Modelling History-Dependent Business Processes

Kees van Hee, Olivia Oanea*, Alexander Serebrenik,
Natalia Sidorova and Marc Voorhoeve

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. Choices in business processes are often based on the process history saved as a log-file listing events and their time stamps. In this paper we introduce a finite-path variant of the timed propositional logics with past for specifying guards in business process models. The novelty is due to the introduction of boundary points *start* and *now* corresponding to the starting and current observation points. Reasoning in presence of boundary points requires three-valued logics as one needs to distinguish between temporal formulas that hold, those that do not hold and "unknown" ones corresponding to "open cases". Finally, we extend a sub-language of the logics to take uncertainty into account.

1 Introduction

In this paper we consider case management systems, an important and generic class of Enterprise Information Systems. An important part of a case management system is a workflow management system (WfMS) that takes care of the distribution of work to agents, which can be either human or application software.

Routing decisions taken by a WfMS can depend on previous observations. For instance, a bank can propose more interesting loan conditions to those customers who paid off the previous loans on time. We call processes executed by such a WfMS *history-dependent processes*. Importance of history-based decisions in workflow management has been recognised in the past [10, 12]. In history-dependent processes actions can be guarded by conditions on the process history. In this paper we propose a temporal logic for specifying *history-based guards* in models of history-dependent processes that we call *LogLogics*.

At any given moment of time, history is a finite object. The inherent incompleteness of our observations should be taken into account. Consider for example a guard saying that every bill was paid within four weeks where we would like to obtain *true* in case every bill was indeed paid within four weeks, *false* if there is a bill that was not paid within four weeks and *unknown* in case there is a bill issued not later than four weeks ago which is not paid yet, while the bills issued more than four weeks ago are paid on time. In some cases the decision on the continuation of the process is made giving the benefit of the doubt, i.e. *unknown* leads to the same choice as *true* in other cases it leads

* Supported by the NWO Open Competitie project MoveBP, Project number 612.000.315

van Hee K., Oanea O., Serebrenik A., Sidorova N. and Voorhoeve M. (2006).
Modelling History-Dependent Business Processes.
In *Proceedings of the 4th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 76-85
DOI: 10.5220/0002499500760085

to the same choice as *false*, and in a number of cases evaluating a guard to *unknown* leads to enabling a special procedure to handle the case.

A number of three-valued (untimed) temporal logics, including L-TL, have been proposed by Nakamura [9] and investigated in [8]. Similarly to L-TL, if a *LogLogics*-formula is evaluated to *true* or *false* at a given time point, this value cannot be changed in the future (i.e., when *now* increases), while *unknown* can become *true* or *false*. Unlike L-TL, not every *LogLogics*-formula has to be eventually evaluated to *true* or *false*. Indeed, as we are going to see, there exist *LogLogics*-formulas ϕ such that $\Box\phi$ will always be evaluated to *unknown*.

Since history is a finite linear sequence of timed events, we consider linear timed temporal logics (defined on infinite sequences) that have been the subject of intensive research in the past, starting with [1–3, 6]. More recent works on the subject include [4, 13]. Due to the nature of history we need to consider not only future but also past temporal operators. Therefore, we have chosen to extend timed propositional logic with past (TPTL+Past) [1–3].

An alternative to TPTL+Past might have been the metric timed logic (MTL) [6, 13]. The reasons for opting for TPTL+Past rather than for MTL are twofold. First of all, TPTL is "more temporal": it uses real clocks to express timed constraints. This allows to express such common for EIS constraints as "event *p* occurred between January 1, 2005 and January 1, 2006". Unlike TPTL, MTL reasons in terms of distances between events. Hence, in order to express the same constraint we need to introduce a special event *q* that occurred on January 1, 2005 and require that *p* followed *q* within one year. Second, as recently shown in [4] TPTL+Past is strictly more expressive than MTL+Past.

Two different semantics for timed temporal logics can be considered: point-wise semantics, where formulas are evaluated over discrete sequences of timed events, and interval-based semantics, where formulas are evaluated over continuous time line [11]. We believe that discrete sequences of timed events are better suited for specifying history-based guards in business processes and we choose the point-wise semantics.

Although processes we consider are in general not probabilistic processes and history is not always a reliable source for forecasting the future behaviour (the client who always paid his bills on time can suddenly go bankrupt), statistical data on history are still widely used to make choices. Following the observations of [5], we define a number of guard patterns for our logic. For these patterns we extend our framework by introducing certainty values that allow us to check, for instance, whether issued bills paid on time in 95% of cases.

