Time and Alternation: An Automata Based Framework to Software Model Checking

Abdelaziz Fellah
University of Sharjah
Dept. of Computer Science
Sharjah, United Arab Emirates
fellah@sharjah.ac.ae

ABSTRACT

In this paper, we present a class of powerful canonical timed alternating automata and a formalism for describing timed linear temporal logic to software model checking. Time and alternation, these two "metaphors" have dominated automata theory research in recent years. For real-time systems, it is important to augment untimed and asynchronous models of computation with the notion of time. Nevertheless, alternation is a powerful parallelism feature that has the potential to improve and reduce the state-space explosion problem in building large software model checking systems. We show that the dual connection between timed automata-theoretical and propositional-logic frameworks support and model software specifications. This can be established through languages over infinite timed words; and has a direct impact for expressing logical aspects and properties of model checking software systems.

Categories and Subject Descriptors

F.4.3 [**Theory of Computation**]: Formal languages, automata; D.1.7 [**Software**]: Model checking, abstract interpretation, model-based testing, verification-based testing.

Keywords

Timed alternating and Buchi automata, software model checking, software verification and testing, timed temporal propositional logic.

1. INTRODUCTION

Finite state automata, a well-known abstraction and specification formalism, have provided a useful framework for studying a numerous variety of problems in computer science. Furthermore, they are widely used in software systems and become a powerful canonical model for describing, modeling and verifying a range of real-world applications. A direction that has been of particular interest is the application of finite state automata theory to software model checking [2, 6, 17], an emerging practical tool for verifying programs' correctness and a light-weight version of formal methods for software development. In recent years, much research has been devoted to the dual connection between model checking and automata theory, leading to significant new results and major research directions in software engineering. Finite state automata, model checking, and temporal logic have not only been used as a formal verification framework but also as a debugging and testing software tool in practice. Thus, model checking and automata theory have evolved from traditional and theoretical paradigms to powerful and practical tools which have found applications in the industry. For example, VeriSoft [21], UPPAAL [16], and SPIN [3]. The application of automata and model checking theory to software systems may give rise to certain problems and "anomalies", potentially to the state-space explosion problem.

The requirements of a classical model checking are the specifications of the system and a set of properties, which are expressed as formulas of *temporal logic*. This was first introduced in [4] and has proved to be a powerful tool for specifying and verifying computer programs. Methods for specifying and verifying programs have been formalized in terms of temporal logic properties, geared toward a fully algorithmic approach of low computational complexity as compared to program verification, formal methods, and theorem proving. *Propositional temporal logic* is one of the most widely used design and well-known specification language [1, 4, 5, 20, 22] for modeling the properties of reactive and real systems. Different logic platforms use different time structures, either *linear* or *branching*, have been studied extensively in the literature [4, 5, 22]. *Linear-time temporal logic* (LTL) and *branching-time* (CTL) formulas are the most common approaches used as verification tools. The semantics of a propositional temporal logic is defined with respect to a labeled transition system (*i.e.,* finite labeled Kriple structure). In simple terms, the model checking can be stated as follows: "given a finite structure M and a formula φ in a propositional temporal logic, is M a model of φ"?

Model checking evolves around two important automata-theoretical concepts - *nondeterminism and alternation* on infinite words. In general, LTL and CTL formulas are often translated into different variety of automata, in particular, *nondeterministic Buchi automata* (NBA) and *alternating Buchi automata* (ABA). Based on the semantic of time,

SAC'10 March 22-26, 2010, Sierre, Switzerland.

we can classify real-time logics as discrete-time or dense-time temporal logics. The semantic of the former formalism involves the time domain $\mathbb{N}$ rather than $\mathbb{R}_{\geq 0}$ as in the later formalism. Discrete-time temporal logics are less expressive than dense-time logics but they allow succinctness in formulas. However, the latter formalism where time is variable growing continuously and uniformly in $\mathbb{R}_{\geq 0}$ are very suitable to express real-time monitoring requirements.

