural way, formalising the causality relationship between continuous states variable and their samplings (see Section 3). We do consider as reference time the sender's clock, instead. So we can determine more precisely in term of sender's clock cycle the period in which the signal is unreliable. We then verify the correctness of the protocol easily using our verification rules of Section 4.2.

Let s be a state variable modelling the signal sent by the sender and r a state variable modelling the signal received by the receiver. We consider the following assumptions about the physical laws and the digitisation. If the sender changes the signal from low to high or from high to low, the receiver's signal is unreliable for δ cycles of the sender clock, $\delta \geq 1$, $\delta \in I\!N$ during which it can be any value chosen nondeterministically among 0 and 1. However if the sender keeps

the signal stable then the same signal will be stable for the receiver. These assumptions can be formalised by $r \overrightarrow{\approx}_\delta s$ as illustrated in Figure 9.

Since a state can be specified by a DC^* formula, a communication protocol can be mode-led as consisting of a language $\mathcal{L}_s$ whose elements are DC^* formulae over state variable s, a language $\mathcal{L}_r$ whose elements are DC^* formulae over state variable r, a coding function $f : \{0,1\}^+ \to \mathcal{L}_s$ and a decoding function $g : \mathcal{L}_r \to \{0,1\}^+$ such that

1. for all $w \in \{0,1\}^+$ there exists a formula $D \in \mathcal{L}_r$ for which $f(w) \Rightarrow D$, and
2. for any $D \in \mathcal{L}_r$, if $f(w) \Rightarrow D$ then $g(D) = w$.

There are only two possibilities for coding information: changes of the signal from high to low or low to high, which are called edges, and distances between edges. We model an edge by

1. $[\![s]\!]^\frown[\![\neg s]\!]$ or $[\![\neg s]\!]^\frown[\![s]\!]$ for the sender,
2. $[\![r]\!]^\frown[\![\neg r]\!]$ or $[\![\neg r]\!]^\frown[\![r]\!]$ for the receiver.

An edge of the sender, in an interval long enough, can introduce several edges of the receiver. For sake of simplicity we consider the following abbreviations, where a and b are positive integers and h a positive real number.

$$\begin{array}{lcl} Hi_Lo_s & \widehat{=} & ([\![s]\!] \wedge \ell = a)^\frown([\![\neg s]\!] \wedge \ell = b) \\ Lo_Hi_s & \widehat{=} & ([\![\neg s]\!] \wedge \ell = a)^\frown([\![s]\!] \wedge \ell = b) \end{array}$$

$$\begin{array}{lcl} Hi_Lo_r & \widehat{=} & ([\![r]\!] \wedge \ell > \delta)^\frown([\![\neg r]\!] \wedge \ell = h) \\ Lo_Hi_r & \widehat{=} & ([\![\neg r]\!] \wedge \ell > \delta)^\frown([\![r]\!] \wedge \ell = h) \end{array}$$

Manchester encoding

$f : \{0,1\}^+ \to \mathcal{L}_s$ in this case is a partial function only. Its domain is defined by a regular expression:

$$1(00 \oplus 01 \oplus 10 \oplus 11)^* \oplus 1(0 \oplus 1)^*00$$

Let f' be a homomorphism defined by

$$\begin{array}{lcl} f'(1) & \widehat{=} & Lo_Hi_s \\ f'(0) & \widehat{=} & Hi_Lo_s \\ f'(w0) & \widehat{=} & f'(w)^\frown f'(0) \\ f'(w1) & \widehat{=} & f'(w)^\frown f'(1) \end{array}$$

Then the Manchester encoding function f (Figure 11) is defined as

$$f(w) \mathrel{\widehat{=}} [\![\neg s]\!]^\frown f'(w)^\frown([\![\neg s]\!] \wedge \ell \geq c)$$

where $c \geq 1$ is a natural number.