2 *LogLogics*

In this section we present *LogLogics*. *LogLogics* aims at the modelling of history-dependent processes based on log-files. Log-files usually record series of events such as "withdraw 100 euro" or "drug A has been administered". However, when making decisions, we sometimes would like to reason on states rather than on events, e.g. "the balance is negative" or "the temperature is higher than 39 degrees". Therefore, given an initial state and a series of events presented in a log, we extend the log by replacing events by pairs (e,s) where *e* is an event and *s* is a state after the occurrence of *e*. For

the initial state, the event is empty. For the sake of readability we refer to these pairs as *states*. The set of all states is denoted Σ.

LogLogics is an adaptation of the Timed Propositional Temporal Logic with past [1, 4] to finite sequences of events limited by two special time points: *start* and *now*. The *start* point refers to the beginning of observations rather than to the absolute begin. While such an absolute begin is well-suited for modelling the behaviour of software systems that have been invoked at some moment of time, it is less appropriate for business processes, where the observations could be available for a recent period of time only. Similarly, *now* is the last time point where observations are available.

Due to the finiteness of observations, the values of traditional temporal operators can become unknown. Consider, for instance, a predicate p that is *true* is a given apple is green. However, as some apples change colour with time, the fact that during the entire period of observations the apple stayed green does not necessary imply that "always green" is true. Nor, in fact does it imply that "always green" is false. In such a situation we would like to say that the value of "always green" is unknown. To formalise this intuition we make use of the three-valued logics with truth values *false*, *unknown* and *true* such that *false* $\prec$ *unknown* $\prec$ *true*, $\min(S)$ and $\max(S)$ are defined for a set S with respect to $\prec$, and

	$\neg$
false	*true*
unknown	*unknown*
true	*false*

$$(x \wedge y) \stackrel{\text{def}}{=} \min\{x,y\}$$
$$(\exists x : x \in S : \phi(x)) \stackrel{\text{def}}{=} \max\{\phi(x) \mid x \in S\}$$
$$(\forall x : x \in S : \phi(x)) \stackrel{\text{def}}{=} \min\{\phi(x) \mid x \in S\}$$

Next we introduce the syntax of *LogLogics*. We assume that a countable set P of atomic prepositions and a countable set V of clock variables are given. Then, *LogLogics*-formulas ϕ are built from atomic propositions, boolean connectives, "until" $\mathcal{U}$ and "since" $\mathcal{S}$ operators, clock constraints and clock resets. Intuitively, $\phi_1 \mathcal{U} \phi_2$ means that at some point of time in the future ϕ_2 holds and till then ϕ_1 holds. Similarly, $\phi_1 \mathcal{S} \phi_2$ means that at some point of time in the past ϕ_2 holds and from that point onwards ϕ_1 holds. Finally, clock reset $x.\phi$, also known as "freeze", sets the value of clock x to the current time.

Formally, *LogLogics*-formulas ϕ are inductively defined as: $\phi := p \mid x \sim y + c \mid x \sim c \mid x.\phi \mid \mathit{false} \mid \mathit{unknown} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U} \phi_2 \mid \phi_1 \mathcal{S} \phi_2$, where $x, y \in V$, $p \in P$, $\sim$ is one of $<, >, \leq, \geq, =, \neq$ and $c \in \mathbb{N}$. We also assume that the abbreviations $\vee, \Rightarrow, \Leftrightarrow, \mathit{true}$ are defined as usual.

In order to define the formal semantics of *LogLogics* we introduce time sequences and timed words.

Definition 1. *A time sequence $\tau = \tau_0 \tau_1 \ldots \tau_n$ is a finite sequence of times $\tau_i \in \mathbb{N}$, $i \in \mathbb{N}$ such that $\tau_i \leq \tau_{i+1}$ for all i.*

A state sequence $\sigma = \sigma_0 \sigma_1 \ldots \sigma_n$ is a finite sequence of states $\sigma_i \in \Sigma$, $i \in \mathbb{N}$. For any atomic proposition $p \in P$ and any $i \in \{0, \ldots, n\}$, $\sigma_i \vdash p$ is either true or false.

A finite timed word $\rho = (\sigma, \tau)$ is a pair consisting of a state sequence σ and a time sequence τ of the same length n. We also write a timed word as a sequence of pairs $(\sigma_0, \tau_0) \ldots (\sigma_n, \tau_n)$.