For branching-time temporal logics, the automata-theoretic counterpart are automata on infinite trees; and a well-known automata framework for model checking algorithms is *alternating finite automata* (AFA) which have been throughly investigated in research, for example, [12, 14, 15, 21]. Alternation, a parallel and powerful concept for formalizing parallelism, it can be viewed as a natural generalization of nondeterminism and AFA have a clear advantage over other types of automata because mainly they are exponentially more succinct than nondeterminism finite automata and double more succinct than deterministic finite automata. AFA on infinite words also know as *alternating Buchi automata* (ABA) or *alternating ω-automata*. A major direction of research on timed automata has extensively targeted real-time computational systems and real-time model checking. See for example, [7, 13, 14, 19].

Unlike automata on finite words, determinization and complementation properties of Buchi automata are nontrivial and difficult operations [8, 9]. But, they are of particular interest in formal verification, checking programs' correctness, complexity theory, and consequently in software model checking. Even though both operations can be avoided in automata-theoretic model checking; they are really useful in developing optimization techniques that are based on language containment, checking and verifying programs' correctness. The complementation construction of an n-state Buchi automata has been significantly improved from $2^{2^{O(n)}}$ to $2^{O(n)}$ [10] and then to $2^{O(n \log n)}$ [11].

In this paper, we add the concept of time to temporal logic in the same way we added time to automata. We introduce timed state-event automata and linear temporal logic. Both concepts extend the untimed versions of traditional timed automata and LTL. Then, we investigate the dual connection between model checking and automata via the powerful features of alternation and time. In addition, we show that timed alternating Buchi automata and timed state-event LTL support the specifications of a system altogether with a set of properties. More importantly, timed alternating automata are linear in the size of the specification and have a clear advantage over other types of automata because they are mainly exponential more succinct than nondeterministic finite automata and double more succinct than deterministic finite automata.

2. TIMED LABELED TRANSITION SYSTEMS

We denote by $\mathbb{R}_{\geq 0}$ and $\mathbb{N}$ the set of all non-negative reals including 0 and the set of positive natural numbers, respectively. A *timed word (event)* w_t over an input *alphabet* of events Σ, is finite (or infinite) sequence over $\Sigma \times \mathbb{R}_{\geq 0}$ of the form $\rho = (e_0, t_0)(e_1, t_1) \cdots$ where the $e_i's$ are event symbols of Σ and the $t_i's$ are in $\mathbb{R}_{\geq 0}$, representing the *time elapsed* with respect to the $e_i's$ since the previous symbol reading. The time t_1 can be thought of representing the amount of time that has elapsed since the starting of time ($t_0 = 0$). The $t_i's$ satisfies the (*monotonicity*) and (*progressiveness*) conditions That is, for all $i \geq 1$, $t_i < t_{i+1}$ and for each $t \in \mathbb{R}_{\geq 0}$ there exists $t_i, i \in \mathbb{N}$ such that $t < t_i$. We denote by Σ_t^* and Σ_t^ω the set of finite and infinite timed words over Σ, respectively.

Let $\mathcal{X}$ be a set of finite clock variables, a *clock constraint* ϕ over $\mathcal{X}$ on a given event symbol $e \in \Sigma$ can be generated by the following grammar:

$$\phi := x \bowtie c \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2$$

where $x \in \mathcal{X}$, $c \in \mathbb{N}$ such that $c \geq 0$, and $\bowtie \in \{=, <, >, \leq, \geq\}$. The operators $\vee$ and $\wedge$ stands for the logical-*or* and logical-*and*, respectively. The set of clock constraints over $\mathcal{X}$ is denoted by $\Phi(\mathcal{X})$. A *clock interpretation (valuation)* ν for $\mathcal{X}$ is a mapping from $\mathcal{X}$ to $\mathbb{R}_{\geq 0}$ (*i.e.,* ν assigns to each clock $x \in \mathcal{X}$ the value $\nu(x)$). A clock interpretation represents the values of all clocks in $\mathcal{X}$ at a given snapshot in time.