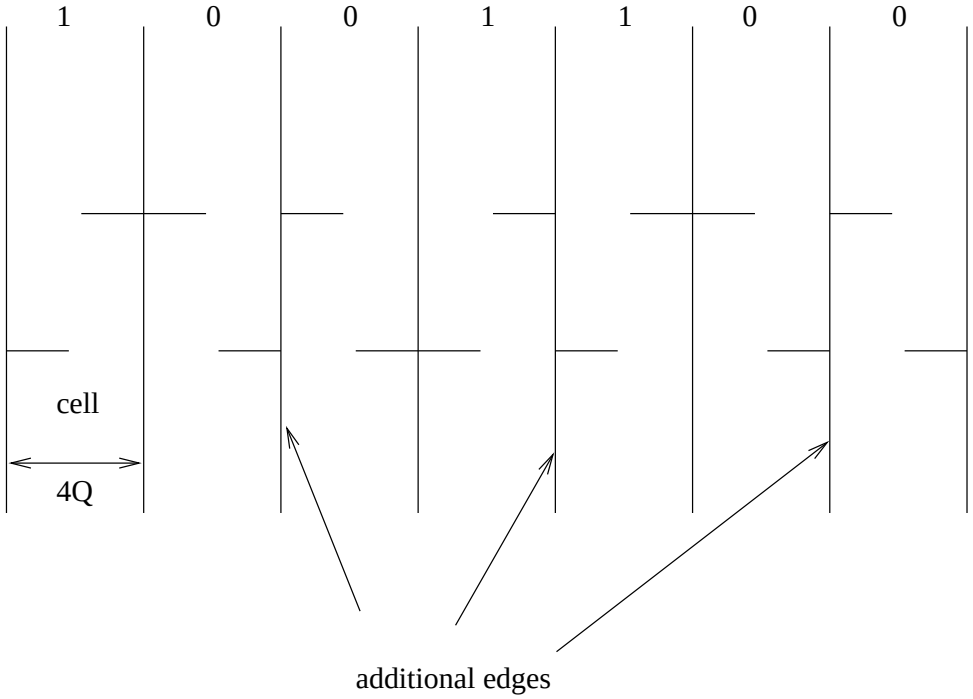


Figure 11: Manchester encoding

Philips decoding

Let us denote:

$$\begin{array}{lcl} FourH & \mathrel{\widehat{=}} & ((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 3Q \leq \ell < 5Q \\ SixH & \mathrel{\widehat{=}} & ((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 5Q \leq \ell < 7Q \\ EightH & \mathrel{\widehat{=}} & ((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 7Q \leq \ell \\ NineH & \mathrel{\widehat{=}} & (\neg\Diamond Lo_Hi_r) \wedge 9Q \leq \ell \end{array}$$

Let

$$\Sigma_r \mathrel{\widehat{=}} \{FourH, SixH, EightH, NineH, ^\frown, (\neg\Diamond Lo_Hi_r), Lo_Hi_r\}$$

and $\mathcal{L}_r$ be represented by the following regular expression over Σ_r:

$$\mathcal{L}_r \mathrel{\widehat{=}} (\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r(^\frown(FourH \oplus SixH \oplus EightH))^*{}^\frown NineH$$

Then, the decoding function $g : \mathcal{L}_r \to \{0,1\}^+$ is defined below. First, the function

$$g' : (\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r(^\frown(FourH \oplus SixH \oplus EightH))^* \to \{0,1\}^+$$

is defined inductively as follows.

Basic case:

$$g'((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \mathrel{\widehat{=}} 1.$$

Induction step: Assume that $g'(D)$ has been defined and $g'(D) = w0$.

$$\begin{array}{lcl} g'(D^\frown FourH) & \mathrel{\widehat{=}} & w00 \\ g'(D^\frown SixH) & \mathrel{\widehat{=}} & w01 \end{array}$$

Assume that $g'(D)$ has been defined and $g'(D) = w1$.

$$\begin{array}{lcl} g'(D^\frown FourH) & \mathrel{\widehat{=}} & w11 \\ g'(D^\frown SixH) & \mathrel{\widehat{=}} & w100 \\ g'(D^\frown EightH) & \mathrel{\widehat{=}} & w101 \end{array}$$

Now, the function g'' is defined as:

$$g''(D^\frown NineH) \mathrel{\widehat{=}} g'(D).$$

Let $Just : \{0,1\}^* \to \{0,1\}^*$ be a homomorphism defined by:

1. if $w = x1$ and $|w|$ is odd the $Just(w) = w$,
2. if $w = x1$ and $|w|$ is even the $Just(w) = w0$,
3. if $w = x0$ and $|w|$ is odd the $Just(w) = w$,
4. if $w = x0$ and $|w|$ is even then
   - if $x = y0$ then $Just(w) = w$,
   - if $x = y1$ then $Just(w) = w0$.

The Philips decoding function $g : \mathcal{L}_r \to \{0,1\}^*$ is defined by

$$g(D) \mathrel{\widehat{=}} Just(g''(D)).$$

For example, suppose that the sender wants to send the sequence of bits 1001010. The sequence is encoded as a DC^* formula f(1001010),

$$f(1001010) = [\![\neg s]\!]^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown Hi_Lo_s{}^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown([\![\neg s]\!] \wedge \ell \geq c).$$

If we can prove that $f(1001010) \Rightarrow D$, where

$$D \mathrel{\widehat{=}} (\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r{}^\frown SixH^\frown SixH^\frown EightH^\frown NineH,$$

then the same sequence 1001010 will be received by the receiver, since $g(D) = 1001010$.

We now verify the correctness of the protocol. The value c is not part of the protocol but only represents the gap between messages. It should be chosen consequently by the engineers to reduce the transmission time of a sequence of messages. The parameter h is a real number according to the sender's clock. We only need $h = 1$ according to the receiver's clock so that the receiver can detect a change of signal.