We assume that $start, now \in \mathbb{N}$ are two special points such that $start \leq now$. Using the notion of timed state sequences we can introduce the semantics of *LogLogics*. The semantics of TPTL+Past has to be adapted to take *start* and *now* into account. For the sake of simplicity, we define the semantics in the same way it is done for infinite timed words. For that purpose, for any finite timed word $(\sigma_{start}, \tau_{start}) \dots (\sigma_{now}, \tau_{now})$ we define a corresponding infinite timed word $\dots (\sigma_{start-1}, \tau_{start-1})$, $(\sigma_{start}, \tau_{start}) \dots (\sigma_{now}, \tau_{now}), (\sigma_{now+1}, \tau_{now+1}) \dots$ such that $\tau_i = \tau_{now} + (i - now)$ for $i \geq now$ and $\tau_i = \tau_{start} + (i - start)$ for $i \leq start$. It should be noted that for a given finite timed word there exist infinitely many corresponding infinite timed words. Moreover, $\sigma_i \vdash p$ is defined to be *unknown* for any $p \in P$, $i \notin \{start, \dots, now\}$.

Definition 2. *Let ρ be an infinite timed word. Let $i \in \mathbb{Z}$ and $\alpha : V \to \mathbb{R}$ be a partial valuation for the clock variables. Then*

- $\langle \rho, i, \alpha \rangle \models p$ *is equal to* $\sigma_i \vdash p$.
- $\langle \rho, i, \alpha \rangle \models$ false *is always* false*;*
- $\langle \rho, i, \alpha \rangle \models$ unknown *is always* unknown*;*
- $\langle \rho, i, \alpha \rangle \models x \sim c$ *is trueif* $\alpha(x) \sim c$*, and false, otherwise;*
- $\langle \rho, i, \alpha \rangle \models x \sim y + c$ *is true if* $\alpha(x) \sim \alpha(y) + c$*, and false, otherwise;*
- $\langle \rho, i, \alpha \rangle \models x.\phi$ *is equivalent to* $\langle \rho, i, \alpha[x \mapsto \tau_i] \rangle \models \phi$*;*
- $\langle \rho, i, \alpha \rangle \models \neg\phi$ *is*
  - *true if* $\langle \rho, i, \alpha \rangle \models \phi$ *is* false*;*
  - *false if* $\langle \rho, i, \alpha \rangle \models \phi$ *is true ;*
  - *unknown, otherwise;*
- $\langle \rho, i, \alpha \rangle \models \phi_1 \wedge \phi_2$ *is*
  - *true if* $\langle \rho, i, \alpha \rangle \models \phi_1$ *and* $\langle \rho, i, \alpha \rangle \models \phi_2$ *are true;*
  - *false if* $\langle \rho, i, \alpha \rangle \models \phi_1$ *or* $\langle \rho, i, \alpha \rangle \models \phi_2$ *is* false*;*
  - *unknown, otherwise;*
- $\langle \rho, i, \alpha \rangle \models \phi_1 \mathcal{U} \phi_2$ *is equivalent to* $\exists j : i \leq j : (\langle \rho, j, \alpha \rangle \models \phi_2 \wedge \forall k : i \leq k < j : \langle \rho, k, \alpha \rangle \models \phi_1)$*;*
- $\langle \rho, i, \alpha \rangle \models \phi_1 \mathcal{S} \phi_2$ *is equivalent to* $\exists j : j \leq i : (\langle \rho, j, \alpha \rangle \models \phi_2 \wedge \forall k : j \leq k : \langle \rho, k, \alpha \rangle \models \phi_1)$*;*

We say that a *LogLogics*-formula is *closed* if any occurrence of a clock variable x is in the scope of a freeze operator "$x.$". For instance, $x.(x > y + 1 \wedge p)$ is not a closed formula since y does not appear in the scope of "$y.$". One can show that the truth value of a closed *LogLogics*-formula is completely defined by the timed word and the time point, i.e., if ϕ is a closed *LogLogics*-formula, then $\langle \rho, i, \alpha \rangle \models \phi$ is equivalent to $\langle \rho, i, \beta \rangle \models \phi$ for any timed word ρ, time point i and clock valuations α and β. From here on we restrict our attention to closed *LogLogics*-formulas and write $\rho \models \phi$ whenever $\langle \rho, now, \varepsilon \rangle \models \phi$ where ε is the empty valuation function.

