How to Fix It: Using Fixpoints in Different Contexts*

Igor Walukiewicz

LaBRI, Université Bordeaux-1
351, Cours de la Libération
F-33 405, Talence cedex, France

Abstract. In this note we discuss the expressive power of μ-calculi. We concentrate on those that are extensions of propositional modal logics with a fixpoint operator. The objective is to try to match the expressive power of monadic second-order logic. We consider different kinds of models: from trees and transition systems up to traces and timed systems.

1 Introduction

Regular languages are a very interesting class because they have many closure properties and many equivalent characterizations. Just to give an example, regular languages of finite words are characterized by finite automata and also by monadic second-order logic (MSOL). The logical characterization testifies the nice closure properties of the class. The automata characterization permits, among others, the development of an important set of tools used in verification. Regular languages can also be characterized by the μ-calculus. It is a logical characterization, but its good algorithmic properties bring it close to the automata characterization. Indeed, the connections between the μ-calculus and alternating automata are so strong that one can almost say that they are the same modulo the vocabulary used to describe them.

It is easy to interpret MSOL over different domains: trees, traces, graphs, real-time sequences. This way we obtain immediately an expressibility yardstick in all these different settings. We can then hope to adapt other characterizations of regular languages to other domains so that they match expressiveness of MSOL. If successful one gets a similar set of powerful tools as the one for regular languages of finite words.

The goal of this note is to put together three attempts [6,11,7] to match the power of MSOL over three different domains. The presented characterizations will be either in terms of fixpoint logic or in terms of alternating automata. We will begin with the equivalence of the μ-calculus and MSOL on infinite binary trees. We will show how this equivalence generalizes nicely to all transition systems. Next, we will proceed to trace models that are the simplest among so called non-interleaving models for concurrency. Unlike words, which are linear

* Work supported by the ACI Sécurité Informatique VERSYDIS.

F. Baader and A. Voronkov (Eds.): LPAR 2004, LNAI 3452, pp. 184–193, 2005.

orders on events, traces are partial orders on events. The intuition is that if two events are not ordered then they have happened concurrently. Once again, after defining a μ-calculus appropriately we are able to get the equivalence with MSOL and automata over traces. Finally, we will discuss the real-time setting. The situation here is more delicate because both the standard automata model as well as MSOL are undecidable. We will present a recent result which gives a decidable fixpoint calculus but with yet unknown expressive power.

2 The μ-Calculus over Transition Systems

In this section we recall the definition of the μ-calculus and discuss the expressive power of the logic over trees and transitions systems. The μ-calculus is an extension of the modal logic K with fixpoint operators. This addition gives a logic with very interesting expressive power. While modal logic K is less expressive than first-order logic, the μ-calculus expresses a good fragment of the properties definable in monadic second-order logic.

Formulas of the μ-calculus over the sets $Prop = \{p_1, p_2, \ldots\}$ of *propositional constants*, $Act = \{a, b, \ldots\}$ of *actions*, and $Var = \{X, Y, \ldots\}$ of *variables*, are defined by the following grammar:

$$F := Prop \mid \neg Prop \mid Var \mid F \vee F \mid F \wedge F \mid \langle Act \rangle F \mid [Act]F \mid \mu Var.F \mid \nu Var.F$$

Note that we allow negations only before propositional constants. This is not a problem as we will be interested in *sentences*, i.e., formulas where all variables are bound by μ or ν. Negation of a sentence is defined by de Morgan laws and the duality between the least and the greatest fixed points (cf. [3,10]). In the following $\alpha, \beta, \ldots$ will denote formulas of the logic.

Formulas are interpreted in *transition systems* which are tuples of the form $\mathcal{M} = \langle S, \{R_a\}_{a \in Act}, \rho \rangle$, where: S is a nonempty set of *states*, $R_a \subseteq S \times S$ is a binary relation interpreting the action a, and $\rho : Prop \to \mathcal{P}(S)$ is a function assigning to each propositional constant a set of states where this constant holds.