Verification

We have to find the values of a, b, δ, and prove that for those a, b, δ

1. For all $D, D' \in \mathcal{L}_r, D \neq D'$ implies that $D \wedge D' \Leftrightarrow false$, i.e at any time the receiver can detect at most one string in its domain.
2. For all $w \in dom(f)$, there is a formula $D \in \mathcal{L}_r$ such that $f(w) \Rightarrow D$ and $g(D) = w$, i.e for any input string in the domain of the sender the receiver can detect at least one string in its domain.

Lemma 5 *Let $D = A^\frown B \in \mathcal{L}_r$, $D' = A^\frown B' \in \mathcal{L}_r$, where $A, B, B' \in \Sigma_r^*$. Then $D \wedge D' \Rightarrow A^\frown(B \wedge B')$.*

This lemma can be proved easily by induction on the length $|A|$ of the word A. The first item follows directly from Lemma 5 with the note that for any $C, C' \in \Sigma_r, C \neq C'$ implies that $C \wedge C' \Rightarrow false$

Proof (Lemma 5).

Basic case: $|A| = 3$.

Then $A \mathrel{\widehat{=}} (\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r$.

From the definition of Lo_Hi_r, it follows that for any $B, B' \in \Sigma_r^*$

$$(A^\frown B) \wedge (A^\frown B') \Rightarrow A^\frown(B \wedge B').$$

Induction step: We assume that for any $A, B, B' \in \Sigma_r^*$ such that $3 \leq |A| \leq n$, if $A^\frown B$, $A^\frown B' \in \mathcal{L}_r$ then $(A^\frown B) \wedge (A^\frown B') \Rightarrow A^\frown(B \wedge B')$.

We must prove that

if $A^\frown X^\frown B$, $A^\frown X^\frown B' \in \mathcal{L}_r$ then $(A^\frown X^\frown B) \wedge (A^\frown X^\frown B') \Rightarrow A^\frown X^\frown(B \wedge B')$, for any $X \in \Sigma_r$.

Let us assume that $A^\frown X^\frown B$, $A^\frown X^\frown B' \in \mathcal{L}_r$. Then the possible values of X are $FourH$, $SixH$ and $EightH$.

$$\begin{array}{ll} (A^\frown X^\frown B) \wedge (A^\frown X^\frown B') & \\ \quad \Rightarrow (A^\frown(X^\frown B)) \wedge (A^\frown(X^\frown B')) & \{A2\} \\ \quad \Rightarrow A^\frown((X^\frown B) \wedge (X^\frown B')) & \{\text{induction hypothesis}\} \end{array}$$

It is easy to prove that $(X^\frown B) \wedge (X^\frown B') \Rightarrow X^\frown(B \wedge B')$ for any $X \in \{FourH, SixH, EightH\}$.

It follows that

$$(A^\frown X^\frown B) \wedge (A^\frown X^\frown B') \Rightarrow A^\frown X^\frown(B \wedge B').$$

□

Lemma 6 *If $a = b$, $a > 2\delta$, $Q > \delta$ and $7Q + \delta \leq 4a \leq 9Q - \delta$ then for any $w \in \{0,1\}^*$,*

- *$w = x1$ implies that there exists $D' \in \mathcal{L}_r$ such that $f(w) \Rightarrow D'$ and $g''(D') = x1$,*
- *$w = x10$ implies that there exists $D' \in \mathcal{L}_r$ such that $f(w) \Rightarrow D'$ and $g''(D') = x1$,*
- *$w = x00$ implies that there exists $D' \in \mathcal{L}_r$ such that $f(w) \Rightarrow D'$ and $g''(D') = x00$,*

Lemma 6 can be proved easily by induction on the length $|w|$ of the string w.

Item 2 then follows immediately from Lemma 6 and the definition of the function g.

We now give the proof of Lemma 6. For simplifying the proof, we consider the following abbreviations.

$A \mathrel{\widehat{=}} [\![\neg s]\!]^\frown f'(x)$

$B \mathrel{\widehat{=}} \ell \leq \delta^\frown([\![\neg r]\!] \wedge \ell \geq c - \delta)$

$C \mathrel{\widehat{=}} \ell \leq \delta - h^\frown([\![r]\!] \wedge \ell \geq a - \delta - h)$

$ass \mathrel{\widehat{=}} (7Q + \delta \leq 4a) \wedge (4a \leq 9Q - \delta) \wedge (Q > \delta)$

Proof (Lemma 6).

Basic case: $|w| = 1$.

