Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 207 (2008) 107–120
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

On Verification of Linear Occurrence Properties of Real-Time Systems

Choe Changil and Dang Van Hung

International Institute for Software Technology
United Nations University
Macau, China

Abstract

Duration Calculus of Weakly Monotonic Time (WDC) is an extension of DC to allow description of discrete processes where several steps of computation can occur at the same time point. In this paper, we introduce Linear Occurrence Invariants (LOI) using WDC and give an algorithm to check real-time automata for LOI by solving integer programming problems. LOI can be used effectively to specify system requirements in some cases including when the system is considered under the true synchrony assumption. We also extend WDC probabilistically to express dependability requirements of real-time systems and develop a technique to check deterministic probabilistic real-time automata for a class of probabilistic WDC formulas.

Keywords: linear occurrence invariants, real-time automata, duration calculus of weakly monotonic time, deterministic probabilistic real-time automata. probabilistic duration calculus.

1 Introduction

Duration Calculus (DC) was introduced in [1] as a logic for specification of real-time systems. It is then developed further in many other works that have been summarized in the monograph published recently [8]. Linear Duration Invariants (LDI) [4] is a decidable subclass of DC formulas, and many works were devoted to the verification of the requirements of real-time systems specified as a LDI, as well as to find out effective algorithms checking various models of real-time systems for LDI [4], [10], [11], [12], [12], [13].

The original DC was intended to specify the requirements of real-time systems. The externally observable behaviors of the system are specified in DC and the internal behaviors of the system may be hidden. However, the system can pass through a number of states within zero time when the behaviors of system are considered under the true synchrony assumption. To deal with such behaviors,

[1] Email: {cci,dvh}@iist.unu.edu

doi:10.1016/j.entcs.2008.03.088

a kind of logical extension of DC, called Duration Calculus of Weakly Monotonic Time (WDC) was suggested and a novel semantics of Timed CSP assuming that the communication and computation take no time was formulated using it [7]. WDC includes new formulas which can be used effectively to describe low level behaviors of system, as well as conserving DC formulas.

In this paper, we introduce Linear Occurrence Invariants (LOI) using WDC and give an algorithm to check real-time automata for LOI by solving integer programming problems. An LOI has the form $c_{min} \leq \ell \leq c_{max} \Rightarrow \sum_{i=1}^{n} k_i \cdot \sum P_i \leq M$ where $\sum P_i$ is the number of occurrences of P_i in the observation time interval. As an example of LOI specification, a property for the communication systems "for any observation interval, the failure rate of transmission should not be more than 10 percent of the number of transmissions" can be represented as $\ell \geq 0 \Rightarrow 90 \cdot \sum failure - 10 \cdot \sum success \leq 0$. It is obvious that LDI having the semantics based on the state duration can not specify this kind of properties for the system model in which several states can occur at the same time point. We believe LOI could be used in like as successfully as LDI in many cases where the systems are considered under the true synchrony assumption.

We also extend WDC to a logic named Probabilistic Duration Calculus of Weakly Monotonic Time (PWDC) to express dependability requirements of real-time systems, such as "with probability 0.7, sender transmits data frames without failure in any observation interval". The way of extension follows the recent work of Kwiatkowska et al [9] to extend CTL to a probabilistic timed CTL. In [9], authors proposed a variant of probabilistic timed automata that allows probabilistic choice only at discrete transitions and used the concept of adversary to resolve the nondeterminism between the passage of time and discrete transitions.

We consider deterministic probabilistic real-time automata model of real-time systems, having nondeterministic choice only for times, which is a subclass of probabilistic real-time automata. The extended logic PWDC consists of formulas representing the constraints for the probability of the satisfaction of a WDC formula by a set of adversaries of the underlying model of a deterministic probabilistic real-time automaton for an observation interval. We then develop techniques to check deterministic probabilistic real-time automata for some subclass of PWDC formulas.

2 Linear Occurrence Invariants and Checking Real-Time Automata against Linear Occurrence Invariants

In this section, we introduce *linear occurrence invariants (LOI)* and describe an algorithm to check a real-time automaton for a LOI using integer programming. We use WDC [7] to define LOI. We will recall WDC when we introduce the probabilistic WDC in the next section, but for now, we consider WDC formulas as DC formulas with the extension that we allow a term to be the number of occurrences of a state variable as well, and assume that several state changes can happen at the same time.

Definition 2.1 A formula of the form

$$\Theta \mathrel{\widehat{=}} c_{min} \leq \ell \leq c_{max} \Rightarrow \sum_{i=1}^{n} k_i \cdot \sum P_i \leq M$$

is called a linear occurrence invariant (LOI), where c_{min} and c_{max} are nonnegative real numbers, c_{max} could be ∞, $k_i\,(1 \leq i \leq n)$ and M are integer numbers, and $P_i\,(1 \leq i \leq n)$ are atomic propositions, $\sum P_i$ denotes the number of occurrences of state P_i in the reference interval.

The meanings of an LOI is that if the length of the observation interval is in between c_{min} and c_{max}, the numbers of occurrences of states in the observation interval satisfy the linear constraint $\sum_{i=1}^{n} k_i \cdot \sum P_i \leq M$. The difference between LOI and LDI is that the former has time-dependent premise and time-independent consequence, but both of premise and consequence of the latter are time-dependent. LOI is not a DC formula. It is a formula of WDC which is considered in Section 4. In continuous time DC, a state duration in an observation interval is defined as the integration of times in which state occurs. LDI which is a linear constraint on the state durations can not distinguish state changes occurring at the same time point when we consider systems under the true synchrony assumption. As we explained in Section 1, LOI will be especially useful for the system models in which the system passes through a number of states within zero time.

Now we describe an algorithm based on the integer programming to check real-time automata for linear occurrence invariants. Let $\mathcal{I} = \{\,[a,b] \in \mathbb{R} \times (\mathbb{R} \cup \{\infty\}) \mid a \leq b\,\}$ where $\mathbb{R}$ is the set of nonnegative real numbers. We consider $[a,b]$ as a closed interval on $\mathbb{R}$ if $b \in \mathbb{R}$, and semi-infinite interval on $\mathbb{R}$ otherwise. Let AP be the set of atomic propositions. Real-time automata is a subclass of timed automata of [2], where each automaton has one clock which is reset after every transition.

Definition 2.2 A real-time automaton $\mathcal{V}$ is a tuple (S, T, L) consisting of

- a finite set S of states,
- a transition relation $T \subseteq S \times \mathcal{I} \times S$.
- a function $L : S \to 2^{AP}$ assigning to each state $s \in S$ the set of atomic propositions which are true in s.

In [4], authors had to assume $b > 0$ for the time constraints of the form $[0, b]$ for a transition, when they develop an algorithm to check real-time automata for linear duration invariants using linear programming. We don't have this assumption for real-time automata in this section. We also consider that every state of a real-time automaton is both an initial state and an accepting state.

