Proceedings of the 5th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, Madrid, Spain, February 15-17, 2006 (pp97-103)

Some Succinctness Properties of Ω-DTAFA

A. FELLAH*
University of Sharjah
Department of Computer Science
Sharjah
United Arab Emirates

S. NOUREDDINE and Z. FRIGGSTAD
University of Lethbridge
Dept. of Math. & Computer Science
Lethbridge, AB, T1K 3M4
Canada

Abstract: A new type of timed alternating finite automata (TAFA) called omega deterministic timed alternating finite automata (Ω-DTAFA) is described. DTAFA and Ω-DTAFA are synchronous and parallel computational models used for modeling real-time constraints computations and developing software systems. These models are extended with a finite set of mutually exclusive real-valued clocks on events which trigger the state transitions of the automaton. We show that Ω-DTAFA are closed under all Boolean operations, including some of the most important operators. We then consider some well-known automata-theoretic properties of TAFA, and investigate those properties in the case of Ω-DTAFA. Alternation, timing and determinism add perfect features to automata expressiveness, parallelism and succinctness, which have practical applications in software and real-time systems.

Key–Words: Automata, formal languages, information theory, timed automata, timed alternating finite automata. Typing manuscripts, LaTeX

1 Introduction

Traditional finite state automata are untimed or asynchronous models of computation in which only the ordering of events, not the time at which events occur, would affect the result of a computation. Timed finite automata (TFA) have become a powerful canonical model for describing timed behaviors and an effective tool for modeling and verifying real-time computations. Timed automata received their first seminal treatment in [1], since then much work has been done in this area of research. In addition, a major direction that has been particularly successful is the application of the timed automata theoretic approach in modeling real-time systems and checking problems, and hence, have applications in the software engineering processes. Several models based on automata theory have already been implemented as an effective verification and validation tools for real-time and embedded systems, for example, research tools such as UPPAAL [14], and KRONOS [13]. Several papers in the literature have adopted finite state machines as the main formalism to model the behavior of web service system. For example, Boolean web-service automata for distributed web services have been introduced in [5] as a parallel model for interaction and interoperability between applications as well as their compositions.

Alternating finite automata (AFA) are a natural generalization of nondeterminism automata which provide a succinct representation for regular languages, but are double-exponentially more succinct than deterministic finite automata (DFA). Independently, AFA were introduced in [3] and [4] under the name of Boolean automata. Since then most of the subsequent research focused on various types of alternating machines to complexity classes, see for example, [8, 9, 10, 12].

Timed alternating finite automata (TAFA) – a class of alternating finite automata augmented with a finite set of real-valued clocks (*i.e.*, timers) were first considered in [6]. Intuitively, a timed alternating finite automaton can be viewed as a *"timed parallel finite automaton"*. Multiple clocks TAFA would be particularly useful in modeling a system that has many dependency relationship since several clocks are available that can be reset during any transition. This, combined with the fact that timing constraints can involve multiple clocks, allows complex dependent relationships to be constructed that cannot similarly be modeled by TFA since TFA do not have the power of parallelism and AFA do not have the functionality of clocks. Despite being very expressive for describing timed behaviors and modeling real-time systems, TFA and TAFA are neither determinizable nor closed under the complementation, and timed regular expressions have no negation operator [6].

In [7] and along the lines of even-clock automata [2], it has been shown that every timed AFA can be determinized since at all times during the run of an automaton, the value of each clock is determined solely

*Corresponding author: fellah@sharjah.ac.ae

by the input sequence and doesn't depend on nondeterminism. The clocks are divided into mutually exclusive sets and a predefined association between the clocks and symbols of the input alphabet define a subclass of timed AFA models called omega deterministic timed alternating finite automata (Ω-DTAFA). These automata could be comparable to event-clock automata [2] and DTAFA [7]. The main complement-property, which fails for arbitrary timed AFA, hold for all deterministic TAFA (DTAFA). That is, Ω-DTAFA are closed under all operations, in particular, for every Ω-DTAFA we can construct an Ω-DTAFA that defines the complement of a timed language.