This implies that $w = 1$.

$$
\begin{array}{lll}
f(w) & = [\![\neg s]\!]^\frown Lo_Hi_s{}^\frown([\![\neg s]\!] \wedge \ell \geq c) & \\
& \Rightarrow (\ell \leq \delta)^\frown([\![\neg r]\!] \wedge \ell > a - \delta)^\frown \ell \leq \delta^\frown & \\
& \quad ([\![r]\!] \wedge \ell \geq a - \delta)^\frown B & \{\text{Def. 8, } a = b,\ DC^*\} \\
& \Rightarrow (\ell \leq \delta)^\frown([\![\neg r]\!] \wedge \ell > a - \delta)^\frown & \\
& \quad ([\![r]\!] \wedge \ell = h)^\frown C^\frown B & \{\text{Rule 15, } a > 2\delta,\ \delta \geq h,\ DC^*\} \\
& \Rightarrow (\ell \leq \delta)^\frown([\![\neg r]\!] \wedge \ell > \delta)^\frown & \\
& \quad ([\![r]\!] \wedge \ell = h)^\frown C^\frown B & \{a > 2\delta,\ DC^*\} \\
& \Rightarrow (\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r{}^\frown & \\
& \quad ((\neg\Diamond Lo_Hi_r) \wedge \ell \geq a + c - \delta - h) & \{\text{arithmetic, } DC^*\} \\
& \Rightarrow (\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r{}^\frown NineH & \{a + c - \delta - h \geq 9Q\}
\end{array}
$$

$g''((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r{}^\frown NineH) = 1$

Induction step: We suppose that for any $w \in dom(f)$ such that $|w| \leq n$, there exists D such that $f(w) \Rightarrow D$ and $g''(D) = w'$, where

$$
w' = \begin{cases} x1 \ \texttt{if} \ w = x1 \\ x1 \ \texttt{if} \ w = x10 \\ x00 \ \texttt{if} \ w = x00 \end{cases}
$$

We must prove that for any $w \in dom(f)$, $|w| = n + 1$ there exists $D' \in \mathcal{L}_r$ such that $f(w) \Rightarrow D'$ and $g''(D') = w'$, where w' is defined as above.

1. $w = x000$.

   Then there exists $D \in \mathcal{L}_r$ such that $f(x00) \Rightarrow D$ and $g''(D) = x00$.

   Let $D \mathrel{\widehat{=}} D''{}^\frown NineH$.

$$
\begin{array}{lll}
f(x00) & = A^\frown Hi_Lo_s{}^\frown Hi_Lo_s{}^\frown([\![\neg s]\!] \wedge \ell \geq c) & \\
& \Rightarrow A^\frown \ell \leq \delta^\frown([\![r]\!] \wedge \ell \geq a - \delta)^\frown \ell \leq \delta^\frown & \{\text{Def. 8,} \\
& \quad ([\![\neg r]\!] \wedge \ell \geq a - \delta)^\frown \ell \leq \delta^\frown([\![r]\!] \wedge & a = b, \\
& \quad \ell \geq a - \delta)^\frown \ell \leq \delta^\frown([\![\neg r]\!] \wedge \ell \geq a - \delta)^\frown B & DC^*\} \\
& \Rightarrow A^\frown \ell \leq \delta^\frown([\![r]\!] \wedge \ell \geq a - \delta)^\frown \ell \leq \delta^\frown & \{\text{Rule 15,} \\
& \quad ([\![\neg r]\!] \wedge \ell \geq a - \delta)^\frown([\![r]\!] \wedge \ell = h)^\frown & a > 2\delta, \\
& \quad C^\frown \ell \leq \delta^\frown([\![\neg r]\!] \wedge \ell \geq a - \delta)^\frown B & \delta \geq h,\ DC^*\} \\
& \Rightarrow A^\frown(\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r{}^\frown C^\frown & \{a > 2\delta,\ DC^*, \\
& \quad \ell \leq \delta^\frown([\![\neg r]\!] \wedge \ell \geq a - \delta)^\frown B & \text{arithmetic}\} \\
& \Rightarrow D''{}^\frown C^\frown \ell \leq \delta^\frown([\![\neg r]\!] \wedge \ell \geq a - \delta)^\frown B & \{\text{Def. 8, } DC^*\}
\end{array}
$$

$$
\begin{array}{lll}
f(x000) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown Hi_Lo_s^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket & \{\text{Def. 8}, \\
& \quad \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r\rrbracket & \{\text{Rule 15}, \\
& \quad \wedge \ell = h)^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 2a-\delta \leq \ell \leq & \\
& \quad 2a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq 2a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown FourH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown FourH^\frown NineH) = x000$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown FourH^\frown NineH.$