For a transition $\rho = (s, [a,b], s')$, the notations $\overleftarrow{\rho} = s$ and $\overrightarrow{\rho} = s'$ are used. $Seq = \rho_1\rho_2...\rho_m$ is called a *sequence* and $TSeq = (\rho_1, t_1)(\rho_2, t_2)...(\rho_m, t_m)$ is called a *time-stamped sequence*, in which $\rho_i = (s_i, [a_i, b_i], s_i')$ and $t_i \in [a_i, b_i]$ for all $i\,(1 \leq i \leq m)$. If a sequence $\rho_1\rho_2...\rho_m$ satisfies $\overrightarrow{\rho_i} = \overleftarrow{\rho}_{i+1}$ for all $i\,(1 \leq i < m)$, it is called a *behavior* and denoted by $Beh = \rho_1\rho_2...\rho_m$. If a time-stamped sequence $(\rho_1, t_1)(\rho_2, t_2)...(\rho_m, t_m)$ satisfies $\overrightarrow{\rho_i} = \overleftarrow{\rho}_{i+1}$ for all $i\,(1 \leq i < m)$, it is called a *time-stamped behavior* and denoted by $TBeh = (\rho_1, t_1)(\rho_2, t_2)...(\rho_m, t_m)$.

The set of behaviors $L_{\mathcal{V}}$ of a real-time automaton $\mathcal{V}$ is a regular language over the alphabet T. Let $LF = \sum_{i=1}^{n} k_i \cdot \sum P_i$. For a $Seq = \rho_1\rho_2 \dots \rho_m$ of $\mathcal{V}$, we define $Seq(LF) = \sum_{i=1}^{n} k_i \cdot Seq(\sum P_i)$ where $Seq\left(\sum P_i\right) = \sum_{j=1}^{m} \left\{ \begin{array}{ll} 1 & \overleftarrow{\rho}_j = P_i \\ 0 & \text{otherwise} \end{array} \right\}$.

For a time-stamped sequence $TSeq = (\rho_1, t_1)(\rho_2, t_2)...(\rho_m, t_m)$ of $\mathcal{V}$, we define $TSeq(LF) = Seq(LF)$ where $Seq = \rho_1\rho_2 \dots \rho_m$, and $TSeq(\ell) = \sum_{i=1}^{m} t_i$.

Definition 2.3 (Satisfaction of Linear Occurrence Invariants) Let Θ be an LOI of the form $c_{min} \le \ell \le c_{max} \Rightarrow \sum_{i=1}^{n} k_i \cdot \sum P_i \le M$.

- Θ is satisfied by a time-stamped sequence $TSeq$ iff $c_{min} \le TSeq(\ell) \le c_{max}$ implies $Seq(LF) \le M$. Otherwise, we say that Θ is violated by $TSeq$.
- Θ is satisfied by a sequence Seq, denoted by $Seq \models \Theta$, iff it is satisfied by every time-stamped sequence obtained from Seq. Otherwise, we say that Θ is violated by Seq.
- Θ is satisfied by a language $L \subseteq T^*$, denoted by $L \models \Theta$, iff $Seq \models \Theta$ for every $Seq \in L$. Otherwise, we say that Θ is violated by L.
- Θ is satisfied by a real-time automaton $\mathcal{V}$ iff $L_{\mathcal{V}} \models \Theta$. Otherwise, we say that Θ is violated by $\mathcal{V}$.

In the rest of this section we describe an algorithm to decide $L_{\mathcal{V}} \models \Theta$ using integer programming. Given two languages L_1 and L_2 over T. L_1 and L_2 are equivalent with respect to Θ (or simply equivalent), denoted by $L_1 \equiv L_2$, iff $L_1 \models \Theta \Leftrightarrow L_2 \models \Theta$. The theorem which is similar to Lemma 2.4 below was formalized and proved in [4]. Lemma 2.4 can be proved in the same way and its proof is omitted.

Lemma 2.4 *For languages* $L_1, L_2 \subseteq T^*$,

- $(L_1L_2) \equiv (L_2L_1)$.
- $(L_1 \cup L_2)^* \equiv (L_1^* L_2^*)$.
- $(L_1(L_2)^*)^* \equiv (\{\epsilon\} \cup (L_1(L_1)^*(L_2)^*))$ *where* ϵ *is the empty sequence.*

In the following, we identify a regular expression with the language it denotes. Like in [4], we can transform the regular language $L_{\mathcal{V}}$ into an equivalent finite union of regular languages of the form $\rho_1 \dots \rho_m Seq_1^* \dots Seq_h^*$, using Lemma 2.4, the distribution law $(L_1 \cup L_2)L = (L_1L \cup L_2L)$ and the idempotent law $(L^*)^* = L^*$. The readers are referred to [4] or [8] for the transformation procedure. Thus, to decide $L_{\mathcal{V}} \models \Theta$, it's enough to develop a technique to decide whether a regular language of the form $\rho_1 \dots \rho_m Seq_1^* \dots Seq_h^*$ satisfies Θ.

Given a time-stamped sequence $TSeq = (\rho_1, t_1)...(\rho_m, t_m)$ of $\mathcal{V}$. For a LDI $D = c_{min} \le \ell \le c_{max} \Rightarrow \sum_{i=1}^{n} c_i \cdot \int P_i \le M$, the linear function $\sum_{i=1}^{n} c_i \cdot \int P_i$ does not change its value when the new tuples of the form $(\rho', 0)$ are concatenated to $TSeq$. Noticing this property, in [4] authors equivalently transformed regular language $\rho_1 \dots \rho_m Seq_1^* \dots Seq_h^*$ further into another regular language L, so called normal form, which is simpler than former and $L \models D$ is decidable using linear programming. For a LOI $\Theta = c_{min} \le \ell \le c_{max} \Rightarrow \sum_{i=1}^{n} k_i \cdot \sum P_i \le M$, the function $\sum_{i=1}^{n} k_i \cdot \sum P_i$ changes its value generally when the new tuples of the form $(\rho', 0)$ are concatenated to $TSeq$. Therefore, we cannot use the same technique in [4] for our case. However, $\sum_{i=1}^{n} k_i \cdot \sum P_i$ has the same value for all time-stamped sequences which are obtained from a sequence. Using this property, we can develop an algorithm to decide $L \models \Theta$ by solving integer programming problems, where L is a regular language of the form $\rho_1 \ldots \rho_m Seq_1^* \ldots Seq_h^*$. For a sequence $Seq = \rho_1 \ldots \rho_m$, we define the function $\ell_{Seq} : T \to \mathbb{R}$, where $T = \{(t_1, \ldots, t_m) \,|\, (\rho_1, t_1) \ldots (\rho_m, t_m) \in TSeq\}$, as $\ell_{Seq}(t_1, \ldots, t_m) = \sum_{i=1}^{m} t_i$. ℓ_{Seq} is a continuous function. We denote the minimal value of ℓ_{Seq} by ℓ_{Seq}^{min} and the maximal value by ℓ_{Seq}^{max}. The minimal value always exists, but the maximal value may not exist in some cases. $\ell_{Seq}^{max} < \infty$ denotes that the maximal value of ℓ_{Seq} exists and $\ell_{Seq}^{max} = \infty$ denotes that the maximal value of ℓ_{Seq} does not exist.