A timed labeled transition system (TLTS) (*i.e.,* finite labeled Kriple structure augmented with time) is a seven-tuple $M = (Q, Q_0, \mathcal{P}, \delta_s, \Sigma, \mathcal{X}, \delta_e)$ where Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states of M, $\mathcal{P}$ is a finite set of atomic propositions over the Boolean domain. $\delta_s : Q \to 2^{\mathcal{P}}$ is a *state-labeling function* which maps a state $q \in Q$ to a set of atomic propositions. Σ is the finite set of event symbols and $\mathcal{X}$ the set of clock variables. $\delta_e \subseteq Q \times \Sigma \times Q \times \Phi(\mathcal{X}) \times 2^{\mathcal{X}}$ is an *event-transition function* which assigns to every transition, from state q to state q', an event, a clock constraint and a set of clocks to be reset. An edge $(q, e, q', \phi, \mathcal{X}_0)$ represents a transition from state q to state q' on event e. The clock constraint ϕ specifies when the transition is enabled, and the set $\mathcal{X}_0 \in \mathcal{X}$ gives the set of clocks to be reset. Note that both states and transitions are labeled. States are labeled with a set of atomic propositions and transitions are labeled with events e, clock constraints ϕ, and a set of clocks to be reset $\mathcal{X}_0$. A *timed-state* is denoted by a pair (s, τ) where $s \subset \mathcal{P}$, $\tau \in \mathbb{R}_{\geq 0}$. That is, the atomic propositions in s are true at time τ.

DEFINITION 2.1. *A Timed* Buchi *state-event automaton (TSEA) is $A = (Q, Q_0, \mathcal{P}, \delta_s, \Sigma, \mathcal{X}, \delta_e, F)$ where*

(i) Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states of M,
(ii) $\mathcal{P}$ is a finite set of atomic state propositions over the Boolean domain,
(iii) $\delta_s : Q \to 2^{\mathcal{P}}$ is a state-labeling function,
(iv) Σ is the finite set of event symbols,
(v) $\mathcal{X}$ the set of clock variables,
(vi) $\delta_e \subseteq Q \times \Sigma \times Q \times \Phi(\mathcal{X}) \times 2^{\mathcal{X}}$ is an event-transition function,
(vii) $F \subseteq Q$ is a set of accepting state.

We label states with atomic propositions $\mathcal{P}$ instead of invariants. For example, in the timed Buchi state-event automaton of Figure 1, $\mathcal{P} = \{p, q, r\}$, the initial state is labeled with $\{p, q\}$ and the final state is labeled with p.

The timing constraints expressed in this automaton is that there is a delay between the occurrences of the different events a, b and c. The delay between c and the following b is always between one and two units of time with respect to the clock y and independently of x. The delay between b and the following a is two units of time with respect to

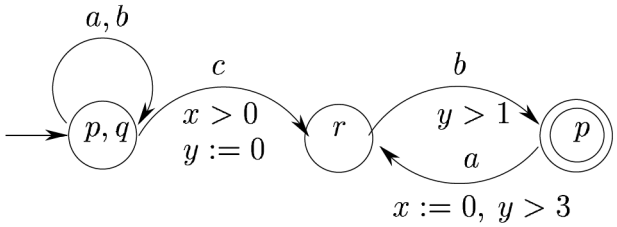

Figure 1: A timed Büchi state-event automaton.

the clock y with resetting x. Some of the edges are not annoted with clocked unconstrained in the automaton (*i.e.,* the clocks in the initial state, namely $x \geq 0$ and $y \geq 0$, x and $y \in \mathcal{X}$).

A *run* (or *path*) π of A is a (possibly infinite) sequence of pairs (q, ν), where $q \in Q$ and ν is a clock interpretation, written as:

$$\pi = (q_\pi, \nu_\pi) = (q_0, \nu_0) \xrightarrow{t_0, \delta_0} (q_1, \nu_1) \xrightarrow{t_1, \delta_1} \cdots$$

where $t_i \in \mathbb{R}_{\geq 0}$, $\delta_i = (q_i, e_i, q_{i+1}, \phi_i, \mathcal{X}_0^i) \in \delta_e$, and $q_i, q_{i+1} \in Q$, $e_i \in \Sigma$, $\phi_i \in \phi$, $\mathcal{X}_0^i \in \mathcal{X}_0$, for $i = 0, 1 \cdots$. Given an TLTS M, we denote $\Pi(M)$ the set of infinite paths of M, starting at Q_0. Let A be a timed Buchi state-event automaton such that $\mathcal{P} = \emptyset$ and $inf(\pi)$ denote the set of $q \in Q$ occurring infinitely often in π (assuming a run is an infinite path). A word w_t is accepted by A if it has a run π such that $inf(\pi) \cap F \neq \emptyset$. An infinite run of A is accepting if there is some accepting states that appear infinitely often in π. A word $w_t \in \Sigma_t^\omega$ is accepted by A if A has an accepting run on w_t. The set of words accepted by A forms the language of A, denoted by $\mathcal{L}(A) = \{w_t \in \Sigma_t^\omega \mid A \text{ accepts } w_t\}$.