2. $w = x100.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x1) \Rightarrow D$ and $g''(D) = x1$. Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$
\begin{array}{lll}
f(x1) & = A^\frown Lo_Hi_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow A^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \{\text{Def. 8}, \\
& \quad \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow A^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \{\text{Rule 15}, \delta \geq h, \\
& \quad (\llbracket r\rrbracket \wedge \ell = h)^\frown C^\frown B & a > 2\delta, DC^*\} \\
& \Rightarrow A^\frown(\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r^\frown C^\frown B & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown C^\frown B & \{DC^*\}
\end{array}
$$

$$
\begin{array}{lll}
f(x100) & = A^\frown Lo_Hi_s^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \\
& \quad (\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta) & \{\text{Def. 8}, \\
& \quad ^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \\
& \quad (\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r\rrbracket \wedge \ell = h)^\frown C^\frown & \{\text{Rule 15}, \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 3a-\delta \leq \ell \leq & \\
& \quad 3a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq 2a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown SixH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown SixH^\frown NineH) = x100$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown SixH^\frown NineH.$

3. $w = x110.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x1) \Rightarrow D$ and $g''(D) = x1$.

   Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$\begin{array}{lll} f(x1) & = A^\frown Lo_Hi_s^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq c) & \\ & \Rightarrow D''^\frown C^\frown B & \{\text{cf. } 2\} \end{array}$$

$$\begin{array}{lll} f(x110) & = A^\frown Lo_Hi_s^\frown Lo_Hi_s^\frown Hi_Lo_s^\frown(\llbracket \neg s \rrbracket \wedge \ell \geq c) & \\ & \Rightarrow D''^\frown C^\frown Lo_Hi_s^\frown Hi_Lo_s^\frown B & \{DC^*\} \\ & \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq a - \delta)^\frown & \{\text{Def. } 8, \\ & \quad (\llbracket r \rrbracket \wedge \ell = h)^\frown C^\frown \ell \leq \delta^\frown(\llbracket r \rrbracket \wedge \ell \geq a - \delta)^\frown & \text{Rule } 15, \\ & \quad \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq a - \delta)^\frown B & DC^*\} \\ & \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 2a - \delta \leq \ell \leq & \\ & \quad 2a + \delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq 3a + c - \delta - h) & \{\text{arithmetic}\} \\ & \Rightarrow D''^\frown FourH^\frown NineH & \{ass\} \end{array}$$

$g''(D''^\frown FourH^\frown NineH) = x11$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown FourH^\frown NineH.$

4. $w = x0010.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x00) \Rightarrow D$ and $g''(D) = x00.$

   Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$\begin{array}{lll} f(x00) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown(\llbracket \neg s \rrbracket \wedge \ell \geq c) & \\ & \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq a - \delta)^\frown B & \{\text{cf. } 1\} \end{array}$$

$$\begin{array}{lll} f(x0010) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown Lo_Hi_s^\frown & \\ & \quad Hi_Lo_s^\frown(\llbracket \neg s \rrbracket \wedge \ell \geq c) & \\ & \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq a - \delta)^\frown & \\ & \quad Lo_Hi_s^\frown Hi_Lo_s^\frown B & \{DC^*\} \\ & \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq a - \delta)^\frown & \{\text{Def. } 8, \\ & \quad \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge \ell \geq a - \delta)^\frown(\llbracket r \rrbracket \wedge \ell = h) & \text{Rule } 15, \\ & \quad ^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq 3a + c - \delta - h) & DC^*\} \\ & \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 3a - \delta \leq \ell \leq & \\ & \quad 3a + \delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq 3a + c - \delta - h) & \{\text{arithmetic}\} \\ & \Rightarrow D''^\frown SixH^\frown NineH & \{ass\} \end{array}$$

$g''(D''^\frown SixH^\frown NineH) = x001$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown SixH^\frown NineH.$

5. $w = x1010.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x10) \Rightarrow D$ and $g''(D) = x1.$

   Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$
\begin{array}{lll}
f(x10) & = A^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow A^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \{\text{Def. 8,} \\
& \quad \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & a=b, \\
& \quad (\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow A^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r\rrbracket \ell = h)^\frown & \\
& \quad C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta) & \{\text{Rule 15,} \\
& \quad {}^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & \delta \geq h,\ DC^*\} \\
& \Rightarrow A^\frown(\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r{}^\frown C^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta) & \{a > 2\delta,\ DC^*, \\
& \quad {}^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & \text{arithmetic}\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & \{DC^*\}
\end{array}
$$

$$
\begin{array}{lll}
f(x1010) & = A^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown Lo_Hi_s{}^\frown & \\
& \quad Hi_Lo_s{}^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge & \\
& \quad \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \{\text{Def. 8,} \\
& \quad (\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \{\text{Rule 15,} \\
& \quad (\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge & \delta \geq h, \\
& \quad \ell \geq a-\delta)^\frown(\llbracket r\rrbracket \wedge \ell = h)^\frown((\neg\Diamond Lo_Hi_r)\wedge & a > 2\delta, \\
& \quad \ell \geq 3a+c-\delta-h) & DC^*\} \\
& \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 4a-\delta \leq \ell \leq & \\
& \quad 4a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq 3a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown EightH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown EightH^\frown NineH) = x101$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown EightH^\frown NineH.$