Theorem 2.5 *The problem $L \models \Theta$ is decidable using integer programming, where $L = \rho_1 \ldots \rho_m Seq_1^* \ldots Seq_h^*$ and $\Theta = c_{min} \le \ell \le c_{max} \Rightarrow \sum_{i=1}^{n} k_i \cdot \sum P_i \le M$.*

Proof. Let $a_i = Seq_i(LF)$ $(1 \le i \le h)$ and $b_j = \rho_j(LF)$ $(1 \le j \le m)$. We first prove theorem in case that $c_{max} < \infty$, $\ell_{Seq_i}^{max} < \infty$ for all $i(1 \le i \le h)$ and $\ell_{\rho_j}^{max} < \infty$ for all $j(1 \le j \le m)$. Consider the following integer programming problem:

$$
\begin{aligned}
&k_1 \ge 0, \ldots, k_h \ge 0.\\
&\ell_{Seq_1}^{min} \times k_1 + \ldots + \ell_{Seq_h}^{min} \times k_h + \ell_{\rho_1}^{min} + \ldots + \ell_{\rho_m}^{min} \le c_{max}.\\
&\ell_{Seq_1}^{max} \times k_1 + \ldots + \ell_{Seq_h}^{max} \times k_h + \ell_{\rho_1}^{max} + \ldots + \ell_{\rho_m}^{max} \ge c_{min}.\\
&a_1 k_1 + \ldots + a_h k_h + b_1 + \ldots + b_m \;\to\; max.
\end{aligned}
$$

It is obvious that $L \models \Theta$ if the maximal value of the objective function is less than or equal to M. We prove that $L \not\models \Theta$ if the maximal value of the objective function is greater than M. From the assumption, there exist nonnegative integers $k_1', \ldots, k_h'$ satisfying $\ell_{Seq'}^{min} \le c_{max}$, $\ell_{Seq'}^{max} \ge c_{min}$ and $Seq'(LF) > M$ for $Seq' = \rho_1 \ldots \rho_m Seq_1^{k_1'} \ldots Seq_h^{k_h'}$. Here, $\ell_{Seq'}^{min} = \ell_{Seq_1}^{min} \times k_1' + \ldots + \ell_{Seq_h}^{min} \times k_h' + \ell_{\rho_1}^{min} + \ldots + \ell_{\rho_m}^{min}$, $\ell_{Seq'}^{max} = \ell_{Seq_1}^{max} \times k_1' + \ldots + \ell_{Seq_h}^{max} \times k_h' + \ell_{\rho_1}^{max} + \ldots + \ell_{\rho_m}^{max}$ and $Seq'(LF) = a_1 k_1' + \ldots + a_h k_h' + b_1 + \ldots + b_m$. Therefore, $[\, c_{min},\ c_{max}\,] \bigcap [\, \ell_{Seq'}^{min},\ \ell_{Seq'}^{max}\,] \ne \emptyset$ and there exists a nonnegative real number c satisfying $c_{min} \le c \le c_{max}$ and $\ell_{Seq'}^{min} \le c \le \ell_{Seq'}^{max}$. From the continuity of the function $\ell_{Seq'}$ there exists a time-stamped sequence $T'Seq'$ satisfying $T'Seq'(\ell) = c$. This means that for $T'Seq'$, $c_{min} \le T'Seq'(\ell) \le c_{max}$ but $T'Seq'(LF) > M$. That is, $L \not\models \Theta$. For the proof of the other cases, we introduce the following convention.

$$0 \cdot \infty = 0, \quad n \cdot \infty = \infty, \quad n + \infty = \infty, \quad n \le \infty \quad \text{for all n.}$$

Using this convention, the general case is proved in the same way as above, but the integer programming problem for the general case in which there is an occurrence of ∞ can generate several integer programming problems with no occurrences of ∞. For example, in case that $c_{max} = \infty$, $\ell_{Seq_1}^{min} \times k_1 + \ldots + \ell_{Seq_h}^{min} \times k_h + \ell_{\rho_1}^{min} + \ldots + \ell_{\rho_m}^{min} \le c_{max}$ is true for all $k_1 \ge 0, \ldots, k_h \ge 0$. Thus, we can decide $L \models \Theta$ by solving the following integer programming problem

$$k_1 \geq 0, \ldots, k_h \geq 0.$$
$$\ell^{max}_{Seq_1} \times k_1 + \ldots + \ell^{max}_{Seq_h} \times k_h + \ell^{max}_{\rho_1} + \ldots + \ell^{max}_{\rho_m} \geq c_{min}.$$
$$a_1 k_1 + \ldots + a_h k_h + b_1 + \ldots + b_m \rightarrow max.$$

□

3 Deterministic Probabilistic Real-Time Automata

In this section, we consider a subclass of probabilistic real-time automata, named *deterministic probabilistic real-time automata* in this paper, where each automaton has nondeterministic choice only for times. The probabilistic timed structures are used as the underlying model of deterministic probabilistic real-time automata. A discrete probability distribution over a set X is a mapping $p : X \rightarrow [0,1]$ such that the set $\{x \mid x \in X \text{ and } p(x) > 0\}$ is finite and $\sum_{x \in X} p(x) = 1$. $Dist(\mathrm{X})$ denotes the set of discrete probability distributions over X.

Definition 3.1 A deterministic probabilistic real-time automaton $\mathcal{Q}$ is a tuple $(Q, prob, L)$ consisting of

- a finite set Q of states,
- a function $prob : Q \rightarrow \mathcal{I} \times Dist(Q)$ assigning to each state $q \in Q$ a pair of the form $([a,b], p)$, where $[a,b] \in \mathcal{I}$ and $p \in Dist(Q)$,
- a function $L : Q \rightarrow 2^{AP}$ assigning to each state $q \in Q$ the set of atomic propositions that are true in that state.

Example 3.2 The Bounded Retransmission Protocol (BRP) is an extended version of the Alternating Bit Protocol (ABP) retransmitting corrupted messages. When the sender of ABP sends a message, it sends the message repeatedly until it receives an acknowledgement indicating successful delivery. When that happens, it starts transmitting the next message. There is no constraint on the number of retransmission of a message. Unlike ABP, BRP allows bounded number of retransmission of a message. Fig.1 shows a deterministic probabilistic real-time automaton model for the sender of BRP, having the maximal number of retransmission 2.