For a given transition system $\mathcal{M}$ and an *assignment* $V : Var \to \mathcal{P}(S)$, the set of states in which a formula φ is true, denoted $\| \varphi \|_V^{\mathcal{M}}$, is defined inductively as follows:

$$\begin{aligned}
\| p \|_V^{\mathcal{M}} &= \rho(p) \qquad \| \neg p \|_V^{\mathcal{M}} = S - \rho(p) \\
\| X \|_V^{\mathcal{M}} &= V(X) \\
\| \langle a \rangle \alpha \|_V^{\mathcal{M}} &= \{s : \exists s'. R_a(s, s') \wedge s' \in \| \alpha \|_V^{\mathcal{M}}\} \\
\| \mu X.\alpha(X) \|_V^{\mathcal{M}} &= \bigcap \{S' \subseteq S : \| \alpha \|_{V[S'/X]}^{\mathcal{M}} \subseteq S'\} \\
\| \nu X.\alpha(X) \|_V^{\mathcal{M}} &= \bigcup \{S' \subseteq S : S' \subseteq \| \alpha \|_{V[S'/X]}^{\mathcal{M}}\}
\end{aligned}$$

We have omitted here the obvious clauses for boolean operators and for $[a]\alpha$ formula. We will omit V in the notation if α is a sentence and will sometimes write $\mathcal{M}, s \models \alpha$ instead of $s \in \| \alpha \|^{\mathcal{M}}$.

The question we are interested in is what kind of properties are expressible in the logic. For this we introduce the notion of a *pointed transition system* which is a pair $(\mathcal{M}, s)$ consisting of a transition system and its state. A μ-calculus sentence defines a class of pointed transition systems $(\mathcal{M}, s)$ such that $\mathcal{M}, s \vDash \alpha$.

To answer the question about the expressive power we will first consider a restricted class of graphs, namely deterministic trees. For this class we get a perfect match with other well known formalism: monadic second-order logic.

Monadic second-order logic (MSOL) is an extension of first-order logic with quantification over sets of elements. The signature of the logic consists of one binary relation R_a for each $a \in Act$ and a monadic relation P_p for each $p \in Prop$. Let $Var_1 = \{x, y, \dots\}$ and $Var_2 = \{X, Y, \dots\}$ be the sets first and second order variables, respectively. The syntax of MSOL is given by the usual rules for the first-order logic together with the following:

- if $X \in Var_2$, $y \in Var_1$ and α is a formula then $X(y)$ and $\exists X.\alpha$ are formulas.

Given a transition system $\mathcal{M} = \langle S, \{R_a\}_{a \in Act}, \rho \rangle$, the semantic of a formula is defined as in the first-order logic. Because we have both first and second-order variables, the valuation V should assign to a variable either an element or a set of elements respectively. The two new constructs are interpreted by:

- $M, V \vDash X(y)$ iff $V(y) \in V(X)$,
- $M, V \vDash \exists X.\alpha$ iff there is $S' \subseteq S$ such that $M, V[S'/X] \vDash \alpha$.

A pointed transition system $(\mathcal{M}, s)$ can be considered as a transition system with one more proposition P_{start} that is true only in s. With this convention we can talk about a class of pointed transition systems defined by a MSOL formula.

A *deterministic tree* is a transition system whose underlying edge labeled graph is a tree and such that for each node and for each relation R_a there is at most one outgoing transition on R_a from the node. Such a tree can be also considered as a pointed transition system where the chosen state is the root of the tree. Observe that for every deterministic tree $\mathcal{M}$ and a state s the truth of a μ-calculus formula in s depends only on the subtree rooted in s. Thus for definability it does not make much sense to consider pointed structures with other distinguished states than the root.

Theorem 1 ([8]). *A set of deterministic trees is definable by a MSOL sentence iff it is definable by a μ-calculus sentence.*

The restriction to deterministic trees is necessary because over graphs MSOL can say much more than the μ-calculus about structural properties of graphs. It can say for example that there is a cycle in a transition system; this explains why we need to restrict to trees. It can also say that the number successors of a node is bigger than some threshold; this explains the condition of determinism.

Consider a standard definition of bisimulation between pointed transition systems (where it is required that bisimilar states should satisfy the same propositions). The two structural properties mentioned above are not invariant under

bisimulation while all μ-calculus definable properties are. Given this, the following theorem says that over transition systems the μ-calculus has the best possible expressive power.

Definition 1. *A property is* invariant under bisimulation *iff for arbitrary two pointed transition systems either both have the property, or both do not have it.*

Theorem 2 ([6]). *A property of transition systems is definable in the μ-calculus iff it is definable in MSOL and it is invariant under bisimulation.*

3 Traces

A trace is a partial order with some particular properties. It can be used to represent an execution of the system where we do not want to put some artificial order on actions that appear concurrently. A trace can be represented by a transition system so it makes sense to talk about monadic second-order logic and the μ-calculus properties of traces. Due to the particularity of the setting it is not reasonable to consider bisimulation invariant properties of traces. Thus the theorem from the previous section looses its appeal. We will see that the μ-calculus is weaker than MSOL over traces. One can obtain the equivalence by giving to the μ-calculus access to some information about the concurrency in the system.

A *trace alphabet* is a pair (Σ, D) where Σ is a finite set of actions and $D \subseteq \Sigma \times \Sigma$ is a reflexive and symmetric *dependence relation*.

A *Σ-labeled graph* is $\langle S, R, \lambda \rangle$ where S is the set of vertices, R defines the edges and $\lambda : S \to \Sigma$ is a labeling function.

Definition 2. *A* trace *or a* dependence graph *over a trace alphabet (Σ, D) is a Σ-labeled graph $\langle E, R, \lambda \rangle$ satisfying the following conditions:*

(T1) $\forall e, e' \in E.\ (\lambda(e), \lambda(e')) \in D \Leftrightarrow (R(e, e') \vee R(e', e) \vee e = e')$.
(T2) *the transitive closure R^* of R is a partial order and* $\{e' : R^*(e', e)\}$ *is finite for every e.*

The nodes of a dependence graph are called events. *An a-*event *is an event $e \in E$ which is labeled by a, i.e., $\lambda(e) = a$. We say that e is* before *e' iff $R^*(e, e')$ holds. In this case we also say that e' is* after *e.*

The first condition of the definition of dependence graphs says that the events are related only if they are labeled by dependent letters. The second says that there cannot be neither circular dependencies nor infinite descending chains. So the past of each event (i.e. the set of events that appear before it) is finite. The traces we consider here are sometimes called real traces [5]. From the definition it follows that they are either finite or countably infinite.

Definition 3. *A* transition system representation *of a trace $G = \langle E, R, \lambda \rangle$ is a transition system $\mathcal{M}(G) = \langle E, R_H, \rho \rangle$ over the set of propositions $\{P_a : a \in \Sigma\}$ where (i) the function ρ is defined by $\rho(P_a) = \{e : \lambda(e) = a\}$; and (ii) R_H is the smallest relation needed to determine R, i.e., $R_H^* = R^*$ and if $R_H(e, e')$ then there is no e'' different from e and e' such that $R_H(e, e'')$ and $R_H(e'', e')$ hold.*

The dependence graph representation has too many transitions to be interesting for the logic as the μ-calculus. Working with this representation would be similar to considering transitive closure of relations and not relations themselves. The transition system representation we have defined uses Hasse diagram of the trace (which exists and is unique in this setting). To be consistent with the definitions from the previous section we also need to change the labeling function. In traces this function assigns a label to each node, in transition systems it assigns a set of nodes to each label. Observe that a representation of a trace is a transition system with a singleton action alphabet.

Thanks to the above definition we can use formalisms of transition systems to talk about traces. In particular we will talk about μ-calculus or MSOL definable sets of traces considered as pointed transition systems. In principle there can be more than one minimal element in a trace and this can pose a problem with definition of a pointed transition system. Because of this we assume a relatively harmless proviso that every trace has the least event. Thus a transition system $\mathcal{M}(G)$ can be considered as pointed transition system $(\mathcal{M}(G), e)$ where e is the least event of G.

It turns out that the class of MSOL definable trace languages have very similar characterizations as in the case of infinite words. The following theorem summarizing many results on traces can be found in [4].

Theorem 3. *Fix a given trace alphabet. For a set L of traces the following are equivalent:*

- *L is definable by a MSOL formula.*
- *L is definable by a c-regular expression.*
- *L is recognizable by an asynchronous automaton.*
- *The set of linearizations of traces in L is a recognizable language of finite and infinite words.*

We will not need asynchronous automata or c-regular expressions, so we will not define them here. If linearizations are concerned, let us just say that a linearization of a trace is an infinite word which corresponds to some linear order of type ω extending the partial order of the trace.

We next show that the μ-calculus over traces cannot even express some first-order definable properties. This example motivates the search for extensions of the μ-calculus that can capture the power of MSOL over traces.