6. $w = x111.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x1) \Rightarrow D$ and $g''(D) = x1.$

   Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$
\begin{array}{lll}
f(x1) & = A^\frown Lo_Hi_s{}^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown B & \{\text{cf. 2}\}
\end{array}
$$

$$
\begin{array}{lll}
f(x111) & = A^\frown Lo_Hi_s^\frown Lo_Hi_s^\frown Lo_Hi_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown Lo_Hi_s^\frown Lo_Hi_s^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \{\text{Def. 8,} \\
& \quad (\llbracket r\rrbracket \wedge \ell = h)^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \text{Rule 15,} \\
& \quad (\llbracket r\rrbracket \wedge \ell = h)^\frown C^\frown B & DC^*\} \\
& \Rightarrow D''((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 2a-\delta \leq \ell \leq & \\
& \quad 2a+\delta)^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 2a-\delta \leq \ell & \\
& \quad \leq 2a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown FourH^\frown FourH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown FourH^\frown FourH^\frown NineH) = x111$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown FourH^\frown FourH^\frown NineH.$

7. $w = x101.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x1) \Rightarrow D$ and $g''(D) = x1.$

   Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$
\begin{array}{lll}
f(x1) & = A^\frown Lo_Hi_s^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown B & \{\text{cf. 2}\}
\end{array}
$$

$$
\begin{array}{lll}
f(x101) & = A^\frown Lo_Hi_s^\frown Hi_Lo_s^\frown Lo_Hi_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown Hi_Lo_s^\frown Lo_Hi_s^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \{\text{Def. 8,} \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \{\text{Rule 15,} \\
& (\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r\rrbracket \wedge \ell = h)^\frown C^\frown B & DC^*\} \\
& \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 4a-\delta \leq \ell \leq & \\
& \quad 4a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown EightH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown EightH^\frown NineH) = x101$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown EightH^\frown NineH.$

8. $w = x001.$

   Then there exists $D \in \mathcal{L}_r$ such that $f(x00) \Rightarrow D$ and $g''(D) = x00.$

   Let $D \mathrel{\widehat{=}} D''^\frown NineH.$

$$
\begin{array}{lll}
f(x00) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & \{\text{cf. 1}\}
\end{array}
$$

$$
\begin{array}{lll}
f(x001) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown Lo_Hi_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad Lo_Hi_s^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \{\text{Def. 8,} \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r\rrbracket \wedge \ell = h)^\frown & \text{Rule 15,} \\
& ((\neg\Diamond Lo_Hi_r) \wedge \ell \geq a+c-\delta-h) & DC^*\} \\
& \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 3a-\delta \leq \ell \leq & \\
& \quad 3a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown SixH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown SixH^\frown NineH) = x001$

Then we define D' as $D' \mathrel{\widehat{=}} D''^\frown SixH^\frown NineH$.

9. $w = x0011$.

   Then there exists $D \in \mathcal{L}_r$ such that $f(x00) \Rightarrow D$ and $g''(D) = x00$.

   Let $D \mathrel{\widehat{=}} D''^\frown NineH$.

$$
\begin{array}{lll}
f(x00) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & \{\text{cf. 1}\}
\end{array}
$$

$$
\begin{array}{lll}
f(x0011) & = A^\frown Hi_Lo_s^\frown Hi_Lo_s^\frown Lo_Hi_s^\frown & \\
& \quad Lo_Hi_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad Lo_Hi_s^\frown Lo_Hi_s^\frown B & \{DC^*\} \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r\rrbracket \wedge \ell = h)^\frown & \{\text{Def. 8,} \\
& C^\frown \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown & \text{Rule 15,} \\
& \quad (\llbracket r\rrbracket \wedge \ell = h)^\frown C^\frown B & DC^*\} \\
& \Rightarrow D''^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 3a-\delta \leq \ell \leq & \\
& \quad 3a+\delta)^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 2a-\delta & \\
& \quad \leq \ell \leq 2a+\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge & \\
& \ell \geq a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''^\frown SixH^\frown FourH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''^\frown SixH^\frown FourH^\frown NineH) = x0011$

Then we define D' as

$D' \mathrel{\widehat{=}} D''^\frown SixH^\frown FourH^\frown NineH$.

10. $w = x1011$.

    Then there exists $D \in \mathcal{L}_r$ such that $f(x10) \Rightarrow D$ and $g''(D) = x1$.