3. TIMED STATE-EVENT LINEAR TEMPORAL LOGIC

Let $\mathcal{P} = p_1, \ldots, p_n$ be a nonempty set of atomic propositions, and where $p \in \mathcal{P}$. LTL formulas φ over $\mathcal{P}$ are composed of atomic propositions, the Boolean operators conjunction ($\vee$), disjunction ($\wedge$), and negation ($\neg$), the unary temporal operators X (*Next*), F (*eventually*), G (*always*), R (*release*), and the binary temporal operator U (*Until*). In simple terms, the model checking can be stated as follows: "given a finite structure M and a formula φ in a propositional temporal logic, is M a model of φ"? In semantic terms, temporal logics are defined over structures and stated as $\langle M, q \rangle \models \varphi$ to indicate that the formula φ is satisfied at state q in structure M.

Timed LTL is an extension of LTL with real time constructs for specifying time dependent logical properties of a system. The most widely known real-time extension of LTL is *Metric Temporal Logic* (MTL) in which time-constrained versions of the operator U has been expressed. Furthermore, other temporal operators can be also redefined using standard conventions to enable the description of time-dependent events, in particular G and F. For instance, $F_{\mathrm{I}}\phi \equiv \top U_{\mathrm{I}}\phi$ and $G_{\mathrm{I}}\phi \equiv \neg F_{\mathrm{I}}\neg\phi$ where F_{I} and G_{I} are the time-constrained versions of the operators F and G, respectively. $\top$ and $\bot$ indicate the true and false propositions, respectively. I denotes the time interval $[a, b]$. Two commonly semantic domains are adopted in MTL semantic domains are adopted, namely, *pointwise semantics* and *continuous semantics*. Time is expressed as the set of natural numbers $\mathbb{N}$ and the set of nonnegative real numbers $\mathbb{R}_{\geq 0}$ in the former and later formalisms. The main drawback is the logic in continuous semantics is undecidable [7] though a restriction on the semantics had lead to versions of MTL which has been shown to be decidable. In addition, MTL in pointwise semantics have been shown to be decidable [1]. Whereas, continuous semantics are more expressive than pointwise semantics in specifying the behavior of real-time systems where time constraints on events and system response time are considered to assert a formula property. For example, $F_{\leq c}\, p$ means the property p holds at some future time within the next c time units, and $p\, U_{\leq 5}\, q$ means p holds until some future time, at most 5 units in future, where q must hold. Moreover, the pointwise timed version of the classical operator U can be expressed as U_I where I denotes the time interval $[a, b]$.

In this paper, we consider dense linear temporal logic class where time is a continuous variable $\in \mathbb{R}_{\geq 0}$ and extend LTL with timed versions of temporal logic operators. Since the main drawback in the logic of continuous semantics is the undecidability, we restrict this logic by assuming that there is a finite number of discontinuities on a closed interval. Define a function $f[l, u] \mapsto S$ which is bounded with a finite number of discontinuities on a bounded interval I and a set S. We define the abstract syntax of *timed state-event linear temporal logic* ($T_s^e LTL$) over the set of atomic proposition $\mathcal{P}$ and the alphabet Σ as follows:

$$\varphi := p \mid e \mid \varphi\ U_I\ \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid x \bowtie c \mid \mathit{reset}\ x$$

where $x \in \mathcal{X}$, $c \in \mathbb{N}$ such that $c \geq 0$, $\bowtie \in \{=, <, >, \leq, \geq\}$, $p \in \mathcal{P}$, and $e \in \Sigma$. I is an interval of the form $[l, u], [l, u), (l, u)$ or $(l, u]$ where $u, l \in \mathbb{R}_{\geq 0}$ and $l \leq u$. The right-end bounded intervals may possibly be ∞. The next operator, X, has no meaning in the continuous context. $\langle M, q \rangle \models \varphi$ indicates that the formula φ is satisfied at state q in the structure M. We write $M \models \varphi$ if and only if, for every infinite path $\pi \in \Pi(M)$, $\pi \models \phi$.

PROPOSITION 3.1. *Let $M = (Q, Q_0, \mathcal{P}, \delta_s, \Sigma, \mathcal{X}, \delta_e)$ and $M' = (Q', Q_0', \mathcal{P}', \delta_s', \Sigma', \mathcal{X}', \delta_e')$ be two TLTS structures. M' is an abstraction of M if and only if (i) $\mathcal{P}' \subseteq \mathcal{P}$, (ii) $\Sigma' = \Sigma$, (iii) $\mathcal{X} = \mathcal{X}'$, (v) For every $\pi \in \Pi(M)$ there exists a path $\pi' \in \Pi(M')$.*

4. INFINITE ALTERNATION

Automata that read infinite sequences are usually referred to as ω-automata; and the simplest and typical class of automata on infinite words are usually called nondeterministic Buchi automaton (NBA) or simply *Buchi automata*. In the paper, we are interested in the time version of NBA.

DEFINITION 4.1. *A timed* Buchi *automaton (TBA) over infinite words is a tuple $A = (Q, Q_0, \mathcal{P}, \Sigma, \mathcal{X}, \delta, F)$ where Q is the finite nonempty set of states, $Q_0 \subseteq Q$ is the nonempty set of initial states, $\mathcal{P}$ is a finite set of atomic propositions over the Boolean domain. Σ is the finite set of event symbols and $\mathcal{X}$ the set of clock variables. $F \subseteq Q$ is a finite accepting states. $\delta \subseteq Q \times \Sigma \times \Phi(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ is the transition function.*

Without loss of generality, we assume that $Q_0 = q_0$, single starting state. If $|Q_0| = 1$ and for $q \in Q$ and $a \in \Sigma$, we have $|\delta(q, a)| = 1$, then A is a deterministic Buchi automaton (DBA). Given a set Q, we define $\mathcal{B}^+(Q)$ as the set of all

positive formulas "true" and "false" over the set Q, all elements $q \in Q$, and Boolean combinations over Q built with $\vee$ and $\wedge$. An *alternating Buchi automaton* (ABA) over an alphabet is defined similar to an NBA, but additionally Q is partitioned into two sets, $\mathbb{E}$ and $\mathbb{U}$ of *existential* and *universal* states, respectively. Acceptance of alternating Buchi automata (ABA) is best defined via games. A run of an ABA is a (potentially infinite) tree of alternation of $\wedge$'s and $\vee$'s. We define a *run* of an ABA on an infinite word $w = w_0 w_1 \cdots \in \Sigma^\omega$. For a state $q \in Q$ of M, trees rooted at q are infinite trees, denoted by $T_{(q)}$. The nodes of $T_{(q)}$ are the finite runs starting from q and ordered by their suffix. Thus, $T(M) = \{T(q) \mid q \in Q_0\}$ is the set of *computation trees* of M. Model checking considers finite structures displaying infinite alternating behaviors.

DEFINITION 4.2. *An alternating Buchi automaton (ABA) is a tuple $A = (\Sigma, Q, q_0, \delta, F)$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, and $F \subseteq Q$ is a set of accepting states.*

5. AUTOMATA-BASED SOFTWARE MODEL CHECKING

There is a strong relationship between automata and logic. The relation between propositional temporal logic and automata can be established through languages over infinite words. Timed state-event linear temporal logic $T_s^e LTL$ formula over a finite timed state-event automaton (TSEA) on infinite words should accept all computations (runs) that satisfy the formula. The timed Buchi state-event automaton approach to model checking can be stated in the following theorem:

THEOREM 5.1. *Let $A_M = (Q, Q_0, \mathcal{P}, \delta_s, \Sigma, \mathcal{X}, \delta_e)$ be a timed labeled transition system describing the system M and φ an $T_s^e LTL$ formula. Then there exists a timed* Buchi *state-event automaton $A_\varphi = (Q, Q_0, \mathcal{P}, \delta_s, \Sigma, \mathcal{X}, \delta_e, F)$ such that $\mathcal{L}(A_M) \subseteq \mathcal{L}(A_\varphi)$, where $\Sigma = 2^{\mathcal{P}}$ and $|Q| \leq 2^{O(|\varphi|)}$. Moreover, M satisfies φ if and only if $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi}) = \emptyset$.*

Proof. The above close relationship transforms a verification problem to an automata-theoretic problem, and described as a language containment problem. The idea of the proof is described as follows: (*i*) A_M can be viewed as an ω-automata whose language, $\mathcal{L}(A_M)$, is the set of all possible computations of A_M starting at $q_0 \in Q_0$. (*ii*) Construct an ω-automata $A_{\neg\varphi}$ whose language is $\mathcal{L}(A_{\neg\varphi})$ (*i.e.*, $\mathcal{L}(\overline{A_\varphi})$ which invalidates φ. (*iii*) Construct the *cross product* of A_M and the negated automaton $A_{\neg\varphi}$, $(A_M \times A_{\neg\varphi})$, whose language is $\mathcal{L}(A_M \times A_{\neg\varphi})$ is the set of computations of A_M that invalidate φ. (*iv*) Finally, check whether $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi}) = \emptyset$. The property holds in A_M if and only if $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi}) = \emptyset$ (*i.e.*, $\mathcal{L}(A_M \times A_{\neg\varphi}) = \emptyset$). Thus, the model checking problem is reduced to checking whether the language of the product automaton is empty. This is refereed to as the emptiness problem on infinite words. That is, whether there is an algorithm that tells whether at least one accepting run exits. If the language is non-empty, we can find an infinite word that will provide an example of a run that is valid in the model A_M, but invalid with respect to the property φ.

Alternating Buchi automata are more succinct that any deterministic and nondeterminate Buchi automata. While the translation from linear temporal logic formulas to nondeterministic Buchi automata is exponential, the translation to alternating Buchi automata is linear. Along the same lines of correspondence between LTL and Buchi automata, we state the following:

THEOREM 5.2. *Let $A_M = (Q, Q_0, \mathcal{P}, \delta_s, \Sigma, \mathcal{X}, \delta_e)$ be an untimed labeled transition system describing the system A_M and φ an LTL formula. Then there exists an alternating Buchi automaton $A_\varphi = (\Sigma, Q, Q_0, \delta, F)$ such that $\mathcal{L}(A_M) = \mathcal{L}(A_\varphi)$, where $\Sigma = 2^{\mathcal{P}}$ and $|Q| = O(|\varphi|)$.*

6. TIMED ALTERNATING BÜCHI AUTOMATA

We denote by the symbol $\mathcal{B}$ the two-element Boolean algebra $\mathcal{B} = (\{0,1\}, \vee, \wedge, ^-, 0, 1)$.

$\mathbb{B}^Q$ is a vector with $|Q|$ elements referring to all the Boolean functions from Q to $\mathcal{B}$, and $\mathbb{R}^{\mathcal{X}}_{\geq 0}$ is a vector with $|\mathcal{X}|$ elements (all non-negative) which refers to all real functions from $\mathcal{X}$ to $\mathbb{R}_{\geq 0}$. We define a timed alternating Buchi automaton as follows:

DEFINITION 6.1. *A timed alternating Büchi automaton (TABA) is a seven-tuple $A = (Q, q_0, \Sigma, \mathcal{X}, \delta, F)$ where Q is a finite set of states, Σ is a finite input alphabet, $\mathcal{X}$ is a finite set of clock variables, and $\delta : (Q \times \Sigma \times \mathbb{R}_{\geq 0}) \mapsto \mathbb{B}^Q(Q \times \mathbb{P}(\mathcal{X}))$ is a finite partial function and where $\mathbb{P}$ denotes the power set.*

COROLLARY 6.1. *The class of languages accepted by timed alternating automata are closed under Boolean operations [13].*

Now, we introduce deterministic timed alternating Buchi automata.