Intuitively, $\langle \rho, i, \alpha \rangle \models \phi_1 \mathcal{U} \phi_2$ is evaluated to *true* in the same cases as in the traditional two-valued logics. It is evaluated to *false* when either ϕ_2 is evaluated to *false* everywhere starting from i, or, whenever ϕ_2 is evaluated to *unknown* or *true*, ϕ_1 gets evaluated to *false* before. In particular, $\langle \rho, i, \alpha \rangle \models p \mathcal{U}$ *unknown* evaluates to *false* if and only if $\sigma_j \vdash p$ is *false* for some $j < i$. In all other cases, $\langle \rho, i, \alpha \rangle \models p \mathcal{U}$ *unknown* evaluates to *unknown*.

It is evident that although for a given finite timed word there exist infinitely many infinite timed words corresponding to it, the truth value of a *LogLogics*-formula depends only on the finite timed word itself:

Lemma 1. *Let* $(\sigma_{start},\tau_{start})\dots(\sigma_{now},\tau_{now})$ *be a finite timed word and let* $\rho' = \dots(\sigma'_{start-1},\tau_{start-1})$, $(\sigma_{start},\tau_{start})\dots(\sigma_{now},\tau_{now}),(\sigma'_{now+1},\tau_{now+1})\dots$ *and let* $\rho'' = \dots(\sigma''_{start-1},\tau_{start-1}),(\sigma_{start},\tau_{start})\dots(\sigma_{now},\tau_{now}),(\sigma''_{now+1},\tau_{now+1})\dots$ *be two corresponding infinite timed words. Then,* $\rho' \models \phi$ *is equivalent to* $\rho'' \models \phi$ *for any LogLogics-formula* ϕ.

Based on the temporal operators $\mathcal{S}$ and $\mathcal{U}$ we introduce additional temporal operators "eventually" ($\Diamond\phi := true\,\mathcal{U}\phi$), "always in the future" ($\Box\phi := \neg(\Diamond\neg\phi)$), "once in the past" ($\Diamondblack\phi := true\,\mathcal{S}\phi$) and "always in the past" ($\boxminus := \neg(\Diamondblack\neg\phi)$). The following proposition provides a more direct way to evaluate *LogLogics*-formulas using the four additional temporal operators.

Proposition 1. *The following statements hold:*

- $\langle\rho,i,\alpha\rangle \models \Diamond\phi$ *is equivalent to* $\exists j\ \ i \le j \wedge \langle\rho,j,\alpha\rangle \models \phi$*;*
- $\langle\rho,i,\alpha\rangle \models \Box\phi$ *is equivalent to* $\forall j\ \ i > j \vee \langle\rho,j,\alpha\rangle \models \phi$*;*
- $\langle\rho,i,\alpha\rangle \models \Diamondblack\phi$ *is equivalent to* $\exists j\ \ j \le i \wedge \langle\rho,j,\alpha\rangle \models \phi$*;*
- $\langle\rho,i,\alpha\rangle \models \boxminus\phi$ *is equivalent to* $\forall j\ \ j > i \vee \langle\rho,j,\alpha\rangle \models \phi$*;*

Example 1. Let us evaluate the formula $\boxminus x.(p \Rightarrow \Diamond y.(q \wedge y \le x+4))$ on the finite timed word $\rho = ((\sigma_0,0),(\sigma_1,1),(\sigma_2,1),(\sigma_3,2),(\sigma_4,5),(\sigma_5,8))$ such that $start = 0$, $now = 5$, $\sigma_i \vdash p$ is *true* for $i = 1$ and $i = 4$, and *false* for $i \in \{0,2,3,5\}$; $\sigma_i \vdash q$ is *true* for $i = 3$ and *false* for $i \in \{0,1,2,4,5\}$ (see Fig. 1). Intuitively, this formula says that whenever p was encountered in the past, q was encountered not later than four time units after that.