    Let $D \mathrel{\widehat{=}} D''^\frown NineH$.

$$
\begin{array}{lll}
f(x10) & = A^\frown Lo_Hi_s^\frown Hi_Lo_s^\frown(\llbracket\neg s\rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''^\frown C^\frown \ell \leq \delta^\frown(\llbracket r\rrbracket \wedge \ell \geq a-\delta)^\frown & \\
& \quad \ell \leq \delta^\frown(\llbracket\neg r\rrbracket \wedge \ell \geq a-\delta)^\frown B & \{\text{cf. 5}\}
\end{array}
$$

$$
\begin{array}{lll}
f(x1011) & = A^\frown Lo_Hi_s{}^\frown Hi_Lo_s{}^\frown Lo_Hi_s{}^\frown & \\
& \quad Lo_Hi_s{}^\frown(\llbracket \neg s \rrbracket \wedge \ell \geq c) & \\
& \Rightarrow D''{}^\frown C^\frown \ell \leq \delta^\frown(\llbracket r \rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \\
& \quad (\llbracket \neg r \rrbracket \wedge \ell \geq a-\delta)^\frown Lo_Hi_s{}^\frown Lo_Hi_s{}^\frown B & \{DC^*\} \\
& \Rightarrow D''{}^\frown C^\frown \ell \leq \delta^\frown(\llbracket r \rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown & \\
& \quad (\llbracket \neg r \rrbracket \wedge \ell \geq a-\delta)^\frown \ell \leq \delta^\frown(\llbracket \neg r \rrbracket \wedge & \{\text{Def. 8,} \\
& \quad \ell \geq a-\delta)^\frown(\llbracket r \rrbracket \wedge \ell = h)^\frown C^\frown \ell \leq \delta^\frown & \text{Rule 15,} \\
& (\llbracket \neg r \rrbracket \wedge \ell \geq a-\delta)^\frown(\llbracket r \rrbracket \wedge \ell = h)^\frown C^\frown B & DC^*\} \\
& \Rightarrow D''{}^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 4a-\delta \leq \ell \leq & \\
& \quad 4a+\delta)^\frown(((\neg\Diamond Lo_Hi_r)^\frown Lo_Hi_r) \wedge 2a-\delta \leq & \\
& \ell \leq 2a-\delta)^\frown((\neg\Diamond Lo_Hi_r) \wedge \ell \geq a+c-\delta-h) & \{\text{arithmetic}\} \\
& \Rightarrow D''{}^\frown EightH^\frown FourH^\frown NineH & \{ass\}
\end{array}
$$

$g''(D''{}^\frown EightH^\frown FourH^\frown NineH) = x1011$

Then we define D' as

$D' \mathrel{\widehat{=}} D''{}^\frown EightH^\frown FourH^\frown NineH.$

□

Reasoning about clock rates of the sender and the reader

Our design technique is more useful as it can allow to reason about the parameters of the system. We have shown that the protocol is correct provided $a > 2\delta$, $Q > \delta$ and $7Q + \delta \leq 4a \leq 9Q - \delta$. We have considered the sender's clock as reference time. Therefore the numbers a, δ and Q are defined according to the sender's clock. However Q should be given according to the receiver's clock as it is used to decode the received signal. Let us denote by ϵ the ratio of the sender's clock rate and receiver's clock rate. If Q' is the value of Q according to the receiver's clock, then $Q = Q'\epsilon$ according to the sender's clock. Given a, δ and Q', we can determine the ratio of the rates of the sender's and receiver's clock for which the protocol is correct. However if

$$\frac{4a+\delta}{9Q'} \leq \epsilon \leq \frac{4a-\delta}{7Q'}$$

then the inequalities $7Q + \delta \leq 4a \leq 9Q - \delta$ are satisfied and the protocol is correct provided $a > 2\delta$ and $Q > \delta$.

For example let $a = 7$, $\delta = 3$ and $Q' = 4$. Then if the ratio of the rates of the sender's and receiver's clock is between $31/36$ and $25/28$ then the protocol is still correct.

7 Conclusion

In this work we have presented a syntactical approach for designing real-time distributed systems that can handle both continuous time and discrete time models in a uniform framework.

We have defined the concept of *observation* to capture the causality relationship between state variables. we have shown that this concept can be used to formalise the behaviour of sensors sampling a continuous state at discrete time points, or to model communication at physical layer between two components of a distributed system.

We have also defined the concept of *control* and have used it to formalise the behaviour of actuators in a controlled system. So the interface between the control program and the physical environment is formalised.

Given a specification of the requirements of a real-time system, our approach allows to derive a discrete specification of the control program satisfying the continuous specification of the system under some assumptions about the behaviours of the environment, the sensors and the actuators. The main idea of our approach is to model the discretization at the state level by introducing the discrete states approximating the continuous ones.

We have developed rules useful for refining a specification toward a discrete implementation and to verify the correctness of an implementation syntactically. Those rules can be implemented (in PVS for example) to assist the verification of designs.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic Symbolic Verification of Embedded Systems. In *14th Annual IEEE Real-time Systems Symposium*, pages 2–11, 1993.