This paper is organized as follows. Section 2 is devoted to preliminaries. Section 3 introduces omega deterministic timed alternating finite automata (Ω-DTAFA) and deterministic timed alternating finite automata (DTAFA), two variants of timed AFA extended with a finite set of restricted real-valued clocks. Section 4 shows the transformation of DTAFA and Ω-DTAFA. Both types of automata are equivalent in language recognition to Alur-Dull timed automata [1]. In section 5, we investigate some automata-theoretic properties of Ω-DTAFA. Finally, in Section 6 we draw some concluding remarks.

2 Preliminaries

We denote by $\mathbb{R}_{\geq 0}$ the set of all non-negative reals including 0. The cardinality of a finite set A is $|A|$. An *alphabet* Δ is a finite, nonempty set whose elements are called *symbols* or *letters*. Any subset of Δ^* is called a *language* over Δ. A *timed word*, w over Δ is a finite sequence $\rho = (a_1, t_1)(a_2, t_2)\cdots(a_i, t_i)$ where the $a_i's$ are symbols of Δ and the $t_i's$ are in $\mathbb{R}_{\geq 0}$ such that for all $i \geq 1$, $t_i < t_{i+1}$. The first element, $a_i's$, of each pair are the input symbols, and the second element, $t_i's$, are the *time elapsed* with respect to the $a_i's$ 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. We assume that $t_1 = 0$. Thus, $t_1 \cdots t_i$ is a finite monotonically non-decreasing time sequence of $\mathbb{R}_{\geq 0}$. A *timed trace* (run) is a finite sequence $(a_1, t_1)(a_2, t_2)\cdots(a_i, t_i)$. The *length* of a word w, denoted by $|w|$, is the total number of symbols in w, where a is a finite sequence of symbols of Δ, and t is a finite monotonically increasing sequence of $\mathbb{R}_{\geq 0}$ and both have the same length. The time language $(\Delta \times \mathbb{R}_{\geq 0})^*$ is the set of all timed words over Δ where λ denotes the empty timed word. Timed words are defined over the combination of the monoid $(\Delta, \circ, \lambda)$ and the time monoid $(\mathbb{R}_{\geq 0}, +, 0)$, where "$\circ$" is the classical concatenation operator (we write *ab* rather than $a \circ b$ for the concatenation). For any language $\mathcal{L} \subseteq \Delta^*$, $\bar{\mathcal{L}} = \Delta^* \backslash \mathcal{L}$, is the complement of $\mathcal{L}$ with respect to Δ^*. For languages $\mathcal{L}_1$ and $\mathcal{L}_2$ over Δ, the union and intersection are denoted by $\mathcal{L}_1 \cup \mathcal{L}_2$ and $\mathcal{L}_1 \cap \mathcal{L}_2$, respectively. For more details, see [1, 11].

3 Omega Deterministic Timed Alternating Finite Automata

In this section we first define a variant of timed alternating finite automata called *deterministic timed alternating finite automata* (DTAFA). Then, we extend DTAFA with a set of Boolean operators which yields the definition of *Omega deterministic timed alternating finite automata* (Ω-DTAFA).

Let $\mathcal{X}$ be a set of clock variables, a *clock constraint* ψ over $\mathcal{X}$ on a given input symbol $a \in \Delta$ can be generated by the following grammar:

$$\psi := x \leq c \mid x < c \mid c \leq x \mid c < x \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2$$

where x is any clock in $\mathcal{X}$ and $c \in \mathbb{R}_{\geq 0}$ such that $c \geq 0$. The operators $\vee$ and $\wedge$ stands for the logical-*or* and logical-*and*, respectively.

A *clock interpretation* 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.

There are some cases where we don't need explicitly to state a constraint if it spans all non-negative reals (*i.e.*, $x \leq c \vee c < x$). Since all clock interpretations for all x can never be negative, the constraint $\psi = 0 \leq x_1 \wedge 0 \leq x_2 \wedge \ldots \wedge 0 \leq x_{|\mathcal{X}|} \wedge \psi$ is implied for all $x_i \in \mathcal{X}$ where $1 \leq i \leq |\mathcal{X}|$. The assignment statement $x := 0$ implies that the clock is reset (the symbol ":=" is the assignment operator). However, the comparison statement $x = 0$ is a clock constraint that is satisfied if and only if the current interpretation of x is 0 (the symbol "=" is the comparison operator).