The system starts in state q_0 and waits for a message delivery request from environment. If a request is received, the system moves to state q_1 and delivers message immediately. After delivering message, there are two probabilistic choices in state q_1. The first choice is that with probability 0.9, the acknowledgement arrives from receiver between one and two time units, and the system moves to state q_2. The second choice is that with probability 0.1, the system fails to receive acknowledgement and moves to state q_3. In state q_3, the system delivers message again and moves to the next state in the same way of q_1. If a message delivery is successful, the system moves to state q_0 and sends the next message. In state q_4, the system delays one time unit for the proper reaction of receiver to the failure, and moves to state q_0. In every transition, the system clock is reset to zero.

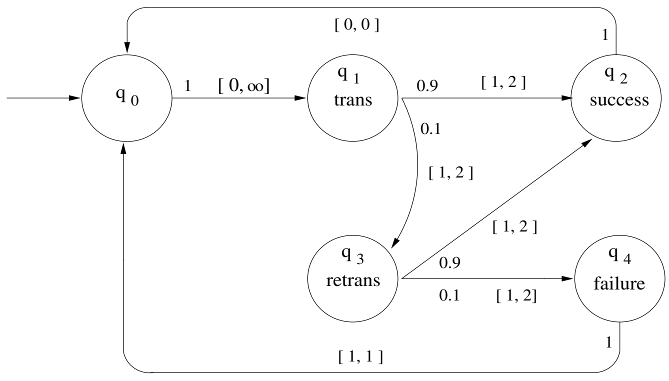

Fig. 1. Sender of Bounded Retransmission Protocol

Definition 3.3 A probabilistic timed structure is a tuple $\mathcal{M} = (Q, Step, L)$ consisting of

- a set Q of states,
- a function $Step : Q \to 2^{\mathbb{R}\times Dist(Q)}$ assigning to each state $q \in Q$ a set $Step(q)$ of pairs of the form (t, p), where $t \in \mathbb{R}$ and $p \in Dist(Q)$,
- a function $L : Q \to 2^{AP}$ assigning to each state $q \in Q$ the set of atomic propositions that are true in that state.

A path of $\mathcal{M}$ is a nonempty finite or infinite sequence of the form $\omega = q_0 \xrightarrow{t_0,p_0} q_1 \xrightarrow{t_1,p_1} q_2 \xrightarrow{t_2,p_2} q_3 \xrightarrow{t_3,p_3} \ldots$ where $q_i \in Q$, $(t_i, p_i) \in Step(q_i)$, and $p_i(q_{i+1}) > 0$. We use the following notations for a path ω. The first state of ω is denoted by $first(\omega)$, and if ω is finite then the last state of ω is denoted by $last(\omega)$. $|\omega|$ denotes the length of ω and is defined as the number of transition occurrences in ω, which is ∞ if ω is infinite. For $k \leq |\omega|$, $\omega(k)$ denotes the kth state of ω, and $step(\omega, k)$ denotes the label of the kth transition in ω. $\omega^{(i)}$ denotes the ith prefix of ω and $\omega\omega'$ denotes the concatenation of two paths ω and ω' when $last(\omega) = first(\omega')$. A position of ω is a pair (i, t), where $i \in \mathbb{N}$ and $t \in \mathbb{R}$ such that $t = 0$ if $t_i = 0$, otherwise $0 \leqslant t < t_i$. Here and below, $\mathbb{N}$ is the set of nonnegative integer numbers. $Pos(\omega)$ denotes the set of positions of ω. The state at position (i, t) is denoted by $state_\omega(i, t)$. For a path ω, we define $\mathcal{D}_\omega(i, t)$, the elapsed time until the position (i, t), as $\mathcal{D}_\omega(i,t) = \begin{cases} \mathcal{D}_\omega(i) & t = 0 \\ \mathcal{D}_\omega(i) + t & t \neq 0. \end{cases}$ where $\mathcal{D}_\omega(i) = \begin{cases} 0 & i = 0 \\ \mathcal{D}_\omega(i) = \sum_{j=0}^{i-1} t_j & i \neq 0 \end{cases}$ is the elapsed time until the ith transition. From the definition of $\mathcal{D}_\omega(i, t)$ it is possible that the two different positions have the same elapsed time until that positions. This occurs when a system passes through a number of states within zero time.

$Path_{fin}$ denotes the set of finite paths of $\mathcal{M}$ and $Path_{inf}$ denotes the set of infinite paths of $\mathcal{M}$. $Path_{fin}(q)$ denotes the set of finite paths starting from state q and $Path_{inf}(q)$ denotes the set of infinite paths starting from state q. Adversaries of a probabilistic timed structure resolve all the nondeterministic choices of the model.

Definition 3.4 An adversary of a probabilistic timed structure $\mathcal{M} = (Q, Step, L)$

is a function A mapping every finite path ω of $\mathcal{M}$ to a pair (t,p) such that $A(\omega) \in Step(last(\omega))$.

The set of adversaries is denoted by $\mathcal{A}$. For an adversary A, we define

$$Path^A_{fin} = \{\omega \in Path_{fin} \mid A(\omega^{(i)}) = step(\omega, i) \text{ for } 0 \le i < |\omega|\},$$
$$Path^A_{inf} = \{\omega \in Path_{inf} \mid A(\omega^{(i)}) = step(\omega, i) \text{ for } 0 \le i\}.$$

Let $Path^A_{fin}(q) = Path^A_{fin} \cap Path_{fin}(q)$ and $Path^A_{inf}(q) = Path^A_{inf} \cap Path_{inf}(q)$. For each state $q \in Q$, a probability measure $Prob^A_q$ over $Path^A_{inf}(q)$ is defined in the following way. A sequential Markov chain $MC^A = (Path^A_{fin}, \mathbf{P}^A)$ is associated with an adversary A, where $\mathbf{P}^A(\omega, \omega') = \begin{cases} p(q) & \text{if } A(\omega) = (t,p) \text{ and } \omega' = \omega \xrightarrow{t,p} q, \\ 0 & \text{otherwise.} \end{cases}$

Let $\mathcal{F}^A_{Path}(q)$ be the smallest σ-algebra on $Path^A_{inf}(q)$ which for all $\omega' \in Path^A_{fin}(q)$ contains the sets $\{\omega \mid \omega \in Path^A_{inf}(q) \text{ and } \omega' \text{ is a prefix of } \omega\}$. Let $Prob^A_{fin} : Path^A_{fin}(q) \to [0,1]$ be the mapping defined inductively on the length of paths in $Path^A_{fin}(q)$ as follows. If $|\omega| = 0$ then $Prob^A_{fin}(\omega) = 1$. If $\omega' = \omega \xrightarrow{t,p} q$ for some $\omega \in Path^A_{fin}(q)$, then we let $Prob^A_{fin}(\omega') = Prob^A_{fin}(\omega)\mathbf{P}^A(\omega, \omega')$. The probability measure $Prob^A_q$ on $\mathcal{F}^A_{Path}(q)$ is the unique measure such that $Prob^A_q(\{\omega \mid \omega \in Path^A_{inf}(q) \text{ and } \omega' \text{ is a prefix of } \omega\}) = Prob^A_{fin}(\omega')$. In this paper, we only consider divergent adversaries. That is, for any infinite paths under our consideration the number of state changes occurring at finite time intervals are always finite.