[3] Eugene Asarin, Paul Caspi, and Oded Maler. A kleene theorem for timed automata. In G. Winskel, editor, *IEEE International Symposium on Logics in Computer Science LICS'97*, pages 160–171, 1997.

[4] Silvano Balemi. Input/output discrete event processes and system modeling. In S. Balemi, P. Kozák, and R. Smedinga, editors, *Progess in Systems and Control Theory*, volume 13. Birkhäuser, Basel-Boston-Berlin, 1993.

[5] Gérard Berry. The foundations of esterel.

[6] D. Bosscher, I. Polak, and F. Vaandrager. Verification of an Audio Control Protocol. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Compoter Science*, pages 170–192. Springer-Verlag, 1994.

[7] Antonio Cerone, Alex J. Cowie, George J. Mile, and Philip A. Moseley. Modelling a Time-Dependent Protocol Using the Circal Process Algebra. In Oded Maler, editor, *Hybrid and Real-Time Systems*, LNCS 1201. Springer-Verlag, 1997.

[8] Zhou Chaochen, C.A.R. Hoare, and Anders P. Ravn. A calculus of duration. *Information Processing Letters*, 40(5):269–276, 1991.

[9] B. Dutertre. On First Order Interval Temporal Logic . Technical Report CSD-TR94-3, Department of Computer science, Royal holloway, university of London, Egham, Surrey TW20 0EX, England, 1995.

[10] C. J. Fidge. Modelling discrete behaviour in a continuous-time formalism. In *Integrated Formal Methods'99*. Springer-Verlag, 1999.

[11] Dimitar P. Guelev. Iteration of Simple Formulas in Duration Calculus. Technical Report 141, UNU/IIST, P.O.Box 3058 Macau, 1998.

[12] M. R. Hansen and Zhou Chaochen. Semantics and Completeness of Duration Calculus. In *Real-Time: Theory and Practice*, LNCS 600, pages 209–225. Springer-Verlag, 1992.

[13] Michael R. Hansen and Zhou Chaochen. Duration calculus: Logical foundations. *Formal Aspects of Computing*, 9:283–330, 1997.

[14] Thomas A. Henzinger, Zohar Manna, and Amir Pneuli. What good are digital clocks? In *19th International Colloquium on Atomata, Languages, and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558. Springer-Verlag, 1992.

[15] Thomas A. Henzinger, Zohar Manna, and Amir Pneuli. Towards refining temporal specifications into hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, volume 736 of *Lecture Notes in Computer Science*, pages 60–76. Springer-Verlag, 1993.

[16] Thomas A. Henzinger, Zohar Manna, and Amir Pneuli. Temporal Proof Methodologies for Timed Transition Systems. *Information and Computation*, 112:273–337, 1994.

[17] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[18] Dang Van Hung. Modelling and verification of biphase mark protocols in duration calculus using pvs/dc. In *1998 International Conference on Application of Concurrency to System Design (CSD'98)*, pages 88–98, Aizu-Wakamatsu, Fukushima, Japan, 23-26 March 1998 1998. IEEE computer Society Press.

[19] Dang Van Hung and Phan Hong Giang. A sampling semantics of duration calculus. In Bengt Jonsson and Joachim Parrow, editors, *Formal Techniques for Real-Time and Fault Tolerant Systems*, volume 1135 of *LNCS*, pages 188–207. Springer-Verlag, 1996.

[20] Dang Van Hung and Dimitar P. Guelev. Completeness of a fragment of Duration Calculus with Iteration. In *Proceedings of Asian Computing Science Conference (ASIAN'99)*. Phuket, Thailand, 1999.

[21] Dang Van Hung and Ko Kwang Il. Verification via Digitazed Models of Real-Time Hybrid Systems. In *Proceedings of Asia-Pacific Software Engineering Conference (APSEC'96)*, pages 4–15. IEEE Computer Society Press, 1996.

[22] Dang Van Hung and Wang Ji. On the design of hybrid control systems using automata model. In V. Chandru and V. Vinay, editors, *Foundations of Software Technology and Theorical Computer Sciences*, volume 1180 of *LNCS*, pages 156–167. Springer-Verlag, December 1996.

[23] Bredan Mahony and Ian Hayes. Using Continuous Real Functions to Model Timed Histories. Technical report, Department of Computer Science, University of Queensland, Australia, 1991.

[24] Jay S. Moore. A Formal Model of Asynchronous Communication and Its Use in Mechanically Verifying a Biphase Mark Protocol. *Formal Aspects of Computing*, 6, 1994.

[25] J. Shoenfield. *Mathematical logic*. Addison-Wesley, Reading, Massachusetts, 1967.

[26] Graeme Smith. Specification and refinement of a real-time control system. In *Australian Computer Science Conference'99-Australian Computer Science Communications*, number 1 in 21. Springer-Verlag, 1999.

View publication stats