We need to evaluate this formula with respect to $\langle\rho,5,\varepsilon\rangle$. First, we observe that we need to minimise $\langle\rho,i,\varepsilon\rangle \models x.(p \Rightarrow \Diamond y.(q \wedge y \le x+4))$ for all $i \le 5$. This is equivalent to minimising the value of $\langle\rho,i,[x \mapsto \tau_i]\rangle \models p \Rightarrow \Diamond y.(q \wedge y \le x+4)$ for $i \le 5$. For $i = 0,2,3,5$, $\langle\rho,i,[x \mapsto \tau_i]\rangle \models p$ is *false* and therefore the implication is *true*. The cases left are:

- $i = 4$.
  Since $\sigma_4 \vdash p$ is *true*, the truth value of the implication coincides with the truth value of $\langle\rho,4,[x \mapsto 5]\rangle \models \Diamond y.(q \wedge y \le x+4)$. To determine the latter value we need to maximise the value of $\langle\rho,j,[x \mapsto 5]\rangle \models y.(q \wedge y \le x+4)$ for $j \ge 4$, i.e., the value of $\langle\rho,j,[x \mapsto 5, y \mapsto \tau_j]\rangle \models (q \wedge y \le x+4)$. For each $j \ge 4$ the value of the conjunction

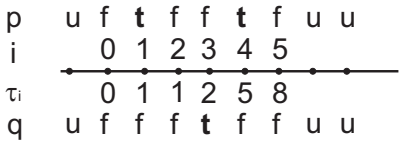

Fig. 1. Finite timed word.

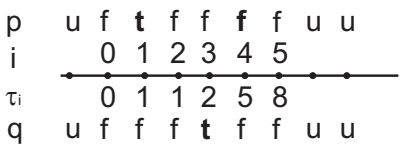

Fig. 2. Finite timed word.

is the smaller value of $\langle \rho, j, [x \mapsto 5, y \mapsto \tau_j] \rangle \models q$ and $\langle \rho, j, [x \mapsto 5, y \mapsto \tau_j] \rangle \models y \leq x+4$.
If $j = 4$ or $j = 5$, $\sigma_j \vdash q$ is *false* and, hence, the value of the conjunction always coincides with it. If $j \geq 6$, $\sigma_j \vdash q$ is *unknown*. Recall that since $j >= now$, $\tau_j = 8 + (j-5)$. If $j = 6$, then $\tau_j = 9$ and $\langle \rho, j, [x \mapsto 5, y \mapsto \tau_j] \rangle \models y \leq x+4$ evaluates to *true*. Hence, the value of the conjunction in this case is *unknown*. If $j > 6$, $\tau_j > 9$ and $\langle \rho, j, [x \mapsto 5, y \mapsto \tau_j] \rangle \models y \leq x+4$ evaluates to *false*. Hence, the smaller value in this case is *false*. To determine the value of the implication for $i = 4$ we should take the maximal value, which is *unknown*, obtained for $j = 6$.

– $i = 1$. As above, since $\sigma_1 \vdash p$ is *true*, the truth value of the implication coincides with the truth value of the maximum (on $j \geq 1$) of the smaller of the two following values: $\langle \rho, j, [x \mapsto 1, y \mapsto \tau_j] \rangle \models q$ and $\langle \rho, j, [x \mapsto 1, y \mapsto \tau_j] \rangle \models y \leq x+4$.
If $j \in \{1,2,4,5\}$, then $\sigma_j \vdash q$ is *false*, and so is the value of the conjunction. If $j \geq 6$ then $\tau_j = \tau_5 + (j-5)$, i.e., $\tau_j \geq 8+1$. Hence, $\langle \rho, j, [x \mapsto 1, y \mapsto \tau_j] \rangle \models y \leq x+4$ evaluates to *false* and the same is true for the conjunction. Finally, for $j = 3$, $\sigma_j \vdash q$ is *true* and $\tau_j = 2 \leq 1+4$. Hence, both conjuncts evaluate to *true* and the conjunction as well. Hence, the maximal value is *true*, $\langle \rho, 1, [x \mapsto 1] \rangle \models \Diamond y.(q \wedge y \leq x+4)$ evaluates to *true* and the truth value of the implication is *true*.
– Finally, consider the case $i < 0$. In this case $\sigma_i \vdash p$ evaluates to *unknown*. Hence, the value of the implication is either *unknown* or *true*.