Proposition 1. *No μ-calculus sentence can distinguish between the transition system representations of the two traces presented in Figure 1. In the left graph the dots stand for the sequence $(dc)^\omega$ and in the right graph for $(cd)^\omega$. In this example the trace alphabet $(\{\bot, a, b, c, d\}, D)$ where D is the smallest symmetric and reflexive relation containing the pairs $\{(a, c), (b, d), (c, d)\} \cup \{\bot\} \times \{a, b, c, d\}$.*

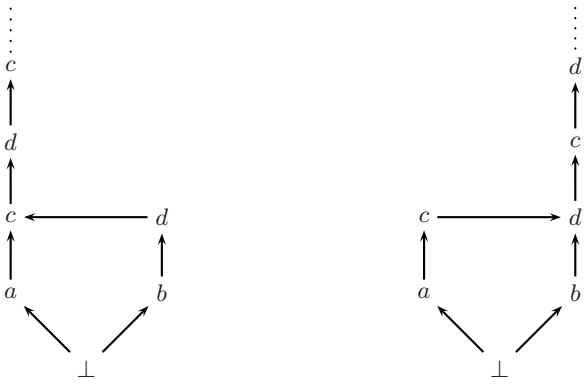

Fig. 1. Indistinguishable traces

Proof. The two pointed transition systems are bisimilar. Proposition follows from the fact that no μ-calculus formula can distinguish between two bisimilar pointed transition systems. □

The two traces from the proposition are distinguishable by a first-order formula because in the left trace first d comes before first c and it is the opposite in the right trace.

The above example implies that one needs to add something to the μ-calculus in order to be able to say more about the structure of a trace. One of the possible extensions is that with concurrency information.

Definition 4. *Let $\mathcal{M}(G) = \langle E, R, \rho\rangle$ be a transition system representation of a trace G. Relation $co \subseteq E \times E$, called the* concurrency relation*, is defined by: $co(e, e')$ iff neither $R^*(e, e')$ nor $R^*(e', e)$ hold.*

Definition 5. *Let $\mathcal{M}(G) = \langle E, R, \rho\rangle$ be a transition system representation of a trace G. We define an enriched representation $\mathcal{M}_{co}(G) = \langle E, R, \rho_{co}\rangle$ by adding new propositions co_a for $a \in Act$ and setting $\rho_{co}(co_a) = \{e : \exists e' \in \lambda(p_a).\ co(e, e')\}$. (Naturally ρ and ρ_{co} coincide on all other propositions.)*

Intuitively a co_a proposition holds in an event if there is a concurrent event labeled by a. This way a μ-calculus formula can get some information about what happens in parallel to the current event. Observe that co relation is definable in MSOL. The following theorem says that $\mathcal{M}_{co}$ representations are reach enough to get the equivalence between the μ-calculus and MSOL.

Theorem 4 ([11]). *For every MSOL formula φ there is a μ-calculus formula α_φ such that for every trace G $\mathcal{M}(G) \vDash \varphi$ iff $\mathcal{M}_{co}(G) \vDash \alpha_\varphi$.*

4 Real-Time Languages

By a *timed word* over Σ we mean a finite sequence

$$w = (a_1, t_1)(a_2, t_2)\dots(a_n, t_n)$$

of pairs from $\Sigma \times \mathbb{R}_+$. Each t_i describes the amount of time that passed between reading a_{i-1} and a_i, i.e., a_1 was read at time t_1, a_2 was read at time t_1+t_2, and so on. Note that we consider only finite timed words.

The presence of a dense domain makes a big difference and adoption of the tools described in previous sections proceeds with some difficulties. The standard automata model for real-time setting [1] is not closed under complement. First-order logic with monadic predicates and $+1$ relation is undecidable over real line. So it the MSOL theory of the real line (without $+1$ predicate) [9]. These negative results indicate that some extra care is need to obtain decidable formalisms with good expressive power.

Actually, it is quite difficult to get a formalism with a reasonable expressive power and closed under boolean operations. One such formalism is event-clock automata [2]. These are a special variant of timed automata where the reset operation on clocks is restricted. Here, we would like to present a different approach. It is well known that the μ-calculus over transition systems is equivalent to alternating automata and that there are very direct translations between the two formalism. Thus we would like to find an alternating automata model for timed-words. The first attempt would be to generalize the standard notion of timed automata [1] by introducing alternation. This is bound to fail as the universality problem for timed automata is undecidable and, in consequence, we would get a model with undecidable emptiness problem.

A solution is to restrict to automata with one clock [7]. With one clock alternating automata can still recognize languages not recognizable by nondeterministic automata with many clocks and moreover they have decidable emptiness problem. Below we define alternating timed automata in full generality and then state the results for the one-clock version.

For a given finite set $\mathcal{C}$ of *clock variables* (or *clocks* in short), consider the set $\Phi(\mathcal{C})$ of clock constraints σ defined by

$$\sigma \quad ::= \quad x < c \;\mid\; x \leq c \;\mid\; \sigma_1 \wedge \sigma_2 \;\mid\; \neg\sigma,$$

where c stands for an arbitrary nonnegative integer constant, and $x \in \mathcal{C}$. For instance, note that tt (always true), or $x = c$, can be defined as abbreviations. Each constraint σ denotes a subset $[\sigma]$ of $(\mathbb{R}_+)^{\mathcal{C}}$, in a natural way, where $\mathbb{R}_+$ stands for the set of nonnegative reals.

Transition relation of a timed automaton [1] is usually defined by a finite set of rules δ of the form

$$\delta \subseteq Q \times \Sigma \times \Phi(\mathcal{C}) \times Q \times \mathcal{P}(\mathcal{C}),$$

where Q is a set of *locations* (control states) and Σ is an input alphabet. A rule $\langle q, a, \sigma, q', r\rangle \in \delta$ means, roughly, that when in a location q, if the next input letter is a and the constraint σ is satisfied by the current valuation of clock variables then the next location can be q' and the clocks in r should be reset to 0. Our definition below uses an easy observation, that the relation δ can be suitably rearranged into a finite partial function

$$Q \times \Sigma \times \Phi(\mathcal{C}) \dot{\to} \mathcal{P}(Q \times \mathcal{P}(\mathcal{C})).$$

The definition below comes naturally when one thinks of an element of the codomain as a disjunction of a finite number of pairs (q, r). Let $\mathcal{B}^+(X)$ denote the set of all positive boolean formulas over the set X of propositions, i.e., the set generated by:

$$\phi \quad ::= \quad X \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2.$$

Definition 6 (Alternating timed automaton). *An* alternating timed automaton *is a tuple* $\mathcal{A} = (Q, q_0, \Sigma, \mathcal{C}, F, \delta)$ *where:* Q *is a finite set of locations,* Σ *is a finite input alphabet,* $\mathcal{C}$ *is a finite set of clock variables, and* $\delta : Q \times \Sigma \times \Phi(\mathcal{C}) \dot{\to} \mathcal{B}^+(Q \times \mathcal{P}(\mathcal{C}))$ *is a finite partial function. Moreover* $q_0 \in Q$ *is an* initial state *and* $F \subseteq Q$ *is a set of* accepting states*. We also put an additional restriction:*

(Partition) *For every* q *and* a*, the set* $\{[\sigma] : \delta(q, a, \sigma)$ *is defined*$\}$ *gives a (finite) partition of* $(\mathbb{R}_+)^{\mathcal{C}}$*.*

The (Partition) condition does not limit the expressive power of automata. We impose it because it permits to give a nice symmetric semantic for the automata as explained below. We will often write rules of the automaton in a form: $q, a, \sigma \mapsto b$.

To define an execution of an automaton, we will need two operations on valuations $\mathbf{v} \in (\mathbb{R}_+)^{\mathcal{C}}$. A valuation $\mathbf{v}{+}t$, for $t \in \mathbb{R}_+$, is obtained from $\mathbf{v}$ by augmenting value of each clock by t. A valuation $\mathbf{v}[r := 0]$, for $r \subseteq \mathcal{C}$, is obtained by reseting values of all clocks in r to zero.

In order to define acceptance, for an alternating timed automaton $\mathcal{A}$ and a timed word $w = (a_1, t_1)(a_2, t_2) \ldots (a_n, t_n)$, we define the *acceptance game* $G_{\mathcal{A},w}$ between two players Adam and Eve. Intuitively, the objective of Eve is to accept w, while the aim of Adam is the opposite. A play starts at the initial configuration $(q_0, \mathbf{v}_0)$, where $\mathbf{v}_0 : \mathcal{C} \to \mathbb{R}_+$ is a valuation assigning 0 to each clock variable. It consists of n phases. The $(k{+}1)$-th phase starts in $(q_k, \mathbf{v}_k)$, ends in some configuration $(q_{k+1}, \mathbf{v}_{k+1})$ and proceeds as follows. Let $\overline{\mathbf{v}} := \mathbf{v}_k + t_{k+1}$. Let σ be the unique constraint such that $\overline{\mathbf{v}}$ satisfies σ and $b = \delta(q_k, a_{k+1}, \sigma)$ is defined. Now, the outcome of the phase is determined by the formula b. There are three cases:

- $b = b_1 \wedge b_2$: Adam chooses one of subformulas b_1, b_2 and the play continues with b replaced by the chosen subformula;
- $b = b_1 \vee b_2$: dually, Eve chooses one of subformulas;
- $b = (q, r) \in Q \times \mathcal{P}(\mathcal{C})$: the phase ends with the result $(q_{k+1}, \mathbf{v}_{k+1}) := (q, \overline{\mathbf{v}}[r := 0])$. A new phase is starting from this configuration if $k{+}1 < n$.