Definition 3.1 *A deterministic timed alternating finite automaton* (DTAFA) *is a seven-tuple* $A = (Q, \Delta, S, g, h, \mathcal{X}, f)$, *where*

(a) Q *is a finite set, the set of states,* (b) Δ *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,* $h : (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}) \times (\Delta \times \mathbb{R}_{\geq 0}) \mapsto (\mathbb{B}^Q \times (\mathbb{R}^{\mathcal{X}}_{\geq 0} \times \mathbb{R})) \times \Delta$, (f) g *is a letter symbol transition function from* Q *into the set of all functions from* $\Delta \times (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$ *into* $\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}$, *that is,* $g : (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}) \times \Delta \mapsto \mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}$, (g) f *is a time accepting function,* $f : \mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0} \mapsto \mathbb{B}$.

We denote by the symbol $\mathbb{B}$ the two-element Boolean algebra $\mathbb{B} = (\{0,1\}, \vee, \wedge, ^-, 0, 1)$. $\mathbb{B}^Q$ is a vector with $|Q|$ elements referring to all the Boolean functions from Q to $\mathbb{B}$, 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 $\mathbb{R}_{\geq 0}$. 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 $1 \leq j \leq |\mathcal{X}|$, $a \in \Delta$ and $t \in \mathbb{R}_{\geq 0}$ such that $t \geq 0$.

We extend h to the set of timed words defined as $(\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}) \times (\Delta \times \mathbb{R}_{\geq 0})^* \mapsto (\mathbb{B}^Q \times (\mathbb{R}^{\mathcal{X}}_{\geq 0} \times \mathbb{R})) \times \Delta$ such that:

$$h(u, wa') = h(g(h(u, w)), a')$$

where $u \in (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0}), w \in (\Delta \times \mathbb{R}_{\geq 0})^*$, and $a' \in \Delta$. Also, it should be noted that $g(u, \lambda) = u$, where λ is the empty word.

For each state $q \in Q$, $x \in \mathcal{X}$ and $a \in \Delta$, we define $g_q(a)$ to be the Boolean function $\mathbb{B}^Q \times \Delta \mapsto \mathbb{B}$ and $g_x(a)$ to be the function $\mathbb{R}^{\mathcal{X}}_{\geq 0} \times \Delta \mapsto \mathbb{R}_{\geq 0}$ such that

$$g_q(u')(a) = g_q(u', a)$$

and

$$g_x(v')(a) = g_x(v', a)$$

$u' \in \mathbb{B}^Q$ and $v' \in \mathbb{R}^{\mathcal{X}}_{\geq 0}$. Thus, the value of $g_q(u')(a)$ is either 1 or 0 and the value of $g_x(v')(a)$ is either 0 or v'_x, where v'_x is an element of v'.

We define $g_Q(u', a)$ to be the function $\mathbb{B}^Q \times \Delta \mapsto \mathbb{B}^Q$ by taking all the $|Q|$ functions $g_q : \mathbb{B}^Q \times \Delta \mapsto \mathbb{B}$, $q \in Q$. Similarly, we define $g_{\mathcal{X}}(v', a)$ to be the real function $\mathbb{R}^{\mathcal{X}}_{\geq 0} \times \Delta \mapsto \mathbb{R}^{\mathcal{X}}_{\geq 0}$ by taking all the $|\mathcal{X}|$ functions $g_x : \mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$, $x \in \mathcal{X}$. For notational convenience, $g_Q(u', a)$ and $g_{\mathcal{X}}(v', a)$ can be written as one mapping, $g(u, a)$, $u \in (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$ and $a \in \Delta$.

All clock constraints on a given input symbol from a given state must be mutually exclusive and must span all $\mathbb{R}^{\mathcal{X}}_{\geq 0}$. For each input symbol $a \in \Delta$, consider all clock constraints associated with this symbol. If any two constraints overlap, then partition them up into mutually exclusive constraints. The result should be a set of mutually exclusive clock constraints whose region, for each clock, spans all non-negative real numbers.

Definition 3.2 *An omega deterministic timed alternating finite automaton (Ω-DTAFA)* $A = (Q, \Delta, S, g, h, \mathcal{X}, f, \Omega)$ *is a deterministic timed alternating finite automaton (DTAFA) over extended Boolean operations, where* $Q, \Delta, S, g, h, \mathcal{X}, f$ *are all the same as in DTAFA and* Ω *is a set of extended Boolean operations such that* $\mathbb{B} \cap \Omega = \emptyset$.