It should be noted that we do not need to complete the analysis of the last case, as to find the truth value of the original formula, we need to take the smallest value obtained. This value is *unknown* and it is obtained for $i = 4$. □

Example 1 also explains the true meaning of *unknown*. The formula is evaluated to *unknown* due to the behaviour on the boundaries of the observation sequence. Consider the finite timed word in Fig. 2 which differs from the one from Fig. 1 only for $i = 4$. One would expect that the response property from Example 1 is evaluated to true, but due to *unknown* values before *start* it still be *unknown*. In such a case one might like to exclude the interval before *start* and/or after *now* from the consideration. The formulation of the response property ϕ that gives *true* in case all p are followed by q within 4 time units, *unknown* in case there are some p within the distance 4 from *now* that are not followed by q (yet) and all other p are followed by q within 4 time units, and *false* in case there are some p farther than 4 units from *now* not followed by q can be formulated as following:

$$\phi = \boxminus x.(x < \tau_{start} \vee x > \tau_{now} \vee (p \Rightarrow \Diamond y.(q \wedge y \leq x+4))) \quad (1)$$

This formula is evaluated to *unknown* for the word from Fig. 1 and to *true* for the word from Fig. 2.

Since similar restrictions turn out to be useful for expressing interesting business properties, we introduce the following short-hand notation:

$$\begin{aligned}
\Box^{now}_{start}x.\phi(x) &\stackrel{\text{def}}{=} \Box x.(x < \tau_{start} \vee x > \tau_{now} \vee \phi(x)) \\
\Diamond^{now}_{start}x.\phi(x) &\stackrel{\text{def}}{=} \Diamond x.(x < \tau_{start} \vee x > \tau_{now} \vee \phi(x)) \\
\boxminus^{now}_{start}x.\phi(x) &\stackrel{\text{def}}{=} \boxminus x.(x < \tau_{start} \vee x > \tau_{now} \vee \phi(x)) \\
\diamondminus^{now}_{start}x.\phi(x) &\stackrel{\text{def}}{=} \diamondminus x.(x < \tau_{start} \vee x > \tau_{now} \vee \phi(x))
\end{aligned}$$

Subscripts and superscripts of boxes and diamonds can be omitted when only one of boundaries is of interest. Using the short-hand notation formula (1) can be written as

$$\boxminus^{now}_{start}x.(p \Rightarrow \Diamond y.(q \wedge y \leq x+4)). \tag{2}$$

Lemma 2. *Let* a restricted *LogLogics*-formulas ψ *be inductively defined as:* $\psi := p \mid x \sim y+c \mid x \sim c \mid x.\psi \mid \mathit{false} \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \Box^{now}_{start}\psi \mid \boxminus^{now}_{start}\psi \mid \Diamond^{now}_{start}\psi \mid \diamondminus^{now}_{start}\psi$, *where* $x,y \in V$, $p \in P$, $\sim$ *is one of* $<,>,\leq,\geq,=,\neq$ *and* $c \in \mathbb{N}$. *Then* $\rho^* \models \psi$ *is evaluated to true or false for any extension* ρ^* *of a finite timed word* ρ .

3 Typical Guards of Interest

Dwyer *et al.* [5] have identified a number of property specification patterns for software verification. In this section we analogously consider *LogLogics* guard specification patterns for business processes.

Occurrence patterns. The first pattern concerns the occurrence of a certain desired event, or dually, the absence of a certain undesirable event. In the most general form this pattern requires that in a given scope a given event occurs at least a and at most b times. In particular, if $b = 0$, the event does not occur at all, and if a equals the number of time points in a scope, the event occurs through the scope.

Occurrence. For instance, we would like to check that the software license has been renewed in 2005. To encode this property we write $\diamondminus x.(p \wedge x \geq$ 'January 1, 2005' $\wedge x \leq$ 'December 31, 2005'$)$, where p stands for the licence renewal. In general, this pattern has the following form: $\diamondminus x.(\phi \wedge x \geq t_1 \wedge x \leq t_2)$. The value *unknown* is returned in case $[t_1,t_2] \not\subseteq [start,now]$.

Bounded Occurrence. This pattern is similar to the previous one but requires a certain event to occur at least k times within a scope:

$$\begin{aligned}
&\diamondminus x_1.(\phi \wedge x_1 \geq t_1 \wedge x_1 \leq t_2 \wedge \\
&\quad \diamondminus x_2.(\phi \wedge x_2 \geq t_1 \wedge x_2 \leq t_2 \wedge x_2 \neq x_1 \wedge \ldots \\
&\qquad \diamondminus x_k.(\phi \wedge x_k \geq t_1 \wedge x_k \leq t_2 \wedge x_k \neq x_1 \wedge \ldots \wedge x_k \neq x_{k-1})))
\end{aligned}$$

Variants of this pattern require the event to occur exactly k or at most k times. Using this pattern we can express the demand that the employee's salary has been increased at least three times between January 1, 2000 and January 1, 2006.

Absence. This pattern is dual to the occurrence pattern and can be written as $\boxminus x.(\neg\phi(x) \vee x < t_1 \vee x > t_2)$, where ϕ denotes an event undesired between time points t_1 and t_2. In this way we can check that during the last year the daily revenue never fell beyond a given threshold.

Universality. This pattern allows to express properties that should hold through the period from t_1 to t_2: $\boxminus x.(\phi(x) \vee x < t_1 \vee x > t_2)$.

Ordering patterns. Ordering patterns can be constructed from the occurrence patterns by demanding that one of them occurs in a scope within a time slot of another one.

Bounded response. This extremely common pattern (an instance of which we considered in Example 1) allows us to express such guards as "every bill is paid within 30 days". In general, the pattern has the following form $\boxminus^{now}_{start} x.((\phi_1(x) \wedge \pi_1(x)) \Rightarrow \Diamond y.(\phi_2(x) \wedge \pi_2(x,y)))$, where π_1 and π_2 are predicates on the clock values. Requirements on business processes typically restrict the value of y from above.

Precedence. The precedence pattern requires that any occurrence of p is preceded by an occurrence of q within a scope: $\boxminus x.((\phi_1(x) \wedge \pi_1(x)) \Rightarrow \diamondminus y.(\phi_2(x) \wedge \pi_2(x,y)))$, where π_1 and π_2 are predicates on the clock values. For instance, we formulate a guard saying that every failure is preceded by some specific event.

Compound patterns. Finally, compound patterns can be constructed from the patterns above by means of conjunction.

4 Introducing Statistical Values

From here on we restrict our attention to temporal formulas constructed according to patterns discussed in Section 3. Furthermore, we assume that the formulas have been rewritten to the negation normal form, i.e. negation is applied to atomic propositions only.

Now we would like to distinguish between propositions that "hold almost all the time" and those that "almost never hold". To this end we redefine the semantics as function θ from formulas to $[0,1]$, where 0 corresponds to falsity and 1 to truth. This will allow us to define guards that take into account statistical aspects of history rather than a presence or universality of events. For instance, a bank can propose more interesting loans to those customers who pay off the previous loans on-time in at least 95% of the cases. In the current paper we present the refined semantics only for pattern instances where all formulas are in fact atomic propositions.

To this end we need to determine $[0,1]$ truth value of a formula ϕ with respect to the given timed word ρ, index i and clock valuation α.

Occurrence patterns. Occurrence and bounded occurrence are always mapped either to 0 or to 1 according to the traditional semantics.

The truth value of $\langle \rho, i, \alpha \rangle \models \Box x.(p \wedge x \geq t_1 \wedge x \leq t_2)$ for universality formulas is defined as

$$\frac{\Sigma_{[\sigma_i \vdash p]=true}(\tau_{i+1} - \tau_i)}{\min\{now, t_2\} - \max\{start, t_1\}}.$$

For absence, replace $[\sigma_i \vdash p] = true$ in the formula above by $[\sigma_i \vdash p] = false$. The formula says which part of the period of interest the property stayed true/false.

Ordering patterns. Since the sub-formulas are atomic prepositions, $\pi_2(x,y)$ has the following form $\pi_{min}(x) \leq y \leq \pi_{max}(x)$. If one of the inequalities is ommitted we take the value of x instead. Using this observation we define the truth value of $\langle \rho, i, \alpha \rangle \models \phi$ for bounded response and precedence as

$$\frac{\Sigma_{\psi_{brp}(i,j,p)} \zeta_j}{|\{j \mid \psi_{brp}(i,j,p)\}|},$$

where $\psi_{brp}(i,j,p) \stackrel{\text{def}}{=} (i \geq j \wedge i \geq start \wedge j \leq now \wedge \pi_1(\tau_j) \wedge [\sigma_j \vdash p] = true)$, and ζ_j is defined as

$$\begin{cases} 1 & \text{if } \exists m :: \ \xi_{brp}(j,m,q) \wedge start \leq m \leq now \\ 0 & \text{if } \neg\exists m :: \ \xi_{brp}(j,m,q) \wedge \tau_{start} \leq \pi_{min}(\tau_j) \wedge \\ & \quad \pi_{max}(\tau_j) \leq \tau_{now} \\ \frac{(\pi_{max}(\tau_j) - \tau_{now}) + (\tau_{start} - \pi_{min}(\tau_j))}{\pi_{max}(\tau_j) - \pi_{min}(\tau_j)} & \text{otherwise} \end{cases}$$

while $\xi_{brp}(j,m,q) \stackrel{\text{def}}{=} (m \simeq j \wedge \pi_{min}(\tau_j) \leq \tau_m \leq \pi_{max}(\tau_j) \wedge [\sigma_m \vdash q] = true)$. For bounded response $\simeq$ is $\geq$, for precedence $\simeq$ is $\leq$. If a restricted form of a pattern is used, $start \leq m \leq now$ should be added to the definition of $\xi_{brp}(j,m,q)$.

The value obtained indicates in which percentage of cases the property of interest holds.

Compound formulas Truth value θ of compound formulas is determined from the truth values of the sub-formulas according to the following rule:

$$\theta(\langle \rho, i, \alpha \rangle \models \phi_1 \wedge \phi_2) \stackrel{\text{def}}{=} \theta(\langle \rho, i, \alpha \rangle \models \phi_1) * \theta(\langle \rho, i, \alpha \rangle \models \phi_2)$$

Example 2. Let ϕ be $\Box^{now}_{start} x.(p \Rightarrow \Diamond y.(q \wedge y \leq x + 4))$ and ρ, *start* and *now* as in Example 1. This formula is an instance of the bounded response pattern and it is already normalised, so we can compute the corresponding truth value.

Then, for given *now*, ψ_{brp} holds for $j = 1$ and $j = 4$. Hence, the denominator is equal to 2. If $j = 1$ there exists $m = 3$ such that $\xi_{brp}(1,m,q)$ and $start \leq m \leq now$. Therefore, $\zeta_1 = 1$. If $j = 4$ there is no such m. However, $\pi_{max}(\tau_j)$, i.e., $5 + 4 = 9$ is greater than τ_{now}, and the third case of the definition of ζ_j applies. Hence, $\zeta_4 = \frac{1}{4}$—indeed three quarters of the expected "responce period" have passed. Summarising these observations we conclude that the truth value of $\rho \models \phi$ is $\frac{1+\frac{1}{4}}{2} = \frac{5}{8}$. □

5 Conclusion

In this paper we have proposed a logics for guards in models of history-dependent processes. Since at any given moment of time information is finite and inherently incomplete, we had to adapt existing timed temporal logics, which resulted in a three-valued logics, *LogLogics*, presented above. Moreover, we have shown how guard patterns common for business processes can be expressed in our logics and defined an additional extension allowing to express certainty of *LogLogics*-formulas for these patterns.

For the future work we consider creating a compositional generalization of our extension of the logic with uncertainty that would be applicable to all formulas. We also plan to create a simple textual language for working with patterns targeted on non-specialists and to build a tool for checking *LogLogics*-formulas on history logs. As shown in [7], the complexity of checking whether a finite path u satisfies an LTL+P formula φ is $O(|u| \times |\varphi|)$. Therefore the complexity of checking formulas of our logic will not form an obstacle for applying it in practice.

References

1. R. Alur and T. A. Henzinger. A really temporal logic. In *FOCS*, pages 164–169, 1989.
2. R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *REX Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer, 1991.
3. R. Alur and T. A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.
4. P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of TPTL and MTL. In R. Ramanujam and S. Sen, editors, *FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 432–443. Springer, 2005.
5. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *ICSE*, pages 411–420, 1999.
6. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
7. N. Markey and Ph. Schnoebelen. Model checking a path (preliminary report). In R. M. Amadio and D. Lugiez, editors, *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR'03)*, volume 2761 of *Lecture Notes in Computer Science*, pages 251–265, Marseilles, France, Aug. 2003. Springer.
8. O. Morikawa. Extended Gentzen-type formulations of two temporal logics based on incomplete knowledge systems. *Notre Dame Journal of Formal Logic*, 42(1):55–64, 2001.
9. A. Nakamura. On a three-valued logic based on incomplete knowledge systems. Technical Report 1, Japan Research Group of Multiple-valued Logic, The Institute of Electronics, Information and Communication Engineers, 1995. Many-Valued Logic Technical Report.
10. Nick, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *CAiSE05*. Springer.
11. J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*. PhD thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 1999.
12. S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems - a survey. *Data Knowl. Eng.*, 50(1):9–34, 2004.
13. P. Thati and G. Rosu. Monitoring algorithms for metric temporal logic specifications. *Electr. Notes Theor. Comput. Sci.*, 113:145–162, 2005.