SRI International

CSL Technical Report SRI-CSL-01-06 • November 28, 2001

# Results on Event-Recording Logic


Maria Sorea*



This research was supported by the National Science Foundation under grants CCR-00-82560 and CCR-00-86096.



* Also affiliated with Universität Ulm, Abt. Künstliche Intelligenz, Germany.

Computer Science Laboratory • 333 Ravenswood Ave. • Menlo Park, CA 94025 • (650) 326-6200 • Facsimile: (650) 859-2844

**Abstract**

We present construction of characteristic formulas for event-recording automata in terms of the $\mathrm{ERL}_\nu$ logic, a real-time extension of the Hennessy-Milner Logic with greatest fixpoints. By combining the construction of characteristic formulas with the model checking algorithm for $\mathrm{ERL}_\nu$, we obtain a decision procedure for timed bisimilarity and timed similarity for event-recording automata. Furthermore, we present decidability results for the fixpoint free fragment of $\mathrm{ERL}_\nu$, denoted by ERL. We show that the model checking problem for event-recording automata and ERL is PSPACE-complete, and that the satisfiability problem for ERL is decidable. We present a tableau-based decision procedure that allows the decision of whether or not a given ERL formula is valid. We introduce a set of reduction rules for constructing tableaux for ERL formulas, and show that these rules form a sound and complete tableau system.

# Contents

# List of Figures

# Chapter 1

# Introduction

Reactive systems engage in continuous interaction with their environments. These systems are ubiquitous in the world of computing and include many concurrent and real-time systems, embedded and process-control systems, and operating systems. Correct design and correct implementation of reactive systems is of utmost importance since these systems are often deployed in safety-critical environments. A formal approach to the correct realization of reactive systems is of particular interest since it guarantees correctness with respect to a formal specification by means of proof. Given two descriptions of the system namely, an implementation $I$ and a specification $S$ the verification problem is to show that $I$ is correct with respect to $S$. If the implementation is described by a transition system $\mathcal{M}$ and the specification is stated in terms of temporal or modal logic formula $\varphi$ (e.g., [Koz83, Sti87, Eme90]), the verification problem is known as the *model checking* problem, and consists in checking if $\mathcal{M}$ satisfies $\varphi$. Alternatively, both the implementation and the specification can be given in the same formalism. In the so-called automata-based verification approach, $I$ and $S$ are expressed as automata. To establish that the system is correct with respect to its specification, one must check whether the behaviors of the two automata are related to each other. Various behavioral relations, such as bisimulation and simulation, have been proposed in literature. An overview and references to the original literature are given by Glabbeek [vG90]. We say that two systems are bisimular, if they have the same behavior, that is, they have the same branching structure. Automata-based verification techniques and the model checking problem are related to each other by the notion of *characteristic formulas* [GS86, SI94]. A characteristic formula is a characterization of the behavior of a transition system modulo a given behavioral relation. The problem of verifying whether two systems are bisimular can be reduced to a model checking problem, that is, whether one system satisfies the characteristic formula of the other.

We study in this paper characteristic formula construction for real-time systems. As a modeling formalism for real-time systems we choose event-recording automata, while the specifications are stated in terms of an event-recording logic with greatest fixpoints. Event-recording automata have been introduced by Alur, Fix, and Henzinger [AFH99] as a determinizable subclass of timed automata [AD94]. Unlike the timed automata model, where the clocks are maintained by the system, event-recording automata have a fixed, predefined association between clocks and input symbols. Although event-recording automata are strictly less expressive than timed automata, most real-time systems that come up in practice can still be modeled in this restricted formalism, and, arguably, event-recording automata models are often more succinct than corresponding timed automata models, because, in contrast to timed automata, the value of clocks is determined solely by the input. The event-recording logic with greatest fixpoints, $\mathrm{ERL}_{\nu}$, is a fragment of the event-recording fixpoint logic, $\mathrm{ERL}_{\mu\nu}$, which was first introduced in [Sor01]. The logic $\mathrm{ERL}_{\nu}$ includes, for each possible event $a$, a *since operator* $S(a)$ for measuring the time since the last occurrence of this event. In addition, there are real-time modalities such as $\langle\, S(b) \leq 3, a \,\rangle\mathtt{tt}$, which expresses the fact that event $a$ must occur at most three time units after the last occurrence of event $b$. $\mathrm{ERL}_{\nu}$ can be seen as a real-time extension of Hennessy-Milner Logic [MR85] with greatest fixpoints. In [Sor01], it is shown that the model checking problem for $\mathrm{ERL}_{\mu\nu}$, and therefore also for $\mathrm{ERL}_{\nu}$, is decidable. In the framework of event-recording automata and $\mathrm{ERL}_\nu$ we present the construction of characteristic formulas for two behavioral relations, timed bisimilarity, as introduced by Wang Yi [Yi90] and timed similarity, a real-time version of the Milner simulation relation [Mil71]. In both cases the size of the formulas is linear with respect to the size of the corresponding event-recording automaton. The results show that the event-recording logic with greatest fixpoints is expressive enough to characterize event-recording automata up to timed bisimilarity.

Furthermore, the paper presents decidability results for the fixpoint free fragment of $\mathrm{ERL}_{\mu\nu}$, a real-time extension of the Hennessy-Milner Logic. This fragment is denoted by ERL. We show that the model checking problem for event-recording automata and ERL is PSPACE-complete, and that the satisfiability problem for ERL is decidable. We present a tableau-based decision procedure that allows the decision of whether or not a given ERL formula is valid. We introduce a set of reduction rules for constructing tableaux for ERL formulas, and show that these rules form a sound and complete tableau system. For timed logics a tableau rule must necessarily incorporate information about the actual value of the clocks. Therefore, the premises and conclusions of our tableau rules are augmented with a *timing context*. Intuitively, a timing context is a set of clock constraints that is satisfied by the actual valuation of the clocks. For example, $C = \{S(a) = 0, S(b) \geq 3\}$ states that the value of the event-recording clock $S(a)$ equals 0, and that the value of $S(b)$ is equal to or greater than 3, respectively.

## Related Work

For untimed systems it has been shown by Cleaveland and Steffen [CS91] that characteristic formulas in the Hennessy-Milner Logic with recursion [Lar88] can be constructed in time linear in the size of the automaton. In the setting of timed systems, Laroussinie, Larsen, and Weise have presented in [LLW95] characteristic formula construction for timed bisimulation equivalence over timed automata using a real-time version of Hennessy-Milner Logic with greatest fixpoints. This logic, denoted by $\mathcal{L}_\nu$, is expressive enough to represent every timed automaton by a single characteristic formula, which uniquely characterizes the timed automaton up to timed bisimilarity. By combining the characteristic formula construction with the model checking algorithm for $\mathcal{L}_\nu$, an algorithm for checking whether two timed automata are timed bisimilar is obtained. However, the technique presented by the authors produces formulas that are exponential in the size of the automaton. Therefore, checking timed bisimilarity for timed automata in this way is not practicable. Aceto, Ingolfsdottir, Pedersen, and Poulsen [AIPP00] present characteristic formula constructions for timed automata using $\mathcal{L}_\nu$ that yield formulas with size linear in the size of the timed automaton they characterize. Our approach for constructing characteristic formulas is similar to the one advocated in [AIPP00].

One characteristic of the $\mathcal{L}_\nu$ logic is that the formulas usually include new clocks, which are not present in the timed automata model. Moreover, $\mathcal{L}_\nu$ includes an operator for introducing formula clocks. For example, the $\mathcal{L}_\nu$ formula ($x$ in $\varphi$) introduces a formula clock $x$ and initializes it to 0. Therefore, to interpret a formula in a given model $\mathcal{M}$, two different sets of clocks must be regarded, the clocks of the timed automaton and the clocks of the formula. In our logic $\mathrm{ERL}_\nu$ we do not make any distinction between formula and automata clocks, since clocks are related in both formalisms to events, and the same events occur in the event-recording automaton as well as in the formula. This property allows us to give a more concise definition of characteristic formulas for timed bisimulation and timed simulation than the definitions given in [AIPP00] or [LLW95].

As shown in [LLW95] the satisfiability problem for the $\mathcal{L}_\nu$ logic is undecidable. The reason for this is that it is not possible to deduce the number of clocks of the automaton from the number of clocks in the formula. However, the bounded satisfiability problem for $\mathcal{L}_\nu$ is decidable (see [LLW95] for more details). That is, for a given formula and bounds $k$ and $M$, a timed automaton with no more than $k$ clocks and no clock being compared with constants greater than $M$ can be found. This problem does not occur in our logic, since the number of clocks of the automaton is identical with the number of clocks of the formula. Therefore, we hypothesize that the (unbounded) satisfiability problem for the entire event-recording fixed point logic $\mathrm{ERL}_{\mu\nu}$ is decidable. As a first step in establishing this, we show that the satisfiability problem for the fixpoint free fragment of $\mathrm{ERL}_{\mu\nu}$ is decidable.

## Organization of the Paper

The rest of the paper is organized as follows. Event-recording automata and their properties are presented in Chapter 2. The notions of timed similarity and timed bisimilarity are also defined in this chapter. Chapter 3 introduces the event-recording logic with greatest fixpoints, and presents constructions of characteristic formulas for timed similarity and timed bisimilarity in terms of this logic. The tableau-based decision procedure for the event-recording Hennessy-Milner logic is presented in Chapter 4. Chapter 5 contains concluding remarks and an overview of work in progress.

# Chapter 2

# Event-Clock Automata

Event-clock automata were introduced by Alur, Fix, and Henzinger [AFH99] as a determinizable subclass of timed automata. Event-clock automata are characterized by a fixed, predefined association between the clocks and the symbols of the input alphabet. Two types of clock are identified, *event-recording* and *event-predicting*. For the purpose of this paper we restrict ourselves to event-recording clocks. The event-recording clock $S(a)$ of the input symbol $a$ gives the time of the last occurrence of the event $a$ relative to the current time. A similar model of time can be found in [SBM92] and [Sha93]. Unlike in the model of timed automata, the resets of the clocks are not controlled by the system. Instead, all clock values are determined by the input word.

## 2.1 Event-Recording Clocks

Let $\Sigma$ be a finite alphabet. Every symbol $a \in \Sigma$ is associated with an event-recording clock $S(a)$. For a given event-recording automaton, the set of event-recording clocks is denoted by $C_\Sigma = \{S(a) | a \in \Sigma\}$. If the alphabet $\Sigma$ is clear from the context, the subscript $\Sigma$ is omitted. An infinite *timed word* $w$ over the alphabet $\Sigma$ is an infinite sequence $(a_0, t_0)(a_1, t_1) \cdots$ of symbols $a_i \in \Sigma$, paired with nonnegative real numbers $t_i \in \mathbb{R}^{\geq 0}$ such that the sequence $t = t_0 t_1 \cdots$ of time-stamps is nondecreasing (that is, $t_i \leq t_{i+1}$, for all $i \geq 0$). Without loss of generality we assume that $t_0 = 0$.

Given an infinite timed word $w = (a_0, t_0)(a_1, t_1)\cdots$, the value of the clock variable $S(a)$ at position $j$ of $w$ equals $t_j - t_i$, where $i$ is the last position preceding $j$ with $a_i = a$. If there is no such position $i$ then the value of $S(a)$ is undefined and is denoted by $\perp$. In the following, the set of nonnegative reals (nonnegative rationals) will be denoted by $\mathbb{R}^{\geq 0}$ ($\mathbb{Q}^{\geq 0}$). We write $\mathbb{R}_{\perp}^{\geq 0}$ ($\mathbb{Q}_{\perp}^{\geq 0}$) for the set of nonnegative real (rational) numbers together with the value $\perp$. A clock-valuation function $v : C_\Sigma \rightarrow \mathbb{R}_{\perp}^{\geq 0}$ assigns a (positive) real or the undefined value to each clock. Clock constraints compare clock values with rational constants or the undefined value $\perp$.

**Definition 1 (Clock-Valuation Function)** Let $w$ be a timed word. For each position $j \geq 0$ in $w$, the *clock-valuation function* $v_j^w$ is a mapping from $C_\Sigma$ to $\mathbb{R}_{\perp}^{\geq 0}$ defined by

$$v_j^w(S(a)) = \begin{cases} t_j - t_i & \text{if there exists a position } i \text{ such that } 0 \leq i < j \\ & \text{and } a_i = a, \text{ and for all } k \text{ with } i < k < j, a_k \neq a \\ \perp & \text{if } a_k \neq a \text{ for all } 0 \leq k < j. \end{cases}$$

**Definition 2 (Clock Constraints)** Given a set $C_\Sigma$ of clock variables (or simply *clocks*), $S(a) \in C_\Sigma$ an arbitrary clock, $\gamma \in \mathbb{Q}_{\perp}^{\geq 0}$, and $\sim \in \{\leq, \geq, <, >, =\}$, the set $\Phi(C_\Sigma)$ of *clock constraints* over $C_\Sigma$ is defined by the grammar

$$g := \top \mid S(a) \sim \gamma \mid S(a) - S(b) \sim \gamma \mid g_1 \wedge g_2.$$

Usually, comparisons between two clocks such as $S(a) - S(b) \sim \gamma$ are not included in the event-clock automata model. However, as for the timed automata model [AD94], such an extension does not change the expressiveness of event-recording automata. Clock constraints over $C_\Sigma$ are interpreted with respect to clock-valuation functions: atomic constraints of the form $\perp \sim \perp$ and $\perp - \perp \sim \perp$ evaluate to true, all other comparisons that involve $\perp$ (e.g., $\perp \leq 2$) evaluate to false.

**Definition 3** For a clock-valuation function $v$ and a clock constraint $g$ over $C_\Sigma$, we write $v \models g$ (to be read as "$v$ *satisfies* $g$") to denote that according to the values given by $v$ the constraint $g$ evaluates to true. Formally, $v \models g$ is defined inductively over the syntactic structure of $g$, where $S(a), S(b) \in C$ are arbitrary clocks, $\gamma \in \mathbb{Q}_{\perp}^{\geq 0}$ and $\sim \in \{\leq, \geq, <, >, =\}$:

$$
\begin{array}{rcl}
v \models \top & & \\
v \models S(a) \sim \gamma & \text{iff} & v(S(a)) \sim \gamma \\
v \models S(a) - S(b) \sim \gamma & \text{iff} & v(S(a)) - v(S(b)) \sim \gamma \\
v \models g_1 \wedge g_2 & \text{iff} & v \models g_1 \text{ and } v \models g_2
\end{array}
$$

For $\delta \in \mathbb{R}^{\geq 0}$, $v + \delta$ denotes the clock valuation that maps each defined clock $S(a) \in C$ to the value $v(S(a)) + \delta$ and equals $\perp$ for all undefined clocks. For a clock $S(a) \in C$, $v[S(a) := 0]$ denotes the clock valuation for $C$ that maps $S(a)$ to the value 0 and agrees with $v$ over the rest of the clocks.

## 2.2 Event-Recording Automata: Syntax and Semantics

An event-recording automaton is a finite graph augmented with a finite set of *clocks*. The vertices are called *locations* and the edges *transitions*. Transitions are instantaneous, and time can elapse only in locations. On a transition the clock corresponding to the taken event is implicitly reset to zero. At any point, the value of a clock $S(a)$ equals the time elapsed since the last occurrence of the event $a$. With each transition a clock constraint is associated. The transition may be taken only if the current values of the clocks satisfy this constraint.

**Definition 4 (Event-Recording Automata)** An *event-recording automaton* $A$ is a tuple $\langle L, L^0, \Sigma, E \rangle$ where

- $\Sigma$ is a nonempty, finite input alphabet.
- $L$ is a nonempty, finite set of locations.

- $L^0 \subseteq L$ is the set of initial locations.

- $E \subseteq L \times L \times \Sigma \times \Phi(C_\Sigma)$ is a finite set of edges. An edge $e = \langle l, l', a, g\rangle$ represents a transition from location $l$ to location $l'$ with the symbol $a$. The clock constraint $g$ specifies when the transition is enabled.

**Definition 5** A *state* of an event-recording automaton $A$ is a pair $(l, v)$ where $l$ is a location of $A$, and $v$ a clock valuation for $C$. An *initial state* of $A$ is of the form $(l_0, v_0)$ where $l_0 \in L_0$ and $v_0$ maps all clocks in $C$ to $\perp$.

**Definition 6 (Semantics of Event-Recording Automata)** The semantics of an event-recording automaton $A$ is defined by associating a labeled transition system $\mathcal{T}_A = \langle Q_A, Q_A^0, \Sigma \cup \mathbb{R}^{\geq 0}, \rightarrow\rangle$ with it, where $Q_A$ is the set of states of $A$, and $Q_A^0$ is the set of initial states. Two types of transition are defined in $\mathcal{T}_A$:

1. A state can change due to elapse of time: given a state $(l, v)$ and $\delta \geq 0$, then $(l, v)\xrightarrow{\delta}(l', v')$ iff
   - $l = l'$, and
   - for all symbols $b \in \Sigma$, if $v(S(b)) = \perp$ then $v'(S(b)) = \perp$, else $v'(S(b)) = v(S(b)) + \delta$.
2. A state can change by taking a transition between two locations: $(l, v)\xrightarrow{a}(l', v')$ iff there is an edge $\langle l, l', a, g\rangle$ such that $v$ satisfies $g$ and $v' = v[S(a) := 0]$.

## 2.3 Timed Behavioral Relations

We introduce now two behavioral relations over timed labeled transition systems (TLTSs), which we study in this paper. The notion of timed bisimulation was introduced by Wang [Yi90].

**Definition 7 (Timed Simulation / Timed Bisimulation)** Given a timed labeled transition system $\mathcal{T} = \langle Q, Q^0, \Sigma \cup \mathbb{R}^{\geq 0}, \rightarrow\rangle$. A *timed simulation* is a relation $\mathcal{R} \subseteq Q \times Q$ such that whenever $q_1 \mathcal{R} q_2$ and $\alpha \in \Sigma \cup \mathbb{R}^{\geq 0}$ then

- Whenever $q_1 \xrightarrow{\alpha} q_1'$ then there exists $q_2'$ such that $q_2 \xrightarrow{\alpha} q_2'$ and $q_1' \mathcal{R} q_2'$.

A symmetric timed simulation is a *timed bisimulation.*

For states $q_1, q_2$ we write $q_1 \precsim q_2$ (resp. $q_1 \sim q_2$) iff there exists a timed simulation (resp. a timed bisimulation) $\mathcal{R}$ with $q_1 \mathcal{R} q_2$. ($q_1 \precsim q_2$ to read as "$q_2$ simulates $q_1$", which means that all transitions possible in state $q_1$ are also possible in state $q_2$, but $q_2$ can perform more than $q_1$).

Two event-recording automata $A$ and $B$ are similar (bisimilar) if their initial states in the corresponding timed transition systems $\mathcal{T}_A$, respectively $\mathcal{T}_B$, are similar (bisimilar), that is $q_A^0 \, \mathcal{R} \, q_B^0$.

# Chapter 3

# Event-Recording Logic with Greatest Fixpoints

In [Sor01] we introduced a real-time logic that closely reflects the model of event-recording automata. This logic, denoted by $\mathrm{ERL}_{\mu\nu}$, includes for each event $a$ from a given finite alphabet $\Sigma$, a *since operator*, $S(a)$, for measuring the time since the last occurrence of this event. This operator has the same semantics as an event-recording clock in the model of event-recording automata. Since the logic includes least and greatest fixpoint operators, it can be viewed as a real-time extension of the modal $\mu$-calculus [Koz83]. The usual box and diamond modalities are augmented with timing constraints. For example, the real-time modality $\langle\, S(b) \leq 3, a \,\rangle\mathtt{tt}$ expresses the fact that event $a$ must occur at most three time units after the last occurrence of event $b$. In this chapter we present characteristic formula construction for a fragment of $\mathrm{ERL}_{\mu\nu}$ that contains only greatest fixpoints. This fragment, denoted by $\mathrm{ERL}_{\nu}$, can be seen as a real-time extension of Hennessy-Milner logic [MR85] with greatest fixpoints.

**Definition 8 (Syntax of $\mathrm{ERL}_{\nu}$)** Let $\Sigma$ be a finite alphabet, $g$ a timing constraint, and $\mathbf{Id}$ a set of identifiers, and $X \in \mathbf{Id}$. The formulas of the *Event-*

*Recording Logic* with greatest fixpoint $\mathrm{ERL}_\nu$ are defined by the grammar:

$$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle\, g, a\,\rangle\varphi \mid [\,g, a\,]\varphi \mid X$$

The set **Id** of identifiers allows for recursive definition of formulas. A declaration $\mathcal{D}$ assigns to each of the identifiers a formula. For an identifier $X$, $X \stackrel{\mathrm{def}}{=} \varphi$ denotes $\mathcal{D}(X) = \varphi$. Intuitively, $X$ stays for the largest solution of the equation $X \stackrel{\mathrm{def}}{=} \varphi$.

Given a labeled transition system $\mathcal{M} = \langle Q, Q^0, \Sigma \cup \mathbb{R}^{\geq 0}, \rightarrow\rangle$, the semantics of an $\mathrm{ERL}_\nu$ formula is given by the set of states $(l, v) \in Q$, for which the formula holds. Subformulas containing free variables are dealt with using *valuation functions* $\vartheta : V \rightarrow \mathcal{P}(Q_A)$, where $V$ denotes the set of free variables. The updating notation $\vartheta[X := (l, v)]$ denotes the valuation $\vartheta'$ that agrees with $\vartheta$ on all variables except $X$, where $\vartheta'(X) = (l, v)$.

**Definition 9 (Semantics of $\mathbf{ERL}_\nu$)** Let $\mathcal{M}$ be a model, and $\vartheta : V \rightarrow \mathcal{P}(Q)$ an assignment function, the set of states $(l, v)$ in which a formula $\varphi$ is true, $[\![\varphi]\!]^{\mathcal{M}}_{\vartheta}$, is defined inductively as follows:

$$\begin{array}{rcl}
[\![\mathtt{tt}]\!]^{\mathcal{M}}_{\vartheta} & := & Q \\
[\![\mathtt{ff}]\!]^{\mathcal{M}}_{\vartheta} & := & \emptyset \\
[\![X]\!]^{\mathcal{M}}_{\vartheta} & := & \vartheta(X) \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}}_{\vartheta} & := & [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \cap [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \\
[\![\varphi_1 \vee \varphi_2]\!]^{\mathcal{M}}_{\vartheta} & = & [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \cup [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \\
[\![\langle\, g, a\,\rangle\varphi]\!]^{\mathcal{M}}_{\vartheta} & := & \{\, (l, v) \in Q \mid \exists l' \in \{\tilde{l} \mid \langle l, \tilde{l}, a, \tilde{g}\rangle \in E\}, \exists \delta \geq 0. \\
& & \quad v \approx\models g \wedge \tilde{g} \text{ and } (l', v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \text{ where } v' = v[S(a) := 0] + \delta\,\} \\
[\![[\,g, a\,]\varphi]\!]^{\mathcal{M}}_{\vartheta} & := & \{(l, v) \in Q \mid \forall l' \in \{\tilde{l} \mid \langle l, \tilde{l}, a, \tilde{g}\rangle \in E\}, \forall \delta \geq 0. \\
& & \quad v \approx\models g \wedge \tilde{g} \text{ implies } (l', v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \text{ where } v' = v[S(a) := 0] + \delta\,\}
\end{array}$$

We also write $\mathcal{M}, (l, v), \vartheta \models \varphi$, to denote that $(l, v) \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}$. The subscript $\vartheta$ is omitted if the formula does not contain free variables ($\varphi$ is a closed formula).

**Remark 1** The formula $[\,\mathtt{ff}, a\,]\mathtt{ff} = \mathtt{tt}$ since there is no time point at which $a$ can occur. The formula $[\,\mathtt{tt}, c\,]\mathtt{ff}$ disallows the occurrence of a $c$ event.

**Definition 10 (Model Checking Problem)** The *model checking problem* for ERL$_\nu$ is formulated as follows. Given a labeled transition system $\mathcal{M} = \langle Q, Q^0, \Sigma \cup I\!R^{\geq 0}, \rightarrow \rangle$, a state $(l, v) \in Q$, and an ERL$_\nu$ formula $\varphi$, determine whether $\mathcal{M}$ satisfies $\varphi$, that is, $\mathcal{M}, (l, v), \vartheta \models \varphi$, where $\vartheta$ is an assignment function for the free variables of $\varphi$.

The following lemma states some properties of the ERL$_\nu$ logic, which are useful for the proof of Theorem 2.

**Lemma 1** Let $a \in \Sigma$ be an event, $\varphi \in$ ERL$_\nu$ a formula, and $g_1, g_2$ timing constraints. Then the following equivalences hold:

1. $\langle\, g_1 \vee g_2, a \,\rangle\varphi \equiv \langle\, g_1, a \,\rangle\varphi \vee \langle\, g_2, a \,\rangle\varphi$

2. $[\, g_1 \vee g_2, a\,]\varphi \equiv [\, g_1, a\,]\varphi \wedge [\, g_2, a\,]\varphi$

**Proof.** 1. We show that $[\![\langle\, g_1 \vee g_2, a \,\rangle\varphi]\!] = [\![\langle\, g_1, a \,\rangle\varphi \vee \langle\, g_2, a \,\rangle\varphi]\!]$

$$
\begin{aligned}
[\![\langle\, g_1 \vee g_2, a \,\rangle\varphi]\!] \;=\; & \{\mathit{by\ Definition\ 9}\} \\
& \{\, (l, v) \in Q \mid \exists l' \in \{\tilde{l} \,|\, \langle l, \tilde{l}, a, \tilde{g}\rangle \in E\}, \exists \delta \geq 0.\ v \approx\!\!\!\!| \;(g_1 \vee g_2) \wedge \tilde{g} \\
& \;\text{and } (l', v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta},\ \text{where } v' = v[S(a) := 0] + \delta \,\} \\
=\; & \{\mathit{by\ Definition\ 3}\} \\
& \{\, (l, v) \in Q \mid \exists l' \in \{\tilde{l} \,|\, \langle l, \tilde{l}, a, \tilde{g}\rangle \in E\}, \exists \delta \geq 0. \\
& \;(v \approx\!\!\!\!| \; g_1 \wedge \tilde{g} \text{ or } v \approx\!\!\!\!| \; g_2 \wedge \tilde{g}) \text{ and } (l', v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \\
& \;\text{where } v' = v[S(a) := 0] + \delta \,\} \\
=\; & \{\mathit{by\ logical\ equivalences}\} \\
& \{\, (l, v) \in Q \mid \exists l' \in \{\tilde{l} \,|\, \langle l, \tilde{l}, a, \tilde{g}\rangle \in E\}, \exists \delta \geq 0. \\
& \;v \approx\!\!\!\!| \; g_1 \wedge \tilde{g} \text{ and } (l', v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \\
& \;\text{where } v' = v[S(a) := 0] + \delta \,\} \;\cup
\end{aligned}
$$

$$
\begin{aligned}
& \{ (l,v) \in Q \mid \exists l' \in \{\tilde{l} \mid \langle l,\tilde{l},a,\tilde{g}\rangle \in E\}, \exists \delta \geq 0. \\
& \quad v \approx\!\!\models g_2 \wedge \tilde{g} \text{ and } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \\
& \quad \text{where } v' = v[S(a) := 0] + \delta \,\} \\
= \;& \{\mathit{by\ Definition}\ 9\} \\
& [\![\langle\, g_1, a\,\rangle\varphi \vee \langle\, g_2, a\,\rangle\varphi]\!]
\end{aligned}
$$

2. We show that $[\![[\, g_1 \vee g_2, a\,]\varphi]\!] = [\![[\, g_1, a\,]\varphi \wedge [\, g_2, a\,]\varphi]\!]$

$$
\begin{aligned}
[\![[\, g_1 \vee g_2, a\,]\varphi]\!] \;=\;& \{\mathit{by\ Definition}\ 9\} \\
& \{(l,v) \in Q \mid \forall l' \in \{\tilde{l} \mid \langle l,\tilde{l},a,\tilde{g}\rangle \in E\}, \forall \delta \geq 0. \\
& \quad v \approx\!\!\models (g_1 \vee g_2) \wedge \tilde{g} \text{ implies } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \\
& \quad \text{where } v' = v[S(a) := 0] + \delta \,\} \\
=\;& \{\mathit{by\ Definition}\ 3\} \\
& \{(l,v) \in Q \mid \forall l' \in \{\tilde{l} \mid \langle l,\tilde{l},a,\tilde{g}\rangle \in E\}, \forall \delta \geq 0. \\
& \quad (v \approx\!\!\models g_1 \wedge \tilde{g} \text{ or } v \approx\!\!\models g_2 \wedge \tilde{g}) \text{ implies } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \\
& \quad \text{where } v' = v[S(a) := 0] + \delta \,\} \\
=\;& \{\mathit{by\ logical\ equivalences}\} \\
& \{(l,v) \in Q \mid \forall l' \in \{\tilde{l} \mid \langle l,\tilde{l},a,\tilde{g}\rangle \in E\}, \forall \delta \geq 0. \\
& \quad v \approx\!\!\models g_1 \wedge \tilde{g} \text{ implies } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta},\ \text{where } v' = v[S(a) := 0] + \delta \text{ and} \\
& \quad v \approx\!\!\models g_2 \wedge \tilde{g} \text{ implies } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta},\ \text{where } v' = v[S(a) := 0] + \delta \,\} \\
=\;& \\
& \{(l,v) \in Q \mid \forall l' \in \{\tilde{l} \mid \langle l,\tilde{l},a,\tilde{g}\rangle \in E\}, \forall \delta \geq 0. \\
& \quad v \approx\!\!\models g_1 \wedge \tilde{g} \text{ implies } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}, \\
& \quad \text{where } v' = v[S(a) := 0] + \delta \,\} \cap \\
& \{(l,v) \in Q \mid \forall l' \in \{\tilde{l} \mid \langle l,\tilde{l},a,\tilde{g}\rangle \in E\}, \forall \delta \geq 0. \\
& \quad v \approx\!\!\models g_2 \wedge \tilde{g} \text{ implies } (l',v') \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta},
\end{aligned}
$$

$$\text{where } v' = v[S(a) := 0] + \delta \,\}$$

$$= \quad \{\textit{by Definition } 9\}$$

$$[\![\langle\, g_1, a\,\rangle\varphi \wedge \langle\, g_2, a\,\rangle\varphi]\!]$$

□

## 3.1 Characteristic Formula Construction

We present a characteristic formula construction in terms of $\mathrm{ERL}_\nu$ for both the timed behavioral relations introduced in Definition 7. For a given event-recording automaton $A$ and $\mathcal{R} \in \{\,\underset{\sim}{\sqsubset}, \approx\}$ the construction defines the characteristic formula $\Phi^{\mathcal{R}}_{A}(k)$ of a location $k$ of $A$ in terms of similar characteristic formulas of the direct successor locations of $k$. Since the automaton may have cycles, the definition of characteristic formulas is in general a recursive definition, which contains greatest fixpoints.

**Definition 11** Let $A = \langle L, L^0, \Sigma, E\rangle$ be an event-recording automaton, $l \in L$ a location, and $a \in \Sigma$ an action. Then we define the *set of all $a$-labeled edges with source $l$* as follows:

$$E_A(l, a) \overset{\text{def}}{=} \{e \in E \mid e = \langle l, l', a, g\rangle\}.$$

**Proposition 1** Let $A = \langle K, k_0, \Sigma_A, E_A\rangle$ and $B = \langle L, l_0, \Sigma_B, E_B\rangle$ be two event-recording automata, and let $u, v$ be corresponding clock valuations. For any transition $\langle l, l', a, g\rangle$, if $(l, v)\mathcal{R}(k, u)$ and $v \models_{\approx} g$ then $u \models_{\approx} g$.

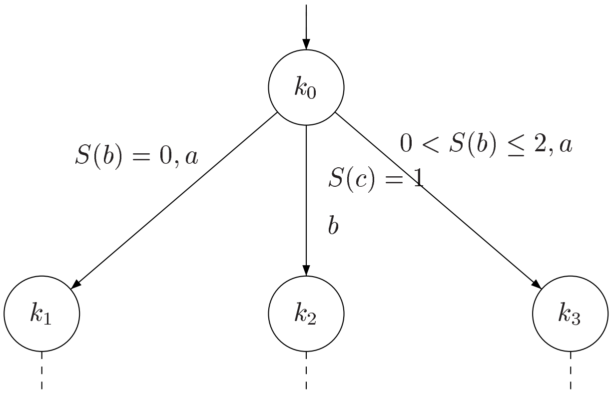
### 3.1.1 Characteristic Formula for Timed Simulation

**Definition 12 (Characteristic Formula for Timed Simulation)** Given an event-recording automaton $A$. The characteristic formula of $A$ consists for each location $k$ of $A$ of the following conjunct:

$$\Phi^{\underset{\sim}{\sqsubset}}_{A}(k) = \bigwedge_{a\in\Sigma}\ \bigwedge_{e\in E_A(k,\,a)} \Big(\langle\, g_e, a\,\rangle \Phi^{\underset{\sim}{\sqsubset}}_{A}(k_e)\Big)$$

where $k$ is a location of $A$, $e = \langle k, k_e, a, g_e \rangle$, and $E_A(k, a)$ denotes the set of $a$-labeled edges from location $k$.

Note that, as usual, an empty disjunction stands for `ff` and an empty conjunction for `tt`. Figure 3.1 illustrates the construction of the simulation formula for event-recording automata.




$$\Phi^{\sqsubseteq}_{\sim}(k_0) := \langle\, S(b) = 0, a \,\rangle \Phi^{\sqsubseteq}_{\sim}(k_1) \wedge \langle\, S(c) = 1, b \,\rangle \Phi^{\sqsubseteq}_{\sim}(k_2) \wedge \langle\, 0 < S(b) \leq 2, a \,\rangle \Phi^{\sqsubseteq}_{\sim}(k_3)$$

Figure 3.1: Simulation formula for event-recording automata.

**Theorem 1** Let $A = \langle K, K_0, \Sigma_A, E_A \rangle$ and $B = \langle L, L_0, \Sigma_B, E_B \rangle$ be two event-recording automata, and $\mathcal{T}_A$, $\mathcal{T}_B$ the corresponding transition systems. Further, let $(k, u), (l, v)$ be arbitrary states of $\mathcal{T}_A$, respectively $\mathcal{T}_B$, where $k \in K$ and $l \in L$. Then $(k, u)$ simulates $(l, v)$ if and only if $\mathcal{T}_A$ satisfies in state $(k, u)$ the characteristic formula of $\mathcal{T}_B$ in location $l$, $\Phi^{\sqsubseteq}_{\sim B}(l)$, that is

$$(l, v) \sqsubseteq_{\sim} (k, u) \;\text{ iff }\; \mathcal{T}_A, (k, u) \models \Phi^{\sqsubseteq}_{\sim B}(l).$$

**Proof.** "$\Rightarrow$" For $k \in K$ and $l \in L$ let $(k, u), (l, v)$ be arbitrary states of $\mathcal{T}_A$, respectively $\mathcal{T}_B$, such that $(k, u)$ simulates $(l, v)$, that is

$$(l, v) \sqsubseteq_{\sim} (k, u).$$

We show that

$$\mathcal{T}_A, (k, u) \models \Phi_{B}^{\underset{\sim}{\sqsubset}}(l)$$

where

$$\Phi_{B}^{\underset{\sim}{\sqsubset}}(l) = \bigwedge_{a \in \Sigma_B} \bigwedge_{e \in E_B(l,\, a)} \Big( \langle\, g_e, a \,\rangle \Phi_{B}^{\underset{\sim}{\sqsubset}}(l_e) \Big), \quad \text{for } e = \langle l, l_e, a, g_e \rangle \in \Sigma_B$$

The proof is a nested induction proof. The first induction is over the elements of the alphabet $\Sigma_B$, the second over the number of edges in the set $E_B(l, a)$, and the last over the nesting depth of $\langle\, g_e, a \,\rangle \Phi_{B}^{\underset{\sim}{\sqsubset}}(l)$.

**I.** Induction over the elements of $\Sigma_B$

**Basis case.** $\Sigma_B = \emptyset$. Then,

$$\Phi_{B}^{\underset{\sim}{\sqsubset}}(l) = \bigwedge_{a \in \Sigma} \emptyset \; = \; \mathtt{tt}$$

Since $\mathcal{T}_A, (k, u) \models \mathtt{tt}$ the desired property follows.

**Induction Hypothesis (IH.I).** The property (1) holds for the alphabet $\Sigma_B = \{a_1, \cdots, a_n\}$.

**Induction Step.** Let $\Sigma_B = \{a_1, \cdots, a_n, a_{n+1}\}$. Then

$$\Phi_{B}^{\underset{\sim}{\sqsubset}}(l) = \bigwedge_{a \in \{a_1, \cdots, a_n\}} \bigwedge_{e \in E_B(l,\, a)} \langle\, g_e, a \,\rangle \Phi_{B}^{\underset{\sim}{\sqsubset}}(l_e) \quad \wedge \quad \bigwedge_{e \in E_B(l,\, a_{n+1})} \langle\, g_e, a_{n+1} \,\rangle \Phi_{B}^{\underset{\sim}{\sqsubset}}(l_e)$$

By Induction Hypothesis (IH.I) we have that

$$\mathcal{T}_A, (k, u) \models \bigwedge_{a \in \{a_1, \cdots, a_n\}} \bigwedge_{e \in E_B(l,\, a)} \langle\, g_e, a \,\rangle \Phi_{B}^{\underset{\sim}{\sqsubset}}(l_e)$$

It remains to show that

$$\mathcal{T}_A, (k, u) \models \Phi_{n+1}$$

where

$$\Phi_{n+1} = \bigwedge_{e \in E_B(l,\, a_{n+1})} \langle\, g_e, a_{a_n+1} \,\rangle \Phi_{B}^{\underset{\sim}{\sqsubset}}(l_e)$$

**II.** This proof follows by induction over the edges of $E_B(l, a_{n+1})$.

**Basis case.** $E_B(l, a_{n+1}) = \emptyset$. Then

$$\bigwedge_{e \in E_B(l,\, a_{n+1})} \langle\, g_e, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e) \quad = \quad \mathtt{tt}$$

and since $\mathcal{T}_A, (k, u) \models \mathtt{tt}$ the desired property follows.

**Induction Hypothesis (IH.II).** For $E_B(l, a_{n+1}) = \{e_1, \cdots e_n\}$

$$\mathcal{T}_A, (k, u) \models \bigwedge_{e \in \{e_1, \cdots e_n\}} \langle\, g_e, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e)$$

holds.

**Induction Step.** Let $E_B(l, a_{n+1}) = \{e_1, \cdots, e_n, e_{n+1}\}$. Then

$$\Phi_{n+1} = \bigwedge_{e \in \{e_1, \cdots, e_n\}} \langle\, g_e, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e) \quad \wedge \quad \langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e)$$

By Induction Hypothesis (IH.II) we have that

$$\mathcal{T}_A, (k, u) \models \bigwedge_{e \in \{e_1, \cdots, e_n\}} \langle\, g_e, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e).$$

It remains to show that

$$\mathcal{T}_A, (k, u) \models \langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e).$$

**III.** The proof follows by induction over the nesting depth of the formula $\langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e)$.

**Basis case.** The nesting depth is 1. Thus,

$$\langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \Phi^{\sqsubseteq}_{\tilde{B}}(l_e) = \langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \mathtt{tt}.$$

Since $\mathcal{T}_B, (l, v) \models \langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \mathtt{tt}$, we obtain by Definition 9 that

$$\exists l' \in \{\tilde{l} \,|\, \langle l, \tilde{l}, a_{n+1}, \tilde{g} \rangle \in E\}, \exists \delta \geq 0.\ v \approx g_{e_{n+1}} \wedge \tilde{g} \text{ and } \mathcal{T}_B, (l', v') \models \mathtt{tt},$$

where $v' = v[S(a_{n+1}) := 0] + \delta$

And since $(l,v) \precsim (k,u)$, it follows by Definition 7 that

$$\exists (k,u) \xrightarrow{a_{n+1}} (k', u[S(a_{n+1}) := 0]) \text{ such that}$$
$$(l', v[S(a_{n+1}) := 0]) \precsim (k', u[S(a_{n+1}) := 0]) \tag{3.1}$$

$$\exists (k', u[S(a_{n+1}) := 0]) \xrightarrow{\delta} (k', u[S(a_{n+1}) := 0] + \delta) \text{ such that}$$
$$(l', v[S(a_{n+1}) := 0] + \delta) \precsim (k', u[S(a_{n+1}) := 0] + \delta) \tag{3.2}$$

From $(l,v) \precsim (k,u)$ and $v \models\approx g_{e_{n+1}} \wedge \tilde{g}$ it follows by Proposition 1 that

$$u \models\approx g_{e_{n+1}} \wedge \tilde{g}. \tag{3.3}$$

(3.1), (3.2) and (3.3) together with Definition 6 lead to

$$\mathcal{T}_A, (k,u) \models \langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \mathtt{tt}.$$

**Induction Hypothesis (IH.III).** $\mathcal{T}_A, (k,u) \models \langle\, g_{e_{n+1}}, a_{n+1} \,\rangle \Phi_B^{\precsim}(l_e)$ for all formula $\Phi_B^{\precsim}(l_e)$ with nesting depth less than or equal to $n$.

**Induction Step.** $(n \to n+1)$ Let $\langle\, g_{e_{n+1}}, a_{a_n+1} \,\rangle \Phi_B^{\precsim}(l_e)$ be a formula with nesting depth $n+1$, where the nesting depth of $\Phi_B^{\precsim}(l_e)$ equals $n$.
Since $\mathcal{T}_B, (l,v) \models \langle\, g_{e_{n+1}}, a_{a_n+1} \,\rangle \Phi_B^{\precsim}(l_e)$ it follows by Definition 9 that

$$\exists l_e \in \{\tilde{l} \,|\, \langle l, \tilde{l}, a_{n+1}, \tilde{g}\rangle \in E\}, \exists \delta \geq 0.\ v \models\approx g_{e_{n+1}} \wedge \tilde{g} \text{ and } \mathcal{T}_B, (l_e, v_e) \models \Phi_B^{\precsim}(l_e)$$
$$\text{where } v_e = v[S(a_{n+1}) := 0] + \delta$$

And since $(l,v) \precsim (k,u)$, it follows by Definition 7 that

$$\exists (k,u) \xrightarrow{a_{n+1}} (k', u[S(a_{n+1}) := 0]) \text{ such that}$$
$$(l_e, v[S(a_{n+1}) := 0]) \precsim (k', u[S(a_{n+1}) := 0]) \tag{3.4}$$

$$\exists (k', u[S(a_{n+1}) := 0]) \xrightarrow{\delta} (k', u[S(a_{n+1}) := 0] + \delta) \text{ such that}$$
$$(l_e, v[S(a_{n+1}) := 0] + \delta) \precsim (k', u[S(a_{n+1}) := 0] + \delta) \tag{3.5}$$

From $(l,v) \precsim (k,u)$ and $v \models\approx g_{e_{n+1}} \wedge \tilde{g}$ it follows by Proposition 1 that

$$u \models\approx g_{e_{n+1}} \wedge \tilde{g}. \tag{3.6}$$

By Induction Hypothesis (IH.III) we have

$$\mathcal{T}_A, (k', u') \models \Phi_B^{\widetilde{\sqsubseteq}}(l_e), \tag{3.7}$$

where $u' = u[S(a_{n+1}) := 0] + \delta$.

(3.4), (3.5), (3.6), and (3.7) yield

$$\exists k' \in \{\tilde{k} \mid \langle k, \tilde{k}, a_{n+1}, \tilde{g}\rangle \in E\}, \exists \delta \geq 0.\ v \models\!\approx g_{e_{n+1}} \wedge \tilde{g} \text{ and } \mathcal{T}_B, (k', u') \models \Phi_B^{\widetilde{\sqsubseteq}}(l_e)$$
$$\text{where } u' = u[S(a_{n+1}) := 0] + \delta$$

which by Definition 9 leads to

$$\mathcal{T}_A, (k, u) \models \langle\, g_{e_{n+1}}, a_{a_n+1}\,\rangle \Phi_B^{\widetilde{\sqsubseteq}}(l_e).$$

"$\Leftarrow$" Given $\mathcal{T}_A, (k, u) \models \Phi_B^{\widetilde{\sqsubseteq}}(l)$, where

$$\Phi_B^{\widetilde{\sqsubseteq}}(l) = \bigwedge_{a \in \Sigma_B} \bigwedge_{e \in E_B(l,\,a)} \Big( \langle\, g_e, a\,\rangle \Phi_B^{\widetilde{\sqsubseteq}}(l_e) \Big), \quad \text{for } e = \langle l, l_e, a, g_e\rangle \in \Sigma_B$$

we have to show that $(l, v) \widetilde{\sqsubseteq} (k, u)$. This proof works by contradiction, that is, we show that

$$(l, v) \widetilde{\not\sqsubseteq} (k, u) \ \ implies \ \ \mathcal{T}_A, (k, u) \not\models \Phi_B^{\widetilde{\sqsubseteq}}(l).$$

From $(l, v) \widetilde{\not\sqsubseteq} (k, u)$ it follows by Definition 7, that there exists some events $\alpha \in \Sigma_A \cup \Sigma_B \cup \mathbb{R}^{\geq 0}$ such that

$$(l, v) \xrightarrow{\alpha} (l_e, v_e)$$

and

$$(k, u) \stackrel{\alpha}{\not\longrightarrow} (k_e, u_e)$$

Thus, by Definition 9 it follows that

$$\mathcal{T}_A, (k, u) \not\models \langle\, g, a\,\rangle \Phi_B^{\widetilde{\sqsubseteq}}(l_e)$$

and therefore

$$\mathcal{T}_A, (k, u) \not\models \Phi_B^{\widetilde{\sqsubseteq}}(l).$$

□

### 3.1.2 Characteristic Formula for Timed Bisimulation

We present the construction of characteristic bisimulation formulas for event-recording automata. A characteristic bisimulation formula characterizes uniquely an event-recording automaton up to timed bisimilarity.

**Definition 13 (Characteristic Formula for Timed Bisimulation)** Given an event-recording automaton $A = \langle K, K_0, \Sigma_A, E_A \rangle$. Let $E_A(k,a)$ be the set of all $a$-transitions of $A$ from location $k$. We define the disjunction of the clock constraints of the set $E_A(k,a)$ by

$$G(k,a) \stackrel{\text{def}}{=} \bigvee_{e \in E_A(k,a)} g_e.$$

Intuitively, the formula $G(k,a)$ describes when the action $a$ can be taken from a state $(k,u)$.

**Definition 14** Given an event-recording automaton $A$. The characteristic formula of $A$ is defined for each location $k$ of $A$ as follows:

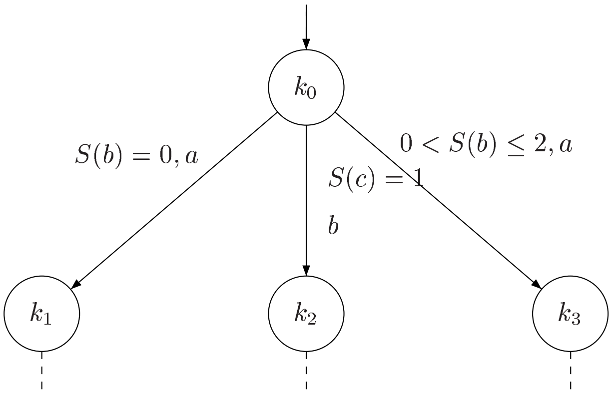
$$\Phi_A^{\approx}(k) = \bigwedge_{a \in \Sigma_A} \Big( \bigwedge_{e \in E_A(k,a)} \langle g_e, a \rangle \Phi_A^{\approx}(k_e) \ \bigwedge \ [G(k,a), a] F(k,a) \ \bigwedge \ [\neg G(k,a), a] \bot \Big)$$

where $k$ is a location of $A$, $e = \langle k, k_e, a, g_e \rangle$, $E_A(k,a)$ denotes the set of $a$-labeled edges from location $k$, and the formula

$$F(k,a) \stackrel{\text{def}}{=} \bigvee_{e \in E_A(k,a)} \Phi_A^{\approx}(k_e)$$

denotes the disjunction of the characteristic formulas for all locations $k_e$ that can be reached from location $k$ by an $a$-transition.

**Example 1** An example illustrates the construction procedure. We construct the characteristic bisimulation formula for the automaton $A$, which can be found in Figure 3.2, in location $k_0$. Let $\Sigma = \{a, b, c\}$. For every event in $\Sigma$ we obtain by Definition 14 the following:




$$\begin{aligned} \Phi^{\approx}(k_0) = \quad & \langle\, S(b) = 0, a\,\rangle \Phi^{\approx}(k_1) \;\wedge\; \langle\, S(c) = 1, b\,\rangle \Phi^{\approx}(k_2) \;\wedge\; \langle\, 0 < S(b) \leq 2, a\,\rangle \Phi^{\approx}(k_3) \;\wedge \\ & [\, 0 \leq S(b) \leq 2, a\,](\Phi^{\approx}(k_1) \vee \Phi^{\approx}(k_3)) \;\wedge\; [\, S(c) = 1, b]\Phi^{\approx}(k_2) \;\wedge \\ & [\, S(b) > 2, a\,]\mathtt{ff} \;\wedge\; [\, S(c) < 1 \vee S(c) > 1, b]\mathtt{ff} \;\wedge\; [\,\mathtt{tt}, c\,]\mathtt{ff} \end{aligned}$$

Figure 3.2: Bisimulation formula for event-recording automata.

| $e \in \Sigma$ | $G(k_0, e)$ | $F(k_0, e)$ |
|---|---|---|
| $a$ | $[S(b) = 0] \vee [0 < S(b) \leq 2] \; = \; [0 \leq S(b) \leq 2]$ | $\Phi^{\approx}_{A}(k_1) \vee \Phi^{\approx}_{A}(k_3)$ |
| $b$ | $[S(c) = 1]$ | $\Phi^{\approx}_{A}(k_2)$ |
| $c$ | $\mathtt{ff}$ | $\mathtt{ff}$ |

And thus we get

$$\begin{aligned} \Phi^{\approx}_{A}(k_0) \quad = \quad & \langle\, S(b) = 0, a\,\rangle \Phi^{\approx}_{A}(k_1) \;\wedge\; \langle\, 0 < S(b) \leq 2, a\,\rangle \Phi^{\approx}_{A}(k_3) \\ \wedge \quad & \langle\, S(c) = 1, b\,\rangle \Phi^{\approx}_{A}(k_2) \\ \wedge \quad & \mathtt{tt} \\ \wedge \quad & [\, 0 \leq S(b) \leq 2, a\,](\Phi^{\approx}_{A}(k_1) \vee \Phi^{\approx}_{A}(k_3)) \\ \wedge \quad & [\, S(c) = 1, b\,]\Phi^{\approx}_{A}(k_2) \\ \wedge \quad & [\,\mathtt{ff}, c\,]\mathtt{ff} \\ \wedge \quad & [\, S(b) > 2, a\,]\mathtt{ff} \\ \wedge \quad & [\, S(c) < 1 \vee S(c) > 1, b\,]\mathtt{ff} \end{aligned}$$

$$\wedge \quad [\mathtt{tt}, c]\mathtt{ff}$$

**Theorem 2** Let $A = \langle K, K_0, \Sigma_A, E_A \rangle$ and $B = \langle L, L_0, \Sigma_B, E_B \rangle$ be two event-recording automata, and $u, v$ be the corresponding clock valuation. Let $(k, u), (l, v)$ be arbitrary states of $A$, respectively $B$, where $k \in K$ and $l \in L$. Then the following holds:

$$(l, v) \approx (k, u) \;\text{ iff }\; A, (k, u) \models \Phi_B^{\approx}(l),$$

where $\Phi_B^{\approx}(l)$ is the characteristic (bisimulation) formula of $B$ in location $l$. We say that $A$ and $B$ are bisimular.

**Proof.** "$\Rightarrow$" Let

$$\Phi_B^{\approx}(l) = \bigwedge_{a \in \Sigma_B} \Big( \bigwedge_{e \in E_B(l,a)} \langle g_e, a \rangle \Phi_B^{\approx}(l_e) \; \bigwedge \; [G(l,a), a] F(l,a) \; \bigwedge \; [\neg G(l,a), a] \bot \Big)$$

be the characteristic (bisimulation) formula for the event-recording automaton $B$ in location $l$, where $e = \langle l, l_e, a, g_e \rangle$, $v : C_{\Sigma_B} \to I\!R_{\bot}^{\geq 0}$ is the clock valuation function, and

$$G(l,a) \stackrel{\text{def}}{=} \bigvee_{e \in E_B(l,a)} g_e \quad \text{and} \quad F(l,a) \stackrel{\text{def}}{=} \bigvee_{e \in E_B(l,a)} \Phi_B^{\approx}(l_e).$$

Further assume $(l, v) \approx (k, u)$. We show by induction over the elements of $\Sigma_B$ that

$$\mathcal{T}_A, (k, u) \models \Phi_B^{\approx}(l).$$

**Basis case.** $\Sigma_B = \emptyset$ Then

$$\Phi_B^{\approx}(l) = \bigwedge_{a \in \Sigma} \emptyset \; = \; \mathtt{tt}$$

Since $\mathcal{T}_A, (k, u) \models \mathtt{tt}$, the desired property follows.

**Induction Hypothesis (IH.I).** $\mathcal{T}_A, (k, u) \models \Phi_B^{\approx}(l)$ holds for the alphabet $\Sigma_B = \{a_1, \cdots, a_n\}$.

**Induction Step.** Let $\Sigma_B = \{a_1, \cdots, a_n, a_{n+1}\}$. Then

$$\Phi_B^{\approx}(l) = \bigwedge_{a' \in \{a_1, \cdots, a_n\}} \Big( \bigwedge_{e \in E_B(l,\, a')} \langle\, g_e, a' \,\rangle \Phi_B^{\approx}(l_e) \ \bigwedge\ [\,G(l,a'), a'\,] F(l,a') \ \bigwedge\ [\,\neg G(l,a'), a'\,] \bot \Big) \tag{3.8}$$

$$\bigwedge_{e \in E_B(l,\, a_{n+1})} \langle\, g_e, a_{n+1} \,\rangle \Phi_B^{\approx}(l_e) \tag{3.9}$$

$$\bigwedge\ [\,G(l, a_{n+1}), a_{n+1}\,] F(l, a_{n+1}) \tag{3.10}$$

$$\bigwedge\ [\,\neg G(l, a_{n+1}), a_{n+1}\,] \bot \tag{3.11}$$

We have to show that $A, (k, u)$ satisfies the above four conjuncts.

**First conjunct (3.8):** $\mathcal{T}_A, (k, u)$ satisfies (3.8) follows by Induction Hypothesis (IH.I).

**Second conjunct (3.9):** In this case the proof follows in the same manner as for Theorem 1.

**Third conjunct (3.10):** For simplicity let $a_{n+1} = a$. We show that $\mathcal{T}_A, (k, u) \models [\,G(l,a), a\,] F(l,a)$.

From Definition 13 and Lemma 1 we get

$$\begin{array}{rcl} & & [\,G(l,a), a\,] F(l,a) \\ \{\text{by Definition 13}\} & \overset{\text{def}}{=} & [\,\bigvee_{e \in E_B(l,\, a)} g_e, a\,] F(l,a) \\ \{\text{by Lemma 1}\} & \overset{\text{def}}{=} & \displaystyle\bigwedge_{e \in E_B(l,\, a)} [\,g_e, a\,] F(l,a) \end{array}$$

It remains to show that

$$\mathcal{T}_A, (k, u) \models \bigwedge_{e \in E_B(l,\, a)} [\,g_e, a\,] F(l,a) \qquad (\star)$$

where

$$F(l,a) \overset{\text{def}}{=} \bigvee_{e \in E_B(l,\, a)} \Phi_B^{\approx}(l_e).$$

The proof of $(\star)$ works by induction over the elements of the set $E_B(l,a)$.

**Basis case.** $E_B(l,a) = \emptyset$. Then, since $\vee\emptyset = \mathtt{ff}$

$$\bigwedge_{e \in E_B(l,a)} [g_e, a]F(l,a) = [\mathtt{ff}, a]\mathtt{ff} = \mathtt{tt}$$

Since $\mathcal{T}_A, (k,u) \models \mathtt{tt}$, the desired property follows.

**Induction Hypothesis (IH.II).** For $E_B(l,a) = \{e_1, \cdots, e_n\}$

$$\mathcal{T}_A, (k,u) \models \bigwedge_{e \in E_B(l,a)} [g_e, a]F(l,a)$$

**Induction Step.** Let $E_B(l,a) = \{e_1, \cdots, e_n, e_{n+1}\}$. Then we obtain the formula

$$\begin{aligned} \Phi = & \bigwedge_{e \in \{e_1, \cdots, e_n\}} [g_e, a]F(l,a) \\ & \bigwedge [g_{e_{n+1}}, a]F(l,a) \end{aligned}$$

By Induction Hypothesis (IH.II) it follows that $\mathcal{T}_A, (k,u)$ satisfies the first conjunct of $\Phi$. It remains to show that

$$\mathcal{T}_A, (k,u) \models [g_{e_{n+1}}, a]F(l,a). \qquad (\star\star)$$

From the assumption of the Theorem we know that

$$\mathcal{T}_B, (l,v) \models [g_{e_{n+1}}, a]F(l,a)$$

In order to prove $(\star\star)$ we proceed by induction over the maximal nesting depth of the formula $[g_{e_{n+1}}, a]F(l,a)$.

**Basis case.** The nesting depth is 1. Then $[g_{e_{n+1}}, a]F(l,a) = [g_{e_{n+1}}, a]\mathtt{tt}$, and since $(l,v) \approx (k,u)$ follows that $\mathcal{T}_A, (k,u) \models [g_{e_{n+1}}, a]\mathtt{tt}$.

**Induction Hypothesis (IH.III).** $\mathcal{T}_A, (k,u) \models [g_{e_{n+1}}, a]F(l,a)$ for all formula $\Phi_B^{\tilde{\sqsubset}}(l_e)$ in $F(l,a)$ with nesting depth less than $n$.

Let $[\,g', a'\,]F(l,a)$ be a formula with nesting depth $n+1$, where the maximal nesting depth of the $\Phi_{\tilde{B}}^{\sqsubseteq}(l_e)$ components of $F(l,a)$ equals $n$.

Since $\mathcal{T}_B, (l,v) \models \langle\, g', a'\,\rangle F(l,a)$ it follows by Definition 9 that

$$\forall l' \in \{\tilde{l} \mid \langle l, \tilde{l}, a', \tilde{g}\rangle \in E\}, \forall \delta \geq 0.\; v \models\!\!\approx g' \wedge \tilde{g} \text{ implies } \mathcal{T}_B, (l', v') \models F(l,a)$$
$$\text{where } v' = v[S(a') := 0] + \delta$$

And since $(l,v) \sqsubseteq\!\!\approx (k,u)$, it follows by Definition 7 that

$$\begin{aligned} &\forall (k,u) \xrightarrow{a'} (k', u[S(a') := 0]) \text{ implies} \\ &\quad (l', v[S(a') := 0]) \sqsubseteq\!\!\approx (k', u[S(a') := 0]) \text{ and} \end{aligned} \tag{3.12}$$

$$\begin{aligned} &\forall (k', u[S(a') := 0]) \xrightarrow{\delta} (k', u[S(a') := 0] + \delta) \text{ implies} \\ &\quad (l', v[S(a') := 0] + \delta) \sqsubseteq\!\!\approx (k', u[S(a') := 0] + \delta) \end{aligned} \tag{3.13}$$

From $(l,v) \sqsubseteq\!\!\approx (k,u)$ and $v \models\!\!\approx g' \wedge \tilde{g}$ it follows by Proposition 1 that

$$u \models\!\!\approx g' \wedge \tilde{g}. \tag{3.14}$$

By Induction Hypothesis (IH.III) we have

$$\mathcal{T}_A, (k', u') \models F(l,a) \tag{3.15}$$

(3.12), (3.13), (3.14), and (3.15) yield

$$\forall k' \in \{\tilde{k} \mid \langle k, \tilde{k}, a', \tilde{g}\rangle \in E\}, \forall \delta \geq 0.\; v \models\!\!\approx g' \wedge \tilde{g} \text{ implies } \mathcal{T}_B, (k', u') \models F(l,a)$$
$$\text{where } u' = u[S(a') := 0] + \delta$$

which by Definition 9 leads to

$$\mathcal{T}_A, (k,u) \models \langle\, g', a'\,\rangle F(l,a).$$

**Fourth conjunct (3.11):** Let $g = \neg G(l,a)$. We have two cases: first $g = \mathtt{ff}$ and second $g \neq \mathtt{ff}$. In the first case, since $[\,\mathtt{ff}, a\,]\mathtt{ff} = \mathtt{tt}$, it follows that $\mathcal{T}_A, (k,u) \models [\,\neg G(l,a), a\,]\mathtt{ff}$.

For the second case, we have from $\mathcal{T}_B, (l, v) \models [\,g, a\,]\mathtt{ff}$ that $\mathcal{T}_B$ cannot perform a $(\delta, a)$-transition with $v + \delta \approx g$. Since $\mathcal{T}_A$ and $\mathcal{T}_B$ are bisimular, the same property holds for the automaton $\mathcal{T}_A$ in the state $(k, u)$. Therefore, $\mathcal{T}_A, (k, u) \models [\,g, a\,]\mathtt{ff}$.

"$\Leftarrow$" This property is proved by contradiction analog the proof of Theorem 1.
$\square$

In [Sor01], it is shown that the model checking problem for fixpoint event-recording logic $\mathrm{ERL}_{\mu\nu}$ is decidable. Thus, timed bisimulation and timed simulation are decidable.

Since the reachability problem for timed automata is PSPACE-hard, as shown by Alur in [Alu91], the model checking problem for event-recording automata and $\mathrm{ERL}_{\mu}$ is PSPACE-hard.

**Corollary 1** The model checking problem for $\mathrm{ERL}_{\nu}$ is PSPACE-hard.

As a consequence of Theorems 2 and 1 and Corollary 1 the following holds.

**Corollary 2** The problem of deciding whether two event-recording automata are bisimilar, respectively similar, is PSPACE-hard.

# Chapter 4

# Event-Recording Hennessy-Milner Logic

We present decidability results for the fixpoint free fragment of the event-recording logic. This fragment, denoted by ERL, is an extension of the Hennessy-Milner Logic with real time. Section 4.1 describes the model checking problem for ERL, and shows that this problem is PSPACE-complete. In Section 4.2 we present a tableau-based decision procedure for the satisfiability problem for ERL.

## 4.1 Model Checking Problem

We analyze the complexity of the model checking problem for ERL. In [Sor01], it is shown that the model checking problem for $\mathrm{ERL}_{\mu\nu}$, a real-time extension of the modal $\mu$-calculus, is decidable. However, no information about the complexity of the problem is provided. We start by defining the model checking problem for ERL. Our claim is that the model checking problem for ERL is PSPACE-complete.

**Definition 15 (Model Checking Problem)** Given an event-recording automaton $A = \langle L, L^0, \Sigma, E\rangle$ with the set of event-recording clocks $C_\Sigma$. Further let $\mathcal{T}_A = \langle Q_A, Q_A^0, \Sigma \cup I\!R^{\geq 0}, \rightarrow\rangle$ be the corresponding labeled transition system, where $Q_A$ is the set of states of the form $s = (l, v)$ with $l \in L$ and $v$ the clock valuation function. Given an ERL-formula $\varphi$ the *model checking problem* consists in verifying whether $\mathcal{T}_A$ satisfies $\varphi$, that is, $\mathcal{T}_A, (l, v) \models \varphi$.

**Theorem 3** The model checking problem for the ERL-Logic is PSPACE-complete.

**Proof.** The proof consists of two parts. First we show that ERL is in PSPACE and then that it is PSPACE-hard.

[PSPACE-membership] Alur, Courcoubetis, and Dill show [ACD93] that a model checking algorithm can be defined by a non deterministic labelling algorithm, which computes the labels of the states as they are required.

[PSPACE-hardness] The proof works by a reduction of the validity problem for QBF (Quantified Boolean Formula) to a model checking problem. Let $\Phi = Q_1 p_1 \cdots Q_n p_n.\varphi$ be an instance of the QBF, where $Q_i \in \{\exists, \forall\}$ and $\varphi$ is a propositional formula over the $p_i$'s. We construct an event-recording automaton $A_\varphi$ over the alphabet $\Sigma_A = \{\epsilon, a_1, \bar{a}_1, \cdots, a_n, \bar{a}_n\}$, where the action $a_i$ is used to encode the positive literal $p_i$, and $\bar{a}_i$ encodes the negative literal $\bar{p}_i$, for $i = \{1, \cdots, n\}$. We use a special event $\epsilon$ to model the first transition of the automaton. $\epsilon$ is an eager event, that is, it must happen at time point $t = 0$. This event is needed for expressing the timing delay of the events $a_1$, $\bar{a}_1$, respectively. The automaton $A_\varphi$ with the timing constraints $g_1 = [S(\epsilon) = 1]$, $g_1' = [S(\epsilon) = 2]$, $g_{i+1} = [S(a_i) = 1 \vee S(\bar{a}_i) = 1]$, and $g_{i+1}' = [S(a_i) = 2 \vee S(\bar{a}_i) = 2]$ is shown in Figure 4.1. The timing constraint $\tilde{\varphi}$ is constructed from the propositional formula $\varphi$ by the substitution

$$\tilde{\varphi} = \varphi[p_i \leftarrow S(a_i) \neq \bot,\ \bar{p}_i \leftarrow S(\bar{a}_i) \neq \bot]$$

Consider the ERL formula

$$\tilde{\Phi} = \langle\, \epsilon \,\rangle \Big( \langle\, g_1, a_1 \,\rangle \tilde{\Phi}_i \diamond \langle\, g_1', \bar{a}_1 \,\rangle \tilde{\Phi}_i \Big)$$

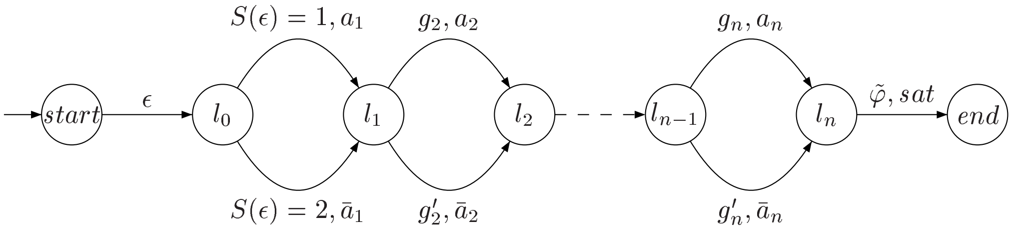



Figure 4.1: Event-recording automaton $A_\varphi$ corresponding to the QBF instance $\Phi = Q_1 p_1 \cdots Q_n p_n . \varphi$.

where $\diamond$ is $\wedge$ (resp. $\vee$) if $Q_i$ is $\forall$ (resp. $\exists$), with

$$\tilde{\Phi}_i = \begin{cases} \langle\, g_n, a_n \,\rangle\langle\, \tilde{\varphi}, sat \,\rangle\texttt{tt} \diamond \langle\, g'_n, \bar{a}_n \,\rangle\langle\, \tilde{\varphi}, sat \,\rangle\texttt{tt}, & \text{if } i = n \\ \langle\, g_i, a_i \,\rangle\tilde{\Phi}_{i+1} \diamond \langle\, g'_i, \bar{a}_i \,\rangle\tilde{\Phi}_{i+1}, & \text{if } i \in \{2, \cdots, n-1\} \end{cases}$$

It remains to show that $\mathcal{T}_{A_\varphi} \models \tilde{\Phi}$ iff $\Phi$ is valid. Assume $\mathcal{T}_{A_\varphi} \models \tilde{\Phi}$. Then the clock valuations $v_i$ in the states of the form $(l_n, v_i)$ satisfy the constraint $\tilde{\varphi}$, which is the enabling condition for the event *sat* to occur. By construction, the values of the clocks $S(a_i), S(\bar{a}_i)$ for all $i \in \{1, \cdots, n\}$ in the states $(l_n, v_i)$ correspond to the satisfying valuation of the QBF formula $\Phi$. Assume now that $\Phi$ is valid. Then there are some states $(l_n, v_i)$ reachable in $A_\varphi$ such that $v_i$ satisfies the constraint $\tilde{\varphi}$. Thus, there exist some paths in $A_\varphi$ leading to a state with location *end*.

$\square$

## 4.2 Tableau-based Decision Procedure

We now address the satisfiability problem for ERL, which was introduced in Section 4.1, Definition 15. We present a tableau-based decision procedure that allows a decision of whether or not a given formula is valid. First, we introduce a set of rules for constructing tableaux and then show that these rules form a sound and complete tableau system. The rules are written with the premises below the line and the conclusion above the line. This is more appropriate for the tableau construction, since a tableau can be viewed as a tree expanding downward.

A tableau rule $\rho$ consists of a *numerator* $\mathcal{N}$ above the line and a list of *denominators* $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$ below the line:

$$\frac{C; \mathcal{N} \ \vdash}{C_1; \mathcal{D}_1 \ \vdash \quad | \ C_2; \mathcal{D}_2 \ \vdash \quad | \ \ldots \ | \ C_n; \mathcal{D}_n \ \vdash} \ (\rho)$$

The numerator and each of the denominators are augmented with a *timing context* $C$. The timing context contains information about the actual value of the clocks. A *tableau system* is a finite collection of tableau rules $\rho_1, \rho_2, \ldots, \rho_m$. Definition 19 contains the tableau rules $\mathrm{T}_{\mathrm{ERL}} = \{(\mathtt{tt}), (\mathtt{ff}), (\wedge), (\vee), (\langle\ \rangle)\}$. For a formula $\varphi$ a $\mathrm{T}_{\mathrm{ERL}}$-tableau is a finite tree with root $\varphi$ whose nodes carry a timing context and finite formula sets. A branch in a tableau is *closed* if the end node is $\{\mathtt{ff}\}$; otherwise it is *open*. A tableau is closed if all its branches are closed.

Before we introduce the tableau rules we define the notion of *timing context* together with the corresponding properties.

**Definition 16 (Timing Context)** Let $\Sigma$ be a finite alphabet. A *timing context* (for short "context") is generated by

$$C ::= \bot \mid \top \mid S(a) \sim \gamma \mid S(a) - S(b) \sim \gamma \mid C_1 \wedge C_2$$

where $S(a), S(b)$ are arbitrary clocks, $\sim \in \{<, \leq, =, >, \geq\}$, and $\gamma \in \mathbb{Q}^{\geq 0}$.

Each context denotes a set of clock valuations in the obvious way:

$$\begin{aligned}
[\![\bot]\!] &= \emptyset \\
[\![\top]\!] &= \{v\} \\
[\![S(a) \sim \gamma]\!] &= \{v \mid v(S(a)) \sim \gamma\} \\
[\![S(a) - S(b) \sim \gamma]\!] &= \{v \mid v(S(a)) - v(S(b)) \sim \gamma\} \\
[\![C_1 \wedge C_2]\!] &= \{v \mid v \in [\![C_1]\!] \text{ and } v \in [\![C_2]\!]\}
\end{aligned}$$

**Remark 2** A timing context corresponds to the set of clock constraints $\Phi(C_\Sigma)$ of a given event-recording automaton.

For a clock valuation $v$ and a timing context $C$ we write $v \models\!\!\approx C$ if the value of $C$, obtained by substituting the clocks $S(\cdot)$ in $C$ with the corresponding value $v(S(\cdot))$, evaluates to true.

On a context $C$ we define the operations of *time elapse* $C\!\uparrow$, which denotes the set of clock valuations obtained by letting time elapse for all clocks in $C$, and *clock reset* $C \odot \{S(a) := 0\}$ as follows. The operation $C_1 \wedge C_2$ is simply defined by intersection.

**Definition 17 (Operations on Context)** Let $C, C_1, C_2$ be timing contexts, $S(a)$ an arbitrary clock, and $v$ a clock valuation function. Then $C\!\uparrow$, $C \odot \{S(a) := 0\}$, $C_1 \wedge C_2$ are *operators on timing contexts* such that the following holds.

$$
\begin{aligned}
[\![C\!\uparrow]\!] &\stackrel{\text{def}}{=} \{v + \delta \mid v \in [\![C]\!] \text{ and } \delta \in I\!R^{\geq 0}\} \\
[\![C \odot \{S(a) := 0\}]\!] &\stackrel{\text{def}}{=} \begin{cases} C \wedge S(a) = 0, & \text{if } S(a) \sim \gamma \notin C \\ (C \wedge S(a) = 0) \setminus S(a) \sim \gamma, & \text{else} \end{cases} \\
[\![C_1 \wedge C_2]\!] &\stackrel{\text{def}}{=} [\![C_1]\!] \cap [\![C_2]\!]
\end{aligned}
$$

Note that the operation of time elapse may enlarge the timing context, as illustrated by the following example.

**Example 2** Let $C = \{S(a) = 0, S(b) > 2\}$ be a timing context. Then

$$C\!\uparrow := \{S(a) \geq 0, S(b) > 2, S(b) - S(a) > 2\}$$

denotes the new context obtained by letting time elapse in $C$.

**Definition 18 (Time-overlapping $a$-Boxes)** Let $\langle g, a \rangle \varphi$ be an ERL formula, and $\Gamma = \{[\,g_1, a_1\,]\varphi_1, [\,g_2, a_2\,]\varphi_2, \cdots, [\,g_k, a_k\,]\varphi_k\}$ a set of box modalities. Then the set of *time-overlapping* $a$-boxes is defined as follows

$$\mathrm{TOB}(g, a) \stackrel{\text{def}}{=} \{\,[\,g', a\,]\varphi' \in \Gamma \mid g' \wedge g \neq \emptyset\,\}.$$

**Definition 19 (Tableau Rules)** Let $\mathrm{T}_{\mathrm{ERL}}$ be the following set of tableau rules:

$$\frac{C;\mathtt{tt},\Gamma\ \vdash}{C;\Gamma\ \vdash}\ (\mathtt{tt})$$

$$\frac{C;\mathtt{ff},\Gamma\ \vdash}{\mathtt{ff}\ \vdash}\ (\mathtt{ff}_\Gamma)$$

$$\frac{\bot;\Gamma\ \vdash}{\mathtt{ff}\ \vdash}\ (\mathtt{ff}_C)$$

$$\frac{C;\varphi\wedge\psi,\Gamma\ \vdash}{C;\varphi,\psi,\Gamma\ \vdash}\ (\wedge)$$

$$\frac{C;\varphi\vee\psi,\Gamma\ \vdash}{C;\varphi,\Gamma\ \vdash\quad\mid\quad C;\psi,\Gamma\ \vdash}\ (\vee)$$

$$\frac{C;\langle g,a\rangle\varphi,\Gamma\ \vdash}{\{C'_z;\Gamma'_z\ \vdash\ \mid z\in Z_g\}}\ (\langle\rangle)$$

In the last rule, $Z_g$ denotes the set of all zones $z$ computed by the algorithm in Figure 4.2. Further, we define for a $z\in Z_g$,

$$C'_z;\ \Gamma'_z \stackrel{\mathrm{def}}{=} \begin{cases} ((C\wedge g)\odot\{S(a):=0\})\!\uparrow;\ \varphi & \text{if } \Gamma=\emptyset \text{ or } \mathrm{TOB}(g,a)=\emptyset \\ ((C\wedge z)\odot\{S(a):=0\})\!\uparrow;\ \{\varphi,\Psi_z\} & \text{else} \end{cases}$$

where $\Psi_z=\{\psi\mid[g',a]\psi\in\mathrm{TOB}(g,a)\text{ and }z\subseteq g'\}$. Note that the rule $(\langle\rangle)$ contains as many denominators as zones $z$ in $Z_g$.

**Remark 3** By construction all zones $z\in Z_g$ are disjoint and $\bigvee_{z\in Z_g} z=g$.

An example illustrates the applicability of the rule $(\langle\rangle)$.

**Example 3** Given the sequent

$$\{S(b)\geq 0\};\langle\, S(b)\geq 1,a\,\rangle\varphi_1,\ \{[\,S(b)\leq 2,a\,]\varphi_2,\ [\,S(b)\geq 2,a\,]\varphi_3\}\ \vdash$$

$$\textbf{Input } \langle\, g, a\,\rangle\varphi, \Gamma \tag{4.1}$$

$$\mathrm{TOB}(g,a) = \{[\,g', a\,]\psi \mid g' \wedge g \neq \emptyset\} \quad (*\ \textit{time-overlapping a-boxes}\ *) \tag{4.2}$$

$$g_1 = g; zl = \{g_1\}; zl' = \emptyset; \tag{4.3}$$

$$\textbf{for all } [\,g', a\,]\psi \in \mathrm{TOB}(g,a) \textbf{ do} \tag{4.4}$$

$$\quad \textbf{for all } z \in zl \textbf{ do} \tag{4.5}$$

$$\quad\quad z' = z \wedge g'; \quad (*\ \textit{zone contained in } z \textit{ and in } g'\ *) \tag{4.6}$$

$$\quad\quad z'' = z \setminus z'; \quad (*\ \textit{zone contained in } z \textit{ and not in } g'\ *) \tag{4.7}$$

$$\quad\quad zl = zl \setminus \{z\}; \tag{4.8}$$

$$\quad\quad \textbf{if } z' \neq \emptyset \textbf{ then } zl' = zl' \cup \{z'\}; \tag{4.9}$$

$$\quad\quad \textbf{if } z'' \neq \emptyset \textbf{ then } zl' = zl' \cup \{z''\} \tag{4.10}$$

$$\quad \textbf{endforall}; \tag{4.11}$$

$$\quad zl = zl' \tag{4.12}$$

$$\textbf{endforall} \tag{4.13}$$

$$\textbf{Output } Z_g = zl. \tag{4.14}$$

Figure 4.2: Algorithm for computing the set $Z_g$.

where the context $C := \{S(b)\geq 0\}$, $g = S(b)\geq 1$, and $\Gamma$ is a conjunction of two formulas $\Gamma = \{[\,S(b)\leq 2, a\,]\varphi_2,\ [\,S(b)\geq 2, a\,]\varphi_3\}$. First, the set of time-overlapping $a$-boxes is computed, $\mathrm{TOB}(g,a) = \{[\,S(b)\leq 2, a\,]\varphi_2, [\,S(b)\geq 2, a\,]\varphi_3\}$. Then, for every time-overlapping box, the zone set $Z_g$ is computed as follow:

- Step 1: Box $[\,S(b)\leq 2, a\,]\varphi_2$
  - Let $z_1$ be the zone contained in $S(b)\geq 1$ and in $S(b)\leq 2$
    $z_1 := 1 \leq S(b) \leq 2$.
  - Let $z_1'$ be the zone contained in $S(b)\geq 1$ but not in $S(b)\leq 2$
    $z_1' := S(b) > 2$.

- Step 2: Box $[\,S(b){\geq}2, a\,]\varphi_3$
  - Let $z_2$ be the zone contained in $z_1$ and in $S(b){\geq}2$
    $z_2 := S(b) = 2$.
  - Let $z_2'$ be the zone contained in $z_1$ but not in $S(b){\geq}2$
    $z_2' := 1 \leq S(b) < 2$.
  - Let $z_2''$ be the zone contained in $z_1'$ and in $S(b){\geq}2$
    $z_2'' = S(b) > 2$.
  - Let $z_2'''$ be the zone contained in $z_1'$ but not in $S(b){\geq}2$
    $z_2''' = \emptyset$.

The non-empty zones computed in the last step are the components of $Z_g$,

$$Z_g = \{z_2, z_2', z_2''\} = \{S(b) = 2, 1 \leq S(b) < 2, S(b) > 2\}$$

For every $z \in Z_g$ the pair $C_z'$; $\Gamma_z'$ is computed according to Definition 19.

$$\begin{array}{rcl}
C'_{z_2} & = & \{S(b) \geq 2 \ \wedge \ S(a) \geq 0 \ \wedge \ S(b) - S(a) = 0\} \\
\Gamma'_{z_2} & = & \{\varphi_1, \varphi_2, \varphi_3\} \\
\\
C'_{z_2'} & = & \{S(b) \geq 1 \ \wedge \ S(a) \geq 0 \ \wedge \ S(b) - S(a) \geq 1 \ \wedge \ S(b) - S(a) < 2\} \\
\Gamma'_{z_2'} & = & \{\varphi_1, \varphi_2\} \\
\\
C'_{z_2''} & = & \{S(b) > 2 \ \wedge \ S(a) \geq 0 \ \wedge \ S(b) - S(a) > 2\} \\
\Gamma'_{z_2''} & = & \{\varphi_1, \varphi_3\}
\end{array}$$

### 4.2.1 Contradiction in a Tableau

A tableau is closed if all branches are closed, that is, the end nodes are `ff`. Therefore, a branch is closed if there is a contradiction (i.e., inconsistency) in $C;\Gamma$. We identify two types of contradiction in a given tableau, an *atomic contradiction* and a *timing contradiction.* An atomic contradiction may occur in $\Gamma$ if some formula and its negation are contained in $\Gamma$. In this case $\Gamma =$ `ff` and therefore there is no model $\mathcal{M}$ such that $\mathcal{M}, s \models$ `ff`. Thus, according to Definition 20, $\Gamma$ is in context $C$ unsatisfiable. A timing contradiction can occur in a context. In this case the context is $\perp$. For example, a context $C = \{S(a) \geq 1 \wedge S(a) < 1\}$ equals $\perp$ since there is no clock valuation $v$ that satisfies $C$. The rules $(\mathtt{ff}_\Gamma)$ and $(\mathtt{ff}_C)$ are used to check for inconsistencies.

**Example 4** The following examples illustrate some applications of the tableau rules.

1. $F_1 = \langle \mathtt{tt}, b \rangle (\langle S(b) = 1, a \rangle \mathtt{tt} \wedge [S(b) = 1, a]\mathtt{ff})$

   The formula $F_1$ is unsatisfiable. A refutation of $F_1$ is shown below. The tableau is closed by the atomic contradiction $\mathtt{tt} \wedge \mathtt{ff}$.

$$\begin{array}{c}
\{\top\};\ \langle \top, b \rangle (\langle S(b) = 1, a \rangle \mathtt{tt} \wedge [S(b) = 1, a]\mathtt{ff})\ \vdash \\ \hline
\{((\top \wedge \top) \odot \{S(b) := 0\})\uparrow\};\ \langle S(b) = 1, a \rangle \mathtt{tt} \wedge [S(b) = 1, a]\mathtt{ff}\ \vdash \\ \hline
\{S(b) \geq 0\};\ \langle S(b) = 1, a \rangle \mathtt{tt}, [S(b) = 1, a]\mathtt{ff}\ \vdash \\ \hline
\{((S(b) \geq 0 \wedge S(b) \geq 1) \odot \{S(a) := 0\})\uparrow\};\ \mathtt{tt}, \mathtt{ff}\ \vdash \\ \hline
\{S(b) \geq 1 \wedge S(a) \geq 0 \wedge S(b) - S(a) \geq 1\};\ \boxed{\mathtt{tt}, \mathtt{ff}}\ \vdash \\ \hline
\mathtt{ff}
\end{array}$$

2. $F_2 = \langle \mathtt{tt}, a \rangle \langle S(a) = 1, b \rangle \langle S(a) < 1, c \rangle \mathtt{tt}$

   The formula $F_2$ is also unsatisfiable. A refutation of $F_2$ is shown below. The tableau is closed by the timing contradiction $S(a) \geq 1 \wedge S(a) < 1$.

$$\begin{array}{c}
\{\top\};\ \langle \top, a \rangle \langle S(a) = 1, b \rangle \langle S(a) < 1, c \rangle \mathtt{tt}\ \vdash \\ \hline
\{S(a) \geq 0\};\ \langle S(a) = 1, b \rangle \langle S(a) < 1, c \rangle \mathtt{tt}\ \vdash \\ \hline
\{S(a) \geq 1 \wedge S(b) \geq 0 \wedge S(a) - S(b) \geq 1\};\ \langle S(a) < 1, c \rangle \mathtt{tt}\ \vdash \\ \hline
\{(\boxed{S(a) \geq 1} \wedge S(b) \geq 0 \wedge S(a) - S(b) \geq 1 \wedge \boxed{S(a) < 1} \wedge S(c) = 0)\uparrow\};\ \mathtt{tt}\ \vdash \\ \hline
\mathtt{ff}
\end{array}$$

### 4.2.2 Soundness and Completeness

**Definition 20** An ERL-formula $\Gamma$ is satisfiable in a context $C$ iff there is a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \approx C$ and $\mathcal{M}, s \models \Gamma$.

**Definition 21** A tableau system is *sound* when if there is a closed tableau for a formula $\varphi$ then $\varphi$ is unsatisfiable.

**Definition 22** A tableau system is *complete* if for every unsatisfiable formula $\varphi$ there is a closed tableau.

The tableau system presented in this paper is sound and complete. For proving this property we need the following Lemma.

**Lemma 2** Let $\Gamma$ be a set of formulas of the form $\langle g, a \rangle\varphi$ or $[g, a]\varphi$ for some events $a \in \Sigma$ and some ERL-formula $\varphi$. Then the following holds:

$$\Gamma \ is\ unsatisfiable$$

$$\text{iff}$$

$$\bigvee_{\langle g,a\rangle\varphi \in \Gamma} \Big(\langle g, a\rangle\varphi, \{[g_i, a]\varphi_i \in \Gamma \mid g \wedge g_i \neq \mathtt{ff}\}\Big) \ is\ unsatisfiable.$$

**Proof.** "$\Rightarrow$" We proceed by induction over the number of $\langle g, a\rangle\varphi$ from $\Gamma$.

**Basis case.** $\Gamma$ contains no $\langle g, a\rangle\varphi$. Then since $\Gamma$ contains only formulas of the form $[g, a]\varphi$, $\Gamma$ is valid. Therefore, "$\Gamma$ *is unsatisfiable*" equals $\mathtt{ff}$. On the other hand we have

$$\bigvee_{\langle g,a\rangle\varphi \in \Gamma} \Big(\langle g, a\rangle\varphi, \{[g_i, a]\varphi_i \in \Gamma \mid g \wedge g_i \neq \mathtt{ff}\}\Big) = \bigvee \emptyset = \mathtt{ff}$$

**Induction Hypothesis.** If $\Gamma$ contains $n$ formulas of the form $\langle g, a\rangle\varphi$, the claim is true.

**Induction Step.** $\Gamma$ contains $n+1$ "diamonds $a$". Together with the $(n+1)$-th diamond, $\Gamma$ may contain also some new $a_{n+1}$-boxes. Let $B(a_{n+1}) = \{[g_j, a_j]\varphi_j \in \Gamma \mid a_j = a_{n+1}\}$ be these boxes. Further, let $\Gamma' = \Gamma \setminus \Gamma''$, where $\Gamma'' = \{\langle g_{n+1}, a_{n+1}\rangle\varphi_{n+1}\}$. If $\Gamma$ is unsatisfiable, then $\Gamma'$ or $\Gamma''$ is unsatisfiable. From $\Gamma'$ unsatisfiable, it follows by Induction Hypothesis (since $\Gamma'$ contains $n$ "$a$-diamonds") that

$$\bigvee_{\langle g,a\rangle\varphi \in \Gamma'} \Big(\langle g, a\rangle\varphi, B_i\Big) \tag{4.15}$$

where $B_i = \{[\, g_i, a\,]\varphi_i \in \Gamma' \mid g \wedge g_i \neq \mathtt{ff}\}$ is unsatisfiable.

From $\Gamma''$ unsatisfiable, it follows also by Induction Hypothesis that

$$\{\langle\, g_{n+1}, a_{n+1}\,\rangle\varphi_{n+1}, \{[\, g_j, a_j\,]\varphi_j \in B(a_{n+1}) \mid g_j \wedge g_{n+1} \neq \mathtt{ff}\}\} \tag{4.16}$$

is unsatisfiable. (4.15) and (4.16) together lead to

$$\bigvee_{\langle\, g, a\,\rangle\varphi \in \Gamma} \Big(\langle\, g, a\,\rangle\varphi, \{[\, g_i, a\,]\varphi_i \in \Gamma \mid g \wedge g_i \neq \mathtt{ff}\}\Big)$$

unsatisfiable.

"$\Leftarrow$" Trivially true, since $\bigwedge_{\langle\, g, a\,\rangle\varphi \in \Gamma} \Big(\langle\, g, a\,\rangle\varphi, \{[\, g_i, a\,]\varphi_i \in \Gamma \mid g \wedge g_i \neq \mathtt{ff}\}\Big) = \Gamma$ and if one of the conjuncts is unsatisfiable then so $\Gamma$. $\square$

**Theorem 4** The tableau system $\mathrm{T_{ERL}}$ is sound and complete.

**Proof. Soundness:**

For all rules

$$\frac{C;\mathcal{N} \;\vdash}{C_1;\mathcal{D}_1 \mid C_2;\mathcal{D}_2 \mid \ldots \mid C_n;\mathcal{D}_n \;\vdash}\ (\rho)$$

we have to show that $C_1;\mathcal{D}_1 \wedge C_2;\mathcal{D}_2 \wedge \ldots \wedge C_n;\mathcal{D}_n$ unsatisfiable *implies* $C;\mathcal{N}$ unsatisfiable. We show the contraposition, that is, for each $\mathrm{T_{ERL}}$-rule we show that if the numerator is satisfiable then at least one of the denominators is satisfiable.

$\boxed{\text{Rule } (\mathtt{tt})}$ From $C;\mathtt{tt},\Gamma$ satisfiable, it follows by Definition 20 that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \approx C$ such that $\mathcal{M}, s \models \mathtt{tt} \wedge \Gamma$. Since $\mathtt{tt} \wedge \Gamma = \Gamma$, we obtain $\mathcal{M}, s \models \Gamma$.

$\boxed{\text{Rule } (\mathtt{ff}_C)}$ Trivially true.

$\boxed{\text{Rule } (\mathtt{ff}_\Gamma)}$ Trivially true.

$\boxed{\text{Rule } (\wedge)}$ From $C;\varphi \wedge \psi,\Gamma$ satisfiable, it follows by Definition 20 that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \approx C$ such that $\mathcal{M}, s \models \varphi \wedge \psi$ and

$\mathcal{M}, s \models \Gamma$. Thus, $\mathcal{M}, s \models \varphi, \psi, \Gamma$.

$\boxed{\text{Rule } (\vee)}$ From $C; \varphi \vee \psi, \Gamma$ satisfiable, it follows by Definition 20 that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \approx C$ such that $\mathcal{M}, s \models \varphi \vee \psi$ and $\mathcal{M}, s \models \Gamma$. Thus, $\mathcal{M}, s \models \varphi \wedge \Gamma$ or $\mathcal{M}, s \models \psi \wedge \Gamma$.

$\boxed{\text{Rule } (\langle\rangle)}$ From $C; \langle\, g, a\,\rangle\varphi, \Gamma$ satisfiable, it follows by Definition 20 that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \approx C$ such that $\mathcal{M}, s \models \langle\, g, a\,\rangle\varphi$ and $\mathcal{M}, s \models \Gamma$. From $\mathcal{M}, s \models \langle\, g, a\,\rangle\varphi$ we get by Definition 9:

$\exists l' \in \{\tilde{l} \mid l \xrightarrow{a,\tilde{g}} \tilde{l}\}, \exists \delta \geq 0. v \approx g \wedge \tilde{g}$ and $\mathcal{M}, (l', v') \models \varphi$ where $v' = v[S(a) := 0] + \delta\,\}$

Further, from $v \approx C \wedge g$ it follows that $v' \approx ((C \wedge g) \odot \{S(a) := 0\})\!\uparrow$. Thus, by Definition 20 we have that

$$\mathcal{M}, (l', v') \models \varphi. \tag{4.17}$$

Now suppose $\mathcal{M}, s \models \Gamma$. We distinguish three cases:

**Case 1** $\Gamma = \emptyset$. Then by Definition 19, $\Gamma'_z = \varphi$ and $C'_z = ((C \wedge g) \odot \{S(a) := 0\})\!\uparrow$. Since $\mathcal{M}, (l', v') \models \varphi$ and $v' \approx C'_z$ it follows by Definition 20 that $\{C'_z; \Gamma'_z \mid z \in Z_g\}$ is satisfiable.

**Case 2** $\Gamma$ contains no time-overlapping $a$-boxes. Analog Case 1.

**Case 3** $\Gamma$ contains time-overlapping $a$-boxes. Let

$$B = \mathrm{TOB}(g, a) = \{[\, g_1, a\,]\psi_1, [\, g_2, a\,]\psi_2, \ldots, [\, g_k, a\,]\psi_k\}$$

be the set of these boxes.

From $\mathcal{M} \models \Gamma$ follows $\mathcal{M} \models B$ and $\mathcal{M} \models \Gamma^*$, where $\Gamma = B \cup \Gamma^*$.

From $\mathcal{M} \models B$ we get $\mathcal{M} \models [\, g_i, a\,]\psi_i$ for all $i = 1, \ldots, k$. Thus, for all $i = 1, \ldots, k$ by Definition 9 we obtain

$\forall l' \in \{\tilde{l} \mid l \xrightarrow{a,\tilde{g}} \tilde{l}\}, \forall \delta \geq 0.\ v \approx g_i \wedge \tilde{g}$ implies $(l', v') \models \psi_i$ where $v' = v[S(a) := 0] + \delta$

Together with Definition 3 we have that

$$v \quad \approx \quad g_1 \wedge \ldots \wedge g_k \tag{4.18}$$

$$\mathcal{M}, (l', v') \quad \models \quad \psi_1 \wedge \ldots \wedge \psi_k \tag{4.19}$$

Let $Z_g$ be the set of all zones $z$ as computed by the algorithm in Figure 4.2. By Definition 19, $\Gamma'_z = \varphi \wedge \Psi_z$ and $C'_z = ((C \wedge z) \odot \{S(a) := 0\})\uparrow$, for all zones $z \in Z_g$. Let $z = g_1 \wedge g_2 \wedge \ldots \wedge g_j$ be such a zone, where $j \in [1, k]$. Then $\Psi_z = \{\psi_1, \ldots, \psi_j\}$. Since $v \approx z$ (due to (4.18)) it follows that

$$v[S(a) := 0] + \delta \approx C'_z. \tag{4.20}$$

All together, we obtain that there is a model $\mathcal{M}$ and a state $s = (l', v')$ such that

- $v' = v[S(a) := 0] + \delta$
- $\mathcal{M}, (l', v') \models \varphi$ (from (4.17))
- $\mathcal{M}, (l', v') \models \Psi_z$ (from (4.19))
- $v' \approx C'_z$ (from (4.20))

which by Definition 20 leads to $C'_z; \Gamma'_z$ is satisfiable.

**Completeness:** For all rules

$$\frac{C; \mathcal{N} \ \vdash}{C_1; \mathcal{D}_1 \mid C_2; \mathcal{D}_2 \mid \ \ldots \ \mid C_n; \mathcal{D}_n \ \vdash} \ (\rho)$$

we have to show that $C; \mathcal{N}$ unsatisfiable implies $C_1; \mathcal{D}_1 \wedge C_2; \mathcal{D}_2 \wedge \ldots \wedge C_n; \mathcal{D}_n$ unsatisfiable. For the rule ($\langle\rangle$) we have by Lemma 2 that the numerator $\mathcal{N} = \{\langle g, a \rangle\varphi, \Gamma\}$ is unsatisfiable in a context $C$ iff

$$\bigvee_{\langle g, a \rangle\varphi \in \mathcal{N}} \Big(\langle g, a \rangle\varphi, \{[g', a]\psi' \in \mathcal{N} \mid g \wedge g' \neq \mathtt{ff}\}\Big) \tag{4.21}$$

is unsatisfiable in $C$. Therefore, for the rule ($\langle\rangle$) we have to show that (1) unsatisfiable implies $C_1; \mathcal{D}_1 \wedge C_2; \mathcal{D}_2 \wedge \ldots \wedge C_n; \mathcal{D}_n$ unsatisfiable.

We show the contraposition; that is, for each $\mathrm{T_{ERL}}$-rule we show that if at least one of the denominators is satisfiable then the numerator is also satisfiable.

$\boxed{\text{Rule } (\mathtt{tt})}$ From $C;\Gamma$ satisfiable, it follows by Definition 20 that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \models\approx C$ such that $\mathcal{M}, s \models \Gamma$. Since $\Gamma = \mathtt{tt} \wedge \Gamma$, it follows that $\mathcal{M}, s \models \mathtt{tt}, \Gamma$. Therefore, $C; \mathtt{tt}, \Gamma$ is satisfiable.

$\boxed{\text{Rule } (\mathtt{ff}_C)}$ Trivially true.

$\boxed{\text{Rule } (\mathtt{ff}_\Gamma)}$ Trivially true.

$\boxed{\text{Rule } (\wedge)}$ From $C; \varphi, \psi, \Gamma$ satisfiable, it follows that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ with $v \models\approx C$ such that $\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models \psi$ and $\mathcal{M}, s \models \Gamma$. Thus, $\mathcal{M}, s \models \varphi \wedge \psi, \Gamma$ and therefore $C; \varphi \wedge \psi, \Gamma$ is satisfiable.

$\boxed{\text{Rule } (\vee)}$ From $C; \varphi, \Gamma$ satisfiable, it follows by Definition 20 that there exists a model $\mathcal{M}_1$ and a state $s_1 = (l_1, v_1)$ with $v_1 \models\approx C$ such that $\mathcal{M}_1, s_1 \models \varphi$ and $\mathcal{M}_1, s_1 \models \Gamma$. Thus, $\mathcal{M}_1, s_1 \models \varphi \wedge \Gamma$. From $C; \psi, \Gamma$ satisfiable, it follows that there exists a model $\mathcal{M}_2$ and a state $s_2 = (l_2, v_2)$ with $v_2 \models\approx C$ such that $\mathcal{M}_2, s_2 \models \varphi$ and $\mathcal{M}_2, s_2 \models \Gamma$. Thus, $\mathcal{M}_2, s_2 \models \psi \wedge \Gamma$. Let $\mathcal{M}$ be a model with $\mathcal{M} = \mathcal{M}_1$ or $\mathcal{M} = \mathcal{M}_2$ and $s$ a state with $s = s_1$ or $s = s_2$. Then $\mathcal{M}, s \models (\varphi \wedge \Gamma) \vee (\psi \wedge \Gamma)$ which leads to $\mathcal{M}, s \models (\varphi \vee \psi) \wedge \Gamma$. Thus, $C; \varphi \vee \psi, \Gamma$ is satisfiable.

$\boxed{\text{Rule } (\langle\rangle)}$ Each of the denominators of this rule has the form $C'_{z_i}; \Gamma'_{z_i}$, where $i \in \{1, \cdots, n\}$ and

$$C'_{z_i};\ \Gamma'_{z_i} \stackrel{\text{def}}{=} \begin{cases} ((C \wedge g) \odot \{S(a) := 0\})\!\uparrow;\ \varphi & \text{if } \Gamma = \emptyset \text{ or } \mathrm{TOB}(g, a) = \emptyset \\ ((C \wedge z_i) \odot \{S(a) := 0\})\!\uparrow;\ \{\varphi, \Psi_{z_i}\} & \text{else} \end{cases}$$

$C$ denotes the context in which the nominator is satisfiable and $\Psi_{z_i} \stackrel{\text{def}}{=} \{\psi \mid [g', a]\psi \in \mathrm{TOB}(g, a) \text{ and } z_i \subseteq g'\}$. We distinguish two cases. First, $\Gamma$ is empty or does not contain any time-overlapping $a$-boxes. In this case, from the assumption that at least one of the denominators is satisfiable we obtain that there exists a model $\mathcal{M}$ and a state $s = (l, v)$ such that

$$v \quad \approx \quad ((C \wedge g) \odot \{S(a) := 0\})\!\uparrow \tag{4.22}$$

$$\mathcal{M}, s \quad \models \quad \varphi \tag{4.23}$$

From the definition of $C'_{z_i}$ together with (4.22), we know that there exists a state $s_0 = (l_0, v_0)$ such that there is an $a$-transition from $s_0$ to $s$ ($s_0 \xrightarrow{a} s$), where $v_0 := v[S(a) := 0] + \delta$. Thus, $v_0 \approx C$, and $\mathcal{M}, s_0 \models \langle\, g, a \,\rangle\varphi$. By Definition 20 we obtain, that the formula $\langle\, g, a \,\rangle\varphi$ (note that $\Gamma = \emptyset$) is satisfiable in the context $C$, that is, $C; \langle\, g, a \,\rangle\varphi, \emptyset$ is satisfiable. For the case that $\Gamma$ does not contain any time-overlapping $a$-boxes, (4.21) reduces to

$$\bigvee_{\langle\, g, a \,\rangle\varphi \,\in\, \mathcal{N}} \Big(\langle\, g, a \,\rangle\varphi, \mathtt{tt}\Big) \tag{4.24}$$

And since $\langle\, g, a \,\rangle\varphi$ holds in the context $C$, (4.24) is satisfiable.

Second, assume $\Gamma$ contains time-overlapping $a$-boxes. Let

$$\mathrm{TOB}(g, a) = \{[\, g_1, a \,]\psi_1, [\, g_2, a \,]\psi_2, \ldots, [\, g_k, a \,]\psi_k\}$$

be the set of these boxes. From the assumption that at least one of the denominators is satisfiable, it follows that there exists a model $\mathcal{M}$, and a state $s$, with $v \approx C'_{z_i}$ such that $\mathcal{M}, s \models \varphi \wedge \Psi_{z_i}$, where $C'_{z_i} \stackrel{\text{def}}{=} ((C \wedge z_i) \odot \{S(a) := 0\})\!\uparrow$ and $\Psi_{z_i} \stackrel{\text{def}}{=} \{\psi \mid [\, g', a \,]\psi \in \mathrm{TOB}(g, a) \text{ and } z_i \subseteq g'\}$, for some $z_i \in Z_g$. Generally, for all $z_i \in Z_g$ we have

$$\mathcal{M}, s \models \bigvee_{z_i \in Z_g} \Big(C'_{z_i}; \{\varphi, \bigwedge_{\psi \in \Psi_{z_i}} \psi\}\Big).$$

This means that the formula

$$\varphi \wedge \bigwedge_{\psi \in \Psi_{z_i}} \psi$$

is satisfiable in the context $C'_{z_i}$. Thus, in the context $C$ the following formula is satisfiable:

$$\bigvee_{z_i \in Z_g} \Big(\langle\, z_i, a \,\rangle\varphi \wedge \bigwedge_{\psi \in \Psi_{z_i}} [\, z_i, a \,]\psi\}\Big) \tag{4.25}$$

Since all $z_i$ are disjoint, we can augment the conjunct $\bigwedge_{\psi \in \Psi_{z_i}} [\, z_i, a\,]\psi$ in (4.25) to obtain a conjunct that is the same in all the disjuncts of (4.25), without changing the satisfiability of the formula. This operation allows us to extract the common conjunct from the disjunction, and we obtain

$$\Big( \bigvee_{z_i \in Z_g} \langle\, z_i, a\,\rangle\varphi \Big) \wedge \Big( \bigwedge_{z_i \in Z_g} \bigwedge_{\psi \in \Psi_{z_i}} [\, z_i, a\,]\psi \Big) \tag{4.26}$$

Now we can apply Lemma 1 and reduce (4.26) to

$$\Big( \langle \bigvee_{z_i \in Z_g} z_i, a\,\rangle\varphi \Big) \wedge \Big( \bigwedge_{z_i \in Z_g} \bigwedge_{\psi \in \Psi_{z_i}} [\, z_i, a\,]\psi \Big) \tag{4.27}$$

Since $\bigvee_{z_i \in Z_g} z_i = g$ we obtain that (4.27) is equivalent to

$$\Big( \langle\, g, a\,\rangle\varphi \Big) \wedge \Big( \bigwedge_{z_i \in Z_g} \bigwedge_{\psi \in \Psi_{z_i}} [\, z_i, a\,]\psi \Big) \tag{4.28}$$

By construction, the second conjunct $( \bigwedge_{z_i \in Z_g} \bigwedge_{\psi \in \Psi_{z_i}} [\, z_i, a\,]\psi )$ denotes exactly the time-overlapping $a$-boxes

$$\mathrm{TOB}(g, a) = \{[\, g', a\,]\psi' \in \Gamma \mid g \wedge g' \neq \texttt{ff}\}.$$

Thus, from (4.28) we obtain that the formula

$$\langle\, g, a\,\rangle\varphi \wedge \{[\, g', a\,]\psi' \in \Gamma \mid g \wedge g' \neq \texttt{ff}\} \tag{4.29}$$

is satisfied in the context $C$, which concludes the proof of completeness for the $\langle\,\rangle$ rule. $\square$

# Chapter 5

# Conclusion

We presented a construction of characteristic formulas for event-recording automata in terms of the $\mathrm{ERL}_\nu$ logic, a real-time extension of the Hennessy-Milner Logic with greatest fixpoints. By combining the construction of characteristic formulas with the model checking algorithm for $\mathrm{ERL}_\nu$, we obtain a decision procedure for timed bisimilarity and timed similarity for event-recording automata. We showed that the satisfiability problem for the event-recording Hennessy-Milner Logic is decidable. This result is a first step in proving the decidability of the satisfiability problem for the event-recording logic $\mathrm{ERL}_{\mu\nu}$. Currently, we extend the timed tableau with rules for formulas with greatest and least fixpoints. In developing the tableau for $\mathrm{ERL}_{\mu\nu}$ we follow basically the approach presented by Niwinski and Walukiewicz in [WN97] that allows the construction of a tableau for a formula of the (untimed) propositional $\mu$-calculus [Koz83].

# Bibliography

[ACD93] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 25 April 1994.

[AFH99] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theoretical Computer Science*, 211(1–2):253–273, January 1999. A preliminary version appeared in the Proceedings of the Sixth International Conference on Computer-aided Verification (CAV 1994), LNCS 818, Springer-Verlag, 1994, pp. 1-13.

[AIPP00] Luca Aceto, Anna Ingolfsdottir, Mikkel Lykke Pedersen, and Jan Poulsen. Characteristic Formulae for Timed Automata. Technical Report RS-00-23, BRICS, September 2000.

[Alu91] R. Alur. *Techniques for Automatic Verification of Real-Time Systems*. PhD thesis, Stanford University, 1991.

[CS91] Rance Cleaveland and Bernhard Steffen. Computing behavioural relations, logically. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium*, volume 510 of *Lecture*

*Notes in Computer Science*, pages 127–138, Madrid, Spain, 8–12 July 1991. Springer-Verlag.

[Eme90] E.A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072, Amsterdam, 1990. Elsevier Science Publishers.

[GS86] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68(1–3):125–145, January/February/March 1986.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27(3):333–354, December 1983.

[Lar88] Kim G. Larsen. Proof systems for Hennessy-Milner Logic with recursion. In M. Dauchet and M. Nivat, editors, *Proceedings CAAP '88*, LNCS 299, pages 215–230, Nancy, March 1988. Springer-Verlag.

[LLW95] F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic — and back. *Lecture Notes in Computer Science*, 969:529–539, 1995.

[Mil71] Robin Milner. An algebraic definition of simulation between programs. In D. C. Cooper, editor, *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 481–489, London, UK, September 1971. William Kaufmann. Also available as Report No. CS-TR-71-205, Computer Science Department, Stanford University.

[MR85] M. Hennessy and R. Milner. Algebraic Laws for Non-Determinism and Concurrency. *Journal of the ACM*, 32:137–161, 1985.

[SBM92] F. B. Schneider, B. Bloom, and K. Marzullo. Putting time into proof outlines. *Lecture Notes in Computer Science*, 600:618–639, 1992.

[Sha93] N. Shankar. Verification of Real-Time Systems Using PVS. In Costas Courcoubetis, editor, *Computer-Aided Verification, CAV '93*, volume 697 of *Lecture Notes in Computer Science*, pages 280–291, Elounda, Greece, June/July 1993. Springer-Verlag.

[SI94] Bernhard Steffen and Anna Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, April 1994.

[Sor01] Maria Sorea. Tempo: A model-checker for event-recording automata. *Workshop on Real-Time Tools*, Aalborg, Denmark, August 2001. Full version available as Technical Report SRI-CSL-01-04, Computer Science Laboratory, SRI International, Menlo Park, CA., 2001.

[Sti87] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49(2-3):311–347, 1987.

[vG90] R. J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In J. C. M. Baeten and J. W. Klop, editors, *CONCUR '90: Theories of Concurrency: Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297, Amsterdam, The Netherlands, 27–30August 1990. Springer-Verlag.

[WN97] I. Walukiewicz and D. Niwiẃski. Games for the $\mu$-calulus. *Theoretical Computer Science*, 163:99–116, 1997.

[Yi90] Wang Yi. Real-time behaviour of asynchronous agents. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR '90: Theories of Concurrency: Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer-Verlag, 1990.