Definition 3.5 Underlying model of a deterministic probabilistic real-time automaton $\mathcal{Q} = (Q, prob, L)$ is the probabilistic timed structure $\mathcal{M}_{\mathcal{Q}} = (Q, Step, L)$ in which $Step(q) = \{(t,p) \mid t \in [a,b] \text{ and } ([a,b],p) \in prob(q)\}$

4 Probabilistic Duration Calculus of Weakly Monotonic Time

In this section, we conservatively extend WDC to a logic that allows to specify dependability properties for real-time systems, such as the constraints for the probability of satisfaction of a WDC formula by the set of adversaries of system model. We call $\mathbb{N} \times \mathbb{R}$ the macro-micro time plane, and each $(k,t) \in \mathbb{N} \times \mathbb{R}$ a macro-micro time point. θ is used to denote the original point of this plane, i.e., $\theta = (0,0)$, and τ is used to range over $\mathbb{N} \times \mathbb{R}$. A partial order $\le$ on $\mathbb{N} \times \mathbb{R}$ is defined as $\tau_1 \le \tau_2$ iff $k_1 \le k_2$ and $t_1 \le t_2$ where $\tau_1 = (k_1, t_1)$ and $\tau_2 = (k_2, t_2)$. We define weakly monotonic time frames on $\mathbb{N} \times \mathbb{R}$ in the following way.

Definition 4.1 A weakly monotonic time frame WT on $\mathbb{N} \times \mathbb{R}$ is a subset of $\mathbb{N} \times \mathbb{R}$ satisfying the following conditions:

- WT is a linearly ordered subset of $\mathbb{N} \times \mathbb{R}$ with respect to $\le$.
- $\pi_1(WT) = \mathbb{N}$ or $\pi_2(WT) = \mathbb{R}$ where $\pi_1(WT) = \{k \mid (k,t) \in WT\}$ and $\pi_2(WT) =$

$\{t \mid (k,t) \in WT\}$.

- If $k \in \pi_1(WT)$ and $k' < k$, then $k' \in \pi_1(WT)$. Similarly, if $t \in \pi_2(WT)$ and $t' < t$, then $t' \in \pi_2(WT)$.
- If $t_1 < t_2$, $(k_1, t_1) \in WT$ and $(k_2, t_2) \in WT$, then $k_1 \leq k_2$.

For each infinite path ω of a probabilistic timed structure $\mathcal{M}$, the set $WT_\omega = \{(k,t) \mid (i,t') \in Pos(\omega),\ k = i \text{ and } t = \mathcal{D}_\omega(i,t')\}$ is a weakly monotonic time frame. Given an infinite path ω and an atomic proposition $P \in AP$. We define a $\{0,1\}$-valued function $P_\omega : WT_\omega \to \{0,1\}$ as

$$P_\omega(k,t) = \begin{cases} 1 & state_\omega(k,t') = q \text{ and } P \in L(q) \\ 0 & \text{otherwise,} \end{cases}$$

where t' is such that $t = \mathcal{D}_\omega(i,t')$. We also define a function $P^1_\omega : \pi_1(WT_\omega) \to \{0,1\}$ as $P^1_\omega(k) = P_\omega(k,0)$ and a partial function $P^2_\omega : \pi_2(WT_\omega) \to \{0,1\}$ as

$$P^2_\omega(t) = \begin{cases} P_\omega(k,t) & \{k \mid (k,t) \in WT_\omega\} \text{ is singleton} \\ \bot & \text{otherwise.} \end{cases}$$

For a macro-micro time point $\tau = (k',t')$, let $R_\tau = \{(k,t) \mid 0 \leqslant k \leqslant k' \text{ and } 0 \leqslant t \leqslant t'\}$. We define the restriction of a weakly monotonic time frame WT_ω to R_τ as $WT_\omega \downharpoonleft R_\tau = WT_\omega \bigcap R_\tau$. $WT_\omega \downharpoonleft R_\tau$ is a linear order subset of WT_ω and has the maximal element which is denoted by τ_ω.

Definition 4.2 The syntax of PWDC is defined as:

$$\Phi ::= \forall[\Psi]_{op\ \lambda} \mid \exists[\Psi]_{op\ \lambda} \mid \neg\Phi \mid \Phi \wedge \Phi,$$

$$\Psi ::= \lceil P \rceil^0 \mid \lceil P \rceil \mid F\ op\ c \mid \neg\Psi \mid \Psi \wedge \Psi \mid \Psi^\frown\Psi,$$

$$F ::= \eta \mid \textstyle\sum_{i=1}^n k_i \cdot \sum P_i \mid \ell \mid \sum_{i=1}^n c_i \cdot \int P_i,$$

where $op \in \{=, \leq, \geq\}$, $\lambda \in [0,1]$, $k_i \in \mathbb{Z}$ and $c_i \in \mathbb{R}$. Here, c takes integer number when F is η or $\sum_{i=1}^n k_i \cdot \sum P_i$, and real number otherwise.

Φ is called a PWDC formula, Ψ is called a WDC formula, and F is called a measurement term. The reason of restriction to linear terms is for simplicity. The set of intervals over WT_ω is defined as $Intv(WT_\omega) = \{[\tau_1, \tau_2] \in WT_\omega \times WT_\omega \mid \tau_1 \leq \tau_2\}$. η, ΣP, ℓ, $\int P$ are called atomic measurement terms. The interpretation of an atomic measurement term on $Intv(WT_\omega)$ is defined as $I^\omega_\eta([\tau_1,\tau_2]) = \pi_1(\tau_2) - \pi_1(\tau_1)$, $I^\omega_{\sum P}([\tau_1,\tau_2]) = \sum_{i=\pi_1(\tau_1)}^{\pi_1(\tau_2)} P^1_\omega(i)$, $I^\omega_\ell([\tau_1,\tau_2]) = \pi_2(\tau_2) - \pi_2(\tau_1)$ and $I^\omega_{\int P}([\tau_1,\tau_2]) = \int_{\pi_2(\tau_1)}^{\pi_2(\tau_2)} P^2_\omega dt$. The interpretation $I^\omega_F([\tau_1,\tau_2])$ of non-atomic measurement terms F on $Intv(WT_\omega)$ is defined in the standard way and omitted here. Given a probabilistic timed structure $\mathcal{M}$ and a WDC formula Ψ. Let q be a state of $\mathcal{M}$ and $[\tau_1,\tau_2]$ be a weakly monotonic time interval $[\tau_1,\tau_2]$ of $\omega \in Path_{inf}(q)$.

Definition 4.3 (Semantics of WDC Formulas)

The satisfaction relation $(q, \omega, [\tau_1, \tau_2]) \models \Psi$ is defined inductively as follows:

$$(q, \omega, [\tau_1, \tau_2]) \models \lceil P \rceil^0 \text{ iff } \tau_1 = \tau_2 \text{ and } P_\omega(\tau_1) = 1$$

$$(q, \omega, [\tau_1, \tau_2]) \models \lceil P \rceil \text{ iff } \tau_1 < \tau_2 \text{ and } P_\omega(\tau) = 1 \text{ for all } \tau : \tau_1 < \tau < \tau_2$$

$$(q, \omega, [\tau_1, \tau_2]) \models F \; op \; c \text{ iff } I_F^\omega([\tau_1, \tau_2]) \text{ op } c$$

For an infinite path ω and a nonnegative real number t, let $\tau_t = (k, t)$ where $k = \min\{ k' \mid (k', t) \in WT_\omega \}$. For a nonnegative integer k, let $\tau_k = (k, 0)$. We define the semantics of PWDC formulas in three different ways. Given a probabilistic timed structure $\mathcal{M}$, a state q of $\mathcal{M}$, and a PWDC formula Φ.

Definition 4.4 (Macro-micro/Macro/Micro Time Semantics of PWDC Formulas) Let τ $((t, k))$ be a macro-micro (macro or micro respectively) time point. The satisfaction relation $(\mathcal{A}, q, \tau(t, k)) \models \Phi$ is defined inductively as follows:

$$(\mathcal{A}, q, \tau(t, k)) \models \forall[\Psi]_{op\,\lambda} \text{ iff } Prob_q^A(\{\omega \mid \omega \in Path_{inf}^A(q) \text{ and }$$
$$(q, \omega, [\theta, \tau_\omega(t, k)]) \models \Psi\}) \, op \; \lambda \text{ for all } A \in \mathcal{A}$$

$$(\mathcal{A}, q, \tau(t, k)) \models \exists[\Psi]_{op\,\lambda} \text{ iff } Prob_q^A(\{\omega \mid \omega \in Path_{inf}^A(q) \text{ and }$$
$$(q, \omega, [\theta, \tau_\omega(t, k)]) \models \Psi\}) \, op \; \lambda \text{ for some } A \in \mathcal{A}$$

Macro time semantics and Micro time semantics are natural adaptations to probabilistic domain of the ways to define semantics in the original DC and its variant logics. But, macro-micro time semantics is a combination of macro time semantics and micro time semantics. The problem $(\mathcal{A}, q, t) \models \forall[\Psi]_{op\,\lambda}$ can be decided by deciding $(\mathcal{A}, q, (k', t)) \models \forall[\Psi]_{op\,\lambda}$ for some k' and the problem $(\mathcal{A}, q, k) \models \forall[\Psi]_{op\,\lambda}$ can be decided by deciding $(\mathcal{A}, q, (k, t')) \models \forall[\Psi]_{op\,\lambda}$ for some t'. For this reason, we concentrate on the development of model checking algorithms relating to Macro-micro time semantics.

5 Checking Deterministic Probabilistic Real-Time Automata for PWDC formulas

In this section, we consider the problem to check deterministic probabilistic real-time automata for some subclass of PWDC formulas. We give two algorithms. The first algorithm is to decide $(\mathcal{A}, q, \tau) \models \forall[\Theta]_{op\,\lambda}$ using linear programming, where Θ is a linear occurrence invariant, and the second algorithm is to decide $(\mathcal{A}, q, \tau) \models \forall[\Box\Theta]_{op\,\lambda}$ for all τ by solving the system of linear equations, where Θ is a linear occurrence invariant of the form $c_{min} \leq \ell \leq c_{max} \Rightarrow \sum P = 0$.

Let $\omega = q_0 \xrightarrow{t_0} q_1 \xrightarrow{t_1} q_2 \xrightarrow{t_2} \ldots$ be a path of a deterministic probabilistic real-time automaton. Here, we dropped the scripts denoting probability values from the path for simplicity. Let $t = t_0 + t_1 + t_2'$ where $t_2' < t_2$ and $\tau = (2, t)$. Then for any linear occurrence invariant Θ, $(q_0, \omega, [\theta, \tau]) \models \Theta$ if and only if $(\rho_0, t_0)(\rho_1, t_1)(\rho_2, t_2') \models \Theta$. For this reason, we consider paths as time-stamped behaviors in this section for the development of checking algorithm.

Now we describe the first algorithm. We explain the main ideas of our algorithm using an example and formalize it later. Let $\mathcal{Q} = (Q, prob, L)$ be the deterministic probabilistic real-time automaton given in Fig.1, q_0 be the starting state of $\mathcal{Q}$, $\tau = (7, 9)$, and $\Theta \mathrel{\widehat{=}} 0 \leq \ell \Rightarrow \sum failure \leq 0$. The problem $(\mathcal{A}, q_0, \tau) \models \forall [\Theta]_{\geq 0.9}$ is decided using linear programming as follows. Let $T = \{\rho_{01}, \rho_{12}, \rho_{13}, \rho_{20}, \rho_{32}, \rho_{34}, \rho_{40}\}$ where $\rho_{01} = (q_0, [0, \infty), q_1)$, $\rho_{12} = (q_1, [1, 2], q_2)$, $\rho_{13} = (q_1, [1, 2], q_3)$, $\rho_{20} = (q_2, [0, 0], q_0)$, $\rho_{32} = (q_3, [1, 2], q_2)$, $\rho_{34} = (q_3, [1, 2], q_4)$ and $\rho_{40} = (q_4, [1, 1], q_0)$. Then $\mathcal{V} = (Q, T, L)$ becomes a real-time automaton. We designate q_0 as the starting state of $\mathcal{V}$. For a sequence $Seq = \rho_{i_1 j_1} \rho_{i_2 j_2} \ldots \rho_{i_m j_m}$ of $\mathcal{V}$, we define $P(Seq) = p_1(q_{j_1}) \times p_2(q_{j_2}) \times \ldots \times p_m(q_{j_m})$ where p_k $(k = 1, \ldots, m)$ satisfies $prob(q_{i_k}) = ([a_k, b_k], p_k)$ in $\mathcal{Q}$. From Fig.1, we can easily see that

$$L_{\mathcal{V}} = R^* \cup (R^* \cdot \rho_{01}) \cup (R^* \cdot \rho_{01}\rho_{12}) \cup (R^* \cdot \rho_{01}\rho_{13}) \cup (R^* \cdot \rho_{01}\rho_{13}\rho_{32}) \cup (R^* \cdot \rho_{01}\rho_{13}\rho_{34}),$$

where $R = R_1 \cup R_2 \cup R_3$, $R_1 = \rho_{01}\rho_{12}\rho_{20}$, $R_2 = \rho_{01}\rho_{13}\rho_{32}\rho_{20}$ and $R_3 = \rho_{01}\rho_{13}\rho_{34}\rho_{40}$.

From $L_{\mathcal{V}}$, we can pick out sequences having length smaller than or equal to $7 (= \pi_2(\tau))$ and not satisfying $\sum failure \leq 0$ by solving linear equations. Let us consider R^*. The linear equation $3k_1 + 4k_2 + 4k_3 = 7$ on the nonnegative integer numbers has two solutions $(1, 1, 0)$ and $(1, 0, 1)$, where 3 is the length of R_1 and 4 is the length of R_2 and R_3. The solution $(1, 1, 0)$ means that R_1R_2 and R_2R_1 are the sequences of length 7 in R^*. The solution $(1, 0, 1)$ means that R_1R_3 and R_3R_1 are another sequences of length 7 in R^*. For the sequence R_1R_3, the prefix $(R_1R_3)^{(7)}$ do not satisfy $\sum failure \leq 0$. Also for the sequence R_3R_1, the prefixes $(R_3R_1)^{(4)}$, $(R_3R_1)^{(5)}$, $(R_3R_1)^{(6)}$ and $(R_3R_1)^{(7)}$ do not satisfy $\sum failure \leq 0$. We denote these prefixes respectively by $E_1 = \{(R_1R_3)^{(7)}\}$ and $E_2 = \{(R_3R_1)^{(4)}, (R_3R_1)^{(5)}, (R_3R_1)^{(6)}, (R_3R_1)^{(7)}\}$. Applying the same procedure to $R^* \cdot \rho_{01}$, $R^* \cdot \rho_{01}\rho_{12}$, $R^* \cdot \rho_{01}\rho_{13}$, $R^* \cdot \rho_{01}\rho_{13}\rho_{32}$, $R^* \cdot \rho_{01}\rho_{13}\rho_{34}$, we can pick out two more sets $E_3 = \{(R_3 \cdot \rho_{01}\rho_{13}\rho_{32})^{(4)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{32})^{(5)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{32})^{(6)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{32})^{(7)}\}$ and $E_4 = \{(R_3 \cdot \rho_{01}\rho_{13}\rho_{34})^{(4)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{34})^{(5)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{34})^{(6)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{34})^{(7)}\}$ from $R^* \cdot \rho_{01}\rho_{13}\rho_{32}$ and $R^* \cdot \rho_{01}\rho_{13}\rho_{34}$ respectively, in which every sequence does not satisfy $\sum failure \leq 0$.

We make tuples by taking at most one element from each E_i $(i = 1, 2, 3, 4)$ without considering order. For example, the tuple $\Sigma_{min} = ((R_1R_3)^{(7)}, (R_3R_1)^{(4)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{32})^{(4)}, (R_3 \cdot \rho_{01}\rho_{13}\rho_{34})^{(4)})$ is a tuple consisting of first element of each E_i. The remaining procedure is to generate linear constraints over the nonnegative real numbers for each tuple and do probability calculation if it is feasible. We use an example to demonstrate the procedure. The following is the linear constraints over the nonnegative real numbers generated from Σ_{min} and $\pi_2(\tau) = 9$.

$$\begin{cases} t_{01}^1 + t_{12}^2 + t_{20}^3 + t_{01}^4 + t_{13}^5 + t_{34}^6 + t_{40}^7 = 9, \\ 0 \le t_{01}^1,\ 1 \le t_{12}^2 \le 2,\ 0 \le t_{20}^3 \le 0,\ 0 \le t_{01}^4, \\ 1 \le t_{13}^5 \le 2,\ 1 \le t_{34}^6 \le 2,\ 0 \le t_{40}^7 \le 1, \\ t_{01}^1 + t_{13}^2 + t_{34}^3 + t_{40}^4 = 9, \\ 1 \le t_{13}^2 \le 2,\ 1 \le t_{34}^3 \le 2,\ 0 \le t_{40}^4 \le 1. \end{cases}$$

The first line is generated from the first sequence of Σ_{min} by changing ρ_{ij} to t_{ij}^k where k denotes the position of ρ_{ij}, and changing concatenation operation to plus operation. The second line and third line are time constraints for the transitions occurring in the first element of Σ_{min}, given in the definition of $\mathcal{V}$. The fourth line and fifth line are generated from the second sequence of Σ_{min} in the same way. The third sequence and fourth sequence of Σ_{min} are equal with the second sequence of Σ_{min} and we don't consider it.

Using linear programming, we can decide that the linear constraints above is feasible. We calculate $P(\Sigma_{min}) = 1 - (P(\rho_{01}\rho_{12}\rho_{20}\rho_{01}\rho_{13}\rho_{34}\rho_{40}) + P(\rho_{01}\rho_{13}\rho_{34}\rho_{40})) = 1-((1\times 0.9\times 1\times 1\times 0.1\times 0.1\times 1)+(1\times 0.1\times 0.1\times 1)) = 1-(0.009+0.01) = 0.981$. From the definition of satisfaction for PWDC formulas, the above procedure applied to Σ_{min} and the resulting value 0.981 mean that for some adversary A of $\mathcal{M}_{\mathcal{Q}}$, $Prob_{q_0}^A(\{\omega \mid \omega \in Path_{inf}^A(q_0) \text{ and } (q, \omega, [\theta, \tau_\omega]) \models \Theta\}) = 0.981$. We apply the above procedure to every tuple Σ and calculate $P(\Sigma)$ if the generated linear constraints from Σ is feasible. The minimum of these values is not less than 0.9 and we can conclude $(\mathcal{A}, q_0, \tau) \models \forall\,[\Theta]_{\ge\, 0.9}$.

Remark 5.1 In fact, we can directly conclude $(\mathcal{A}, q_0, \tau) \models \forall\,[\Theta]_{\ge\, 0.9}$ only with value $P(\Sigma_{min}) = 0.981$. This is because the value 0.981 which is calculated from the tuple Σ_{min} consisting of first element of each E_i is the minimum of the values calculated from each feasible tuple, i.e., the tuple generating feasible linear constraints. In this paper, we don't consider technical details relating to the complexity of algorithm.

Given a deterministic probabilistic real-time automaton $\mathcal{Q} = (Q, prob, L)$ and a state q. We define $T = \{\, (q', [a,b], q'') \mid q' \in Q,\ q'' \in Q,\ prob(q') = ([a,b], p),\ p(q'') > 0\,\}$. Then, (Q, T, L) becomes a real-time automaton. We designate q as starting state of (Q, T, L) and denote this real-time automaton by $\mathcal{Q}_q$. $L_{\mathcal{Q}_q}$ denotes the set of behaviors of $\mathcal{Q}_q$ and $H(L_{\mathcal{Q}_q})$ denotes the star-height of $L_{\mathcal{Q}_q}$.

Theorem 5.2 *Let us assume that $H(L_{\mathcal{Q}_q}) \le 1$. The problem $(\mathcal{A}, q, \tau) \models \forall[\Theta]_{op\ \lambda}$ is decidable using linear programming, where Θ is a linear occurrence invariants.*

The details of the proof is in [14]. Now we describe second algorithm. Given a deterministic real-time automaton $\mathcal{Q}$ and its state q.

Theorem 5.3 *The problem $(\mathcal{A}, q, \tau) \models \forall[\Box\Theta]_{op\ \lambda}$ for all τ, where $\Theta = (c_{min} \le \ell \le$*

$c_{max} \Rightarrow \sum P = 0)$, *is decidable by solving the system of linear equations.*

Proof. Note that from the assumption for Θ, it simply says that the probability p that P never occurs in a run satisfies $op\lambda$. For each adversary A of $\mathcal{M}_{\mathcal{Q}}$ we define

$$Path_{\neg P}(q) = \{\omega \mid \omega \in Path^A_{inf}(q) \text{ and } L(\omega(k)) \not\ni P \text{ for all } k\},$$
$$Pr_{\neg P}(q) = \{Prob^A_q(Path_{\neg P}(q))\}.$$

$Pr_{\neg P}(q)$ has the same value for all adversaries because of the determinism for discrete transitions, and $(\mathcal{A}, q, \tau) \models \forall[\Box\Theta]_{op\ \lambda}$ for all τ if and only if $Pr_{\neg P}(q)\ op\ \lambda$. Note that the premise $c_{min} \leq \ell \leq c_{max}$ of Θ and τ need not be considered in our case. Thus, it's enough to develop a technique to calculate $Pr_{\neg P}(q)$. Let $q_1, q_2, \ldots, q_m$ be the states of $\mathcal{Q}$ satisfying $L(q_i) \not\ni P$ and $p(q_i) > 0$ for all $i\,(1 \leq i \leq m)$. Here, p is the probability distribution satisfying $prob(q) = ([a, b], p)$. We have the following set constraint

$$Path_{\neg P}(q) = \textstyle\bigcup_{i=1}^m (q \xrightarrow{t_i,\, p} q_i) \cdot Path_{\neg P}(q_i)$$

relating to the states $q, q_1, q_2, \ldots, q_m$. From this set relation, we also have the following linear equation

$$Pr_{\neg P}(q) = \textstyle\sum_{i=1}^m p(q_i) \cdot Pr_{\neg P}(q_i).$$

Applying this procedure to all states of $\mathcal{Q}$, we have the system of linear equations. Solving this system of linear equations we can obtain the value of $Pr_{\neg P}(q)$. □

Example 5.4 Let us consider the sender of BRP and a PWDC formula $\forall[\Box\Theta]_{\geq\ 0.6}$, where $\Theta = (c_{min} \leq \ell \leq c_{max} \Rightarrow \sum failure = 0)$. Applying the procedure given in the proof of theorem, we have the following system of linear equations.

$$\begin{cases} Pr_{\neg failure}(q_0) = Pr_{\neg failure}(q_1), \\ Pr_{\neg failure}(q_1) = 0.9 \cdot Pr_{\neg failure}(q_2) + 0.1 \cdot Pr_{\neg failure}(q_3), \\ Pr_{\neg failure}(q_2) = Pr_{\neg failure}(q_0), \\ Pr_{\neg failure}(q_3) = 0.9 \cdot Pr_{\neg failure}(q_2). \end{cases}$$

Solving this system of linear equation, we have $Pr_{\neg failure}(q_0) = 0$. This means that the problem $(\mathcal{A}, q_0, \tau) \not\models \forall[\Theta]_{\geq\ 0.6}$ for some τ.

6 Conclusion and Future Work

We have studied a subclass of WDC (Duration Calculus of Weakly Monotonic Time) called LOI (Linear Occurrence Invariants) and and presented an algorithm to check real-time automata for LOI using integer programming techniques. We have also introduced PWDC (Probabilistic Duration Calculus of Weakly Monotonic Time) to specify dependability requirements of real-time system and presented some techniques to check deterministic probabilistic real-time automata for PWDC formulas. Though these algorithms work only for simple class of PWDC formulas, we believe that they can be impvoved for a large class of PWDC formulas, and this will be presented in our future work.

References

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

[2] R. Alur and D.L Dill. A Theory of Timed Automata. *Theoretical Computer Science*, pages 183-235, 1994.

[3] P.D'Argenio, J.-P.Katoen, T.Ruys, and J.Tretmans. Modeling and verifying a bounded retransmission protocol. *Proc. of COST 247 International Workshop on Applied Formal Methods in System Design, Maribor, Slovenia, Technical Report. University of Maribor, 1996*

[4] Zhou Chaochen, Zhang Jingzhong, Yang Lu and Li Xiaoshan. Linear Duration Invariants. Research Report 11, UNU-IIST, P.O.Box 3058, Macau, 1993. Published in: *Formal Techniques in Real-time and Fault-tolerant systems*, LNCS 863, 1994.

[5] Paritosh. K.Pandya, Shankara Narayanan Krishna and Kuntal Loya. On Sampling Abstraction of Continuous Time Logic with Durations. *Technical Report TIFR-PKP-GM-2006/1. Tata Institute of Fundamental Research, India*

[6] Dang Van Hung and Zhang Miaomiao. On Verification of Probabilistic Timed Automata against Probabilistic Duration Properties. *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications RTCSA 2007, Daegu, Korea, August 21-24, 2007.* IEEE Computer Society Press, Los Alamitos, California, pp. 165 – 172.

[7] Paritosh. K. Pandya and Dang Van Hung. Duration Calculus of Weakly Monotonic Time. *Proceedings of FTRTFT98, LNCS 1486, pp 55-64, Springer-Verlag, 1998.*

[8] Zhou Chaochen and Michael R Hansen. Duration Calculus: A Formal Approach to Real-Time Systems. Springer-Verlag, 2004.

[9] Marta Kwiatkowska, Gethin Norman, Roberto Segala and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101-150, 2002.

[10] Y. Kesten, A. Pnueli, J Sifakis and S. Yovine. Integration Graphics: A Class of Decidable Hybrid Systems. LNCS 736, pages 179-208, Springer-Verlag, 1994.

[11] Victor A. Braberman and Dang Van Hung. On Checking Timed Automata for Linear Duration Invariants. Technical Report 135, UNU/IIST, P.O.Box 3058, Macau, February 1998. Proceedings of the 19th Real-Time Systems Symposium RTSS'98, December 2-4, 1998, Madrid, Spain, IEEE Computer Society Press 1998, pp.264-273.

[12] Li Xuandong and Dang Van Hung. Checking Linear Duration Invariants by Linear Programming. Research Report 70, UNU-IIST, P.O.Box 3058, Macau, 1996. Published in LNCS 1179, Springer-Verlag, 1996, 321-332.

[13] Pham Hong Thai and Dang Van Hung. Verifying Linear Duration Constraints of Timed Automata. Technical Report 306, UNU-IIST, P.O.Box, 3058, Macau, 2004. Presented at and Published in the ICTAC'04, 2004.

[14] Choe Changil and Dang Van Hung. On Verification of Linear Occurrence Invariants of Real-Time Systems. Technical Report 375, UNU-IIST, P.O.Box, 3058, Macau, 2007.