In Ω-DTAFA, $\mathbb{B}^Q$ would be the set of all functions from Q into $(\mathbb{B} \cup \Omega)$. ($(\mathbb{B} \cap \Omega) = \emptyset$ is specified so an operator in $\mathbb{B}$ is not redefined). Each operator in Ω must be given by a truth table.

Definition 3.3 *Let* $A = (Q, \Delta, S, g, h, \mathcal{X}, f, \Omega)$ *be an* Ω-DTAFA *and* $w \in (\Delta \times \mathbb{R})^*$ *be a timed word.* w *is accepted by* A *iff* $f(g(h(s, w))) = 1$, *where* $s \in (\mathbb{B}^Q \cup \Omega \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$ *is the characteristic vector of* S. *For each* s_q, $q \in Q$, $s_q = 1$ *iff* $q \in S$; *and for each* s_x, $s_x = 0$, *where* $x \in \mathcal{X}$.

Example 3.1 Consider the following Ω-DTAFA $A = (Q, \Delta, S, g, h, \mathcal{X}, f, \Omega)$ where $Q = \{q_0, q_1, q_2\}$, $\Delta = \{a, b, c\}$, $S = \{q_0\}$, $\mathcal{X} = \{x, y\}$, $f(q_0, q_1, q_2, x, y) = q_0 \wedge \overline{q_1} \wedge q_2$, $\Omega = \emptyset$, and g is given by the following tables:

State	a	b $x < 1$	b $x \geq 1$	c $y = 0$	c $y > 0$
q_0	q_1	$q_1 \vee \overline{q_2}$	q_2	q_0	0
q_1	$\overline{q_0} \wedge \overline{q_1}$	$\overline{q_1} \vee q_2$	1	$q_1 \vee q_2$	0
q_2	0	q_0	q_1	q_0	q_2

State-table for Ω-DTAFA A.

clk res	a	b $x < 1$	b $x \geq 1$	c $y = 0$	c $y > 0$
x	$\overline{q_1 \wedge q_2} \cdot x$	$\overline{q_0 \vee q_1} \cdot x$	$\overline{0} \cdot x$	$\overline{0} \cdot x$	$\overline{1} \cdot x$
y	$\overline{\overline{q}_0 \wedge \overline{q}_2} \cdot y$	$\overline{0} \cdot y$	$\overline{0} \cdot y$	$\overline{0} \cdot y$	$\overline{0} \cdot y$

Clock-reset (clk res) table for Ω-DTAFA A.

The characteristic vector of S is $s = (q_0, q_1, q_2, x, y) = (1, 0, 0, 0, 0)$.

The clock-reset table simply gives the function for the u_x element of u in $g(u, a)$, where $u \in (\mathbb{B}^Q \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$, $a \in \Delta$, for all clocks $x \in \mathcal{X}$. If the Boolean value is 1, then we write $1 \cdot x = x$ (or $x \cdot 1 = x$) to indicate a no reset. The symbol "$\cdot$" is the *reset operator* and not the usual concatenation symbol. If the Boolean value is 0, then we write $0 \cdot x = 0$ (or $x \cdot 0 = 0$) to indicate a reset. In fact, the reset operator "$\cdot$" simulates the functionality of the "multiplication operation" in the sense that anything multiplied by zero is zero. In addition, if an entry of the table contains an expression that is 0, this implies the expression is $0 \cdot x$ (or $x \cdot 0$). Likewise, if an entry of the table contains

an expression that is x, this implies the expression is $1 \cdot x$ (or $x \cdot 1$). Moreover, $(q_1 \vee q_2) \cdot x$ means that x is reset if $q_1 = 0$ and $q_2 = 0$. The following example traces the non-acceptance of a timed word.

Example 3.2 Let $w = (c,1)(a,2)$

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

Therefore w is rejected.