DEFINITION 6.2. *A deterministic timed alternating* Buchi *automaton (DTABA) is a seven-tuple $A = (Q, \Sigma, S, g, h, \mathcal{X}, f)$, where*

(a) Q is a finite set, the set of states,
(b) Σ is an alphabet, the input alphabet,
(c) $S \subseteq Q$ is the set of all starting states,
(d) $\mathcal{X}$ is a finite set, the set of clocks,
(e) h is a time transition function, $h : (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}) \times (\Sigma \times \mathbb{R}_{\geq 0}) \mapsto (\mathbb{B}^Q \times (\mathbb{R}^{\mathcal{X}}_{\geq 0} \times \mathbb{R}_{\geq 0})) \times \Sigma$,
(f) g is a letter symbol transition function from Q into the set of all functions from $\Sigma \times (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$ into $\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}$, that is, $g : (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}) \times \Sigma \mapsto \mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}$,
(g) f is a time accepting function, $f : \mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0} \mapsto \mathcal{B}$.

More specifically, the function h is defined as:

$$h((q_1, q_2, \ldots, q_{|Q|}, x_1, x_2, \ldots, x_{|\mathcal{X}|}), (a, t)) =$$
$$((q_1, q_2, \ldots, q_{|Q|}, x_1 + t, x_2 + t, \ldots, x_{|\mathcal{X}|} + t), a)$$

where $q_i \in Q$ for $1 \leq i \leq |Q|$, $x_j \in \mathcal{X}$ for $1 \leq j \leq |\mathcal{X}|$, $a \in \Sigma$ and $t \in \mathbb{R}_{\geq 0}$ such that $t \geq 0$.

We extend h to the set of timed words defined as $(\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}) \times (\Sigma \times \mathbb{R}_{\geq 0})^* \mapsto (\mathbb{B}^Q \times (\mathbb{R}^{\mathcal{X}}_{\geq 0} \times \mathbb{R}_{\geq 0})) \times \Sigma$ such that:
$h(u, wa') = h(g(h(u, w)), a')$
where $u \in (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}), w \in (\Sigma \times \mathbb{R}_{\geq 0})^*$, and $a' \in \Sigma$. Also, it should be noted that $g(u, \lambda) = u$, where λ is the empty word.

DEFINITION 6.3. *Let* $A = (Q, \Sigma, S, g, h, \mathcal{X}, f)$ *be a DTABA and* $w \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ *be a timed word.* w *is accepted by* A *if and only if* $f(g(h(s, w_t))) = 1$, *where* $s \in (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$ *is the characteristic vector of* S. *Moreover, for each* s_q, $q \in Q$, $s_q = 1$ *if and only if* $q \in S$; *and for each* s_x, $s_x = 0$, *where* $x \in \mathcal{X}$.

The following examples trace the acceptance and rejection of two timed words.

EXAMPLE 6.1. *Let* $w_t = (b, 2)(a, 3)(c, 3)$

$$
\begin{aligned}
& f(g(h(s, (b,2)(a,3)(c,3)))) \\
= \; & f(g(h(g(h(s, (b,2), (a,3))), (c,3)))) \\
= \; & f(g(h(g(h(g(h(s, (b,2))), (a,3))), (c,3)))) \\
= \; & f(g(h(g(h(g(h(g(h((1,0,0,0,0,0), (b,2))), (a,3))), (c,3)))) \\
= \; & f(g(h(g(h(g(h(g((1,0,0,2,2,b), (a,3))), (c,3)))) \\
= \; & f(g(h(g(h(h((0,1,0,2,2), (a,3))), (c,3)))) \\
= \; & f(g(h(g((0,1,0,3,3), a)), (c,3)))) \\
= \; & f(g(h((1,0,0,3,0), (c,3)))) \\
= \; & f(g((1,0,0,3,0), c)) \\
= \; & f(1,0,1,0,0) = 1 \wedge \overline{0} \wedge 1 = 1
\end{aligned}
$$

Therefore w_t *is accepted.*

LEMMA 6.1. *Timed alternating* Buchi *automata are double-exponential more succinct than deterministic timed* Buchi *automata.*

LEMMA 6.2. *Timed alternating* Buchi *automata are exponentially more succinct than timed* Buchi *automata.*

LEMMA 6.3. *Deterministic timed alternating* Buchi *automata are exponentially more succinct than deterministic timed* Buchi *automata.*

Proof. Due to space limitations, some proofs are omitted in this paper. However, for complete and detailed proofs, the reader may directly contact the author.

7. CONCLUSION

In this paper, we have shown that timed alternating finite automata theoretical results and LTL model checking algorithms can be incorporated within large and complex software systems. The approach presented involves both states and events and in general timed automata techniques and results can be used for verifying timed linear logic formulas. Moreover, timed state-event automata, deterministic timed alternating Buchi automata, and timed state-even LTL formalisms facilitate software model checking and specifications. Finally, alternation has a great impact on the state-space explosion problem in the sense that timed alternating automata are double-exponentially more succinct than deterministic timed automata and exponentially more succinct that timed automata. Finally, this investigation can be further explored and improved by considering different types of timed LTL and automata.

8. REFERENCES

[1] J. Ouaknine, and J. Worrell. On the decidability of metric temporal logic. *Logic in Comp. Sc., LICS 2005. Proceedings of the* 20^{th} *Annual IEEE Symp.*, 188–197, 2005.

[2] W. Visser, K. Hevelund, G. Brat, and S. Park. Model Checking Programs. *Proceedings of the* 15^{th} *IEEE Int. Conf. on Automated Software Eng.*, 11–15, 2000.

[3] G.J. Holzmann. The Model Checker SPIN. *IEEE Trans. Soft. Eng.*, 23(5): 279-297, 1997.

[4] A. Pnueli. The Temporal Logic of Programs. *Proceedings of* 18^{th} *IEEE Symp. Foundations of Comput. Sci.*, 46–77, 1977.

[5] E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM,* 33(1): 151–178, 1986.

[6] M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, Cambridge, MA, 1999.

[7] R. Alur and D.L. Dill. A Theory of timed automata. *Theor. Comput. Sci.*, 162(2): 183–235, 1994.

[8] O. Kupferman and M. Vardi. From complementation to certification. *Theor. Comput. Sci.*, 345(1): 83–100, 2005.

[9] O. Kupferman. Avoiding determinization. *LICS,* 243–254, 2006.

[10] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem of Buchi automata with applications to temporal logic. *Theor. Comput. Sci.*, 49: 217–237, 1987

[11] S. Safra. On the complexity of ω-automata. *Proceedings of the* 29^{th} *FOCS,* 319–327, 1988.

[12] S. Yu. *Regular languages. G. Rozenberg, A. Salomaa, eds.: Handbook of Formal languages*, Vol. 1 Springer, 41–110, 1997.

[13] A. Fellah, Z. Friggstad and S. Noureddine, Deterministic timed AFA: A new class of timed alternating finite automata. *Journal of Comput. Sci.* 3(1): 1–8, 2007.

[14] A. Fellah, H. Jurgensen, and S. Yu. Constructions for alternating finite automata. *Int. Journal of Comput. Math.,* 35, 117–132, 1994.

[15] M. Slanina, H.B Sipma, and Z. Manna. Deductive verification of alternating systems. *Formal Aspects Comput.*, 20(4-5): 507–560, 2008.

[16] T. Amnell, G. Behrmann, et *al.* UPPAAL - now, next, and future. *Modeling and verification of parallel processes,* 99–124, 2001.

[17] P. Godefroid. Model checking for programming. *Languages using VeriSoft Proceedings of the* 24^{th} *ACM Symp. on Principles of Prog. Lang.,* Paris. 1997

[18] M. Roberson, M. Agnew, *et al.*. Efficient software model checking of soundness of type systems *ACM SIGPLAN notices archive*, 43(10): 493–504, 2008.

[19] S. Chaki, E Clarke, *et al.*, Concurrent software verification with states, events, and deadlocks. *Formal Aspects of Computing* 17(4), 2005.

[20] C. Fritz. Constructing Buchi automata from linear temporal logic using simulation relations for alternating Buchi automata. *CIAA*, 35–48, 2003.

[21] C. Fritz and T. Wilke Simulation relations for alternating Buchi automata. *Theor. Comput. Sci.* 338(1-3): 275–314, 2005.

[22] D.N. Jansen and R.J. Wieringa1. Extending CTL with actions and real time. *Journal of Logic and Comput.*, 12(4): 607–621, 2002.