The winner is Eve if q_n is accepting ($q_n \in F$), otherwise Adam wins.

Definition 7 (Acceptance). *The automaton* $\mathcal{A}$ accepts w *iff Eve has a winning strategy in the game* $G_{\mathcal{A},w}$*. By* $L(\mathcal{A})$ *we denote the language of all timed words* w *accepted by* $\mathcal{A}$*.*

To show the power of alternation we give an example of an automaton for a language not recognizable by standard (i.e. nondeterministic) timed automata (cf. [1]).

Example 1. Consider a language consisting of timed words w over a singleton alphabet $\{a\}$ that contain no pair of letters such that one of them is precisely one time unit later than the other. The alternating automaton for this language has three states q_0, q_1, q_2. State q_0 is initial. The automaton has a single clock x and the following transition rules:

$$q_0, a, tt \;\mapsto\; (q_0, \emptyset) \wedge (q_1, \{x\}) \qquad q_1, a, x{\neq}1 \;\mapsto\; (q_1, \emptyset)$$
$$q_1, a, x{=}1 \;\mapsto\; (q_2, \emptyset) \qquad q_2, a, tt \;\mapsto\; (q_2, \emptyset)$$

States q_0 and q_1 are accepting. Clearly, Adam has a strategy to reach q_2 iff the word is not in the language, i.e., some letter is one time unit after some other.

As one expects, we have the following:

Proposition 2. *The class of languages accepted by alternating timed automata is effectively closed under all boolean operations: union, intersection and complementation. These operations to do not increase the number of clocks of the automaton.*

The closure under conjunction and disjunction is straightforward since we permit positive boolean expressions as values of the transition function. Due to the condition (Partition) the automaton for the complement is obtained by exchanging conjunctions with disjunctions in all transitions and exchanging accepting states with non-accepting states.

An alternating timed automaton $\mathcal{A}$ is called *purely existential* if no conjunction appears in the transition rues δ. It is obvious that every purely-existential automaton is a standard nondeterministic timed automaton. The converse requires a proof because of the (Partition) condition.

Proposition 3. *Every standard nondeterministic automaton is equivalent to a purely-existential automaton.*

The automaton from Example 1 uses only one clock. This shows that one clock alternating automata can recognize some languages not recognizable by nondeterministic automata with many clocks. The converse is also true. It is enough to consider the language consisting of the words containing an appearance of a letter b at times t_1, t_2, for some $0{<}t_1{<}t_2{<}1$, and such that there is no b at time between t_1 and t_2 while there is precisely one b between $t_1{+}1$ and $t_2{+}1$.

Theorem 5 ([7]). *The emptiness problem is decidable for one-clock alternating timed automata.*

Thanks to this theorem we have a decidable formalism of timed-words that is closed under boolean operations. One can think of developing a variant of the

μ-calculus which would correspond to these automata. It should not be too difficult given that the translation between alternating automata and the μ-calculus works in many other settings. The main obstacles in this line of development are rather negative complexity results. The complexity of the emptiness problem for one-clock alternating automata is not primitively recursive [7]. Moreover, if we consider infinite words and alternating Büchi timed automata with one clock then the problem becomes undecidable.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. R. Alur, L. Fix, and T. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 204, 1997.
3. Andreé Arnold and Damian Niwiñski. *The Rudiments of the Mu-Calculus*, volume 146 of *Studies in Logic*. North-Holand, 2001.
4. Werner Ebinger. Logical definability of trace languages. In Volker Diekert and Grzegorz Rozenberg, editors, *The Book of Traces*, pages 382–390. World Scientific, 1995.
5. Paul Gastin and Antoine Petit. Infninite traces. In V. Diekert and G. Rozenberg, editors, *The Book of Traces*. World Scientific, 1995.
6. David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277, 1996.
7. Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. In *FOSSACS'05*, Lecture Notes in Computer Science, 2005. To appear.
8. Damian Niwiński. Fixed points vs. infinite generation. In *LICS '88*, pages 402–409, 1988.
9. Saharon Shelah. The monadic second order theory of order. *Annals of Mathematics*, 102:379–419, 1975.
10. Colin Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.
11. Igor Walukiewicz. Local logics for traces. *Journal of Automata, Languages and Combinatorics*, 7(2):259–290, 2002.