4 From DTAFA to Ω-DTAFA

Let A^1 and A^2 be two DTAFA. Denote by $\Omega(A^1)$ and $\Omega(A^2)$ the Ω-DTAFA corresponding to the DTAFA A^1 and A^2, respectively. Thus, $\Omega(A^1) = (Q^1, \Delta^1, S^1, g^1, h^1, \mathcal{X}^1, f^1, \Omega^1)$ and $\Omega(A^2) = (Q^2, \Delta^2, S^2, g^2, h^2, \mathcal{X}^2, f^2, \Omega^2)$ where $\Omega^1 = \emptyset$ and $\Omega^1 = \emptyset$. It follows that each function with $\mathbb{B}^Q$ represents Q into $(\mathbb{B} \cup \Omega)$ for $\mathbb{B}^Q$.

Lemma 4.1 *Given two DTAFA (or Ω-DTAFA) A^1 and A^2, we can construct an Ω-DTAFA A that accepts the language $L(A) = L(A^1) \star L(A^2)$ where $\star$ is a binary operation.*

Proof.
If A^1 and A^2 are DTAFA, then convert them to Ω-DTAFA.
Assume that A^1 and A^2 are now Ω-DTAFA and $A^1 = (Q^1, \Delta^1, S^1, g^1, h^1, \mathcal{X}^1, f^1, \Omega^1)$ and $A^2 = (Q^2, \Delta^2, S^2, g^2, h^2, \mathcal{X}^2, f^2, \Omega^2)$. We can construct an Ω-DTAFA $A = (Q, \delta, S, g, h, \mathcal{X}, f, \Omega)$ accepting $L(A^1) \star L(A^2)$ using the following algorithm:

Preconditions:
$Q^1 \cap Q^2 = \emptyset$, (it follows that $S^1 \cap S^2 = \emptyset$) and $\mathcal{X}^1 \cap \mathcal{X}^2 = \emptyset$. Let *o* be a Boolean operator, if $o \in \Omega^1$ and *o* is in Ω^2, then it must be defined with the same truth table.

Algorithm:
$Q = Q^1 \cup Q^2 \cup \{q_x^1, q_x^2\}$, $\Delta = \Delta^1 \cup \Delta^2$, $S = S^1 \cup S^2$, $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$, $\Omega = \Omega^1 \cup \Omega^2 \cup \{\star\}$. The function h is defined in terms of Q, Δ and $\mathcal{X}$.

$$\begin{aligned}
& f(q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2 \ldots, q_{|Q^2|}^2, q_x^2, \\
& \quad x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2) = \\
& (f^1(q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1) \wedge \overline{q_x^1}) \star \\
& (f^2(q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2) \wedge \overline{q_x^2})
\end{aligned}$$

Let $g^1(u^1, a) = v^1, g^2(u^2, a) = v^2$, we define $g(u, a) = v$, such that:
$u^1 = (q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1)$,
$u^2 = (q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2)$ and
$u = (q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, q_x^2,$
$x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2)$
and $u^1, v^1 \in (\widehat{\mathbb{B}}^{Q^1} \times \mathbb{R}^{\mathcal{X}^1})$, $u^2, v^2 \in (\widehat{\mathbb{B}}^{Q^2} \times \mathbb{R}^{\mathcal{X}^2})$, $u, v \in (\widehat{\mathbb{B}}^{Q} \times \mathbb{R}^{\mathcal{X}}_{\geq 0})$, where $\widehat{\mathbb{B}} = (\mathbb{B} \cup \Omega)$ and $x_j^i \in \mathcal{X}^i$ for $1 \leq j \leq |\mathcal{X}^i|$ and $i = 1, 2$.

$$\begin{array}{llllll}
v_q = v_q^1 & \text{iff} & q \in Q^1 & \text{and} & a \in \Delta^1 \\
v_q = v_q^2 & \text{iff} & q \in Q^2 & \text{and} & a \in \Delta^2 \\
v_q = 0 & \text{iff} & (q \in Q^1 & \text{and} & a \notin \Delta^1) \\
 & \text{or} & (q \in Q^2 & \text{and} & a \notin \Delta^2) \\
v_q = 1 & \text{iff} & (q = q_x^1 & \text{and} & a \notin \Delta^1) \\
 & \text{or} & (q \in q_x^2 & \text{and} & a \notin \Delta^2) \\
v_q = u_q & \text{iff} & (q = q_x^1 & \text{and} & a \in \Delta^1) \\
 & \text{or} & (q \in q_x^2 & \text{and} & a \in \Delta^2) \\
v_x = v_x^1 & \text{iff} & x \in \mathcal{X}^1 & \text{and} & a \in \Delta^1 \\
v_x = v_x^2 & \text{iff} & x \in \mathcal{X}^2 & \text{and} & a \in \Delta^2 \\
v_q = 0 & \text{iff} & (x \in \mathcal{X}^1 & \text{and} & a \notin \Delta^1) \\
 & \text{or} & (x \in \mathcal{X}^2 & \text{and} & a \notin \Delta^2)
\end{array}$$

□

5 Properties of Ω-DTAFA

Given two Ω-DTAFA A^1 and A^2 such that $A^1 = (Q^1, \Delta^1, S^1, g^1, h^1, \mathcal{X}^1, f^1, \Omega^1)$, and $A^2 = (Q^2, \Delta^2 S^2, g^2, h^2, \mathcal{X}^2, f^2, \Omega^2)$, where A^1 and A^2 have n and m states, respectively. Then, there exists an Ω-DTAFA A, where $L(A) = L(A^1) \star L(A^2)$ for some binary operator $\star$, that has, in the worst case if no state reduction is performed, the following number of states:

Lemma 5.1 *Let A^1, A^2 be two Ω-DTAFA, with n and m states, respectively. There exists an Ω-DTAFA A such that $L(A) = L(A^1) \star L(A^2)$ for some binary operator $\star$ for which the following results hold:*
(i) A has $n + m + 2$ states iff $\Delta^1 \not\subseteq \Delta^2$ and $\Delta^2 \not\subseteq \Delta^1$.
(ii) A has $n + m + 1$ states iff $\Delta^1 \subset \Delta^2$ or $\Delta^2 \subset \Delta^1$.
(iii) A has $n + m$ states iff $\Delta^1 = \Delta^2$.

Theorem 5.1 *Timed regular languages accepted by Ω-DTAFA are closed under complementation.*

Proof.
Given an Ω-DTAFA $A = (Q, \Delta, S, g, h, \mathcal{X}, f, \Omega)$, the language accepted by A is $\mathcal{L}(A)$. The complement of this language $\overline{\mathcal{L}(A)}$ is accepted by an Ω-DTAFA $A' = (Q, \Delta, S, g, h, \mathcal{X}, f', \Omega)$ such that $f' = \overline{f}$ (the logical negation of the accepting function f). □

Theorem 5.2 *Ω-DTAFA are closed under all Boolean operations. Moreover, they are closed under several other important operators (For example, i.e., logical implication* ($\Leftarrow$), logical bi-conditional ($\Rightarrow$), difference (/), and symmetric difference ($\ominus$)).

Proof.
Due to space constraints we only give the proof and the algorithm of the union operation. Then, we can adapt this proof to construct the corresponding Ω-DTAFA of the other binary operations. Let A^1 and A^2 are two Ω-DTAFA such that $A^1 = (Q^1, \Delta^1, S^1, g^1, h^1, \mathcal{X}^1, f^1, \Omega^1)$, and $A^2 = (Q^2, \Delta^2, S^2, g^2, h^2, \mathcal{X}^2, f^2, \Omega^2)$ be two Ω-DTAFA. We can construct an Ω-DTAFA $A = (Q, \Delta, S, g, h, \mathcal{X}, f, \Omega)$ such that $\mathcal{L}(A) = \mathcal{L}(A^1) \cup \mathcal{L}(A^2)$.

Preconditions:
$Q^1 \cap Q^2 = \emptyset$, (it follows that $S^1 \cap S^2 = \emptyset$), $\mathcal{X}^1 \cap \mathcal{X}^2 = \emptyset$.

Algorithm:
$Q = Q^1 \cup Q^2 \cup \{q_x^1, q_x^2\}$ such that $q_x^1, q_x^2 \notin Q^1 \cup Q^2$, $\Delta = \Delta^1 \cup \Delta^2$, $S = S^1 \cup S^2$, $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$ and $\Omega = \Omega^1 \cup \Omega^2$. h is defined as in Section 3 given Q, Δ and $\mathcal{X}$. The function f is defined as follows:

$$f(q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2 \ldots, q_{|Q^2|}^2, q_x^2, x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2) =$$
$$(f^1(q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1) \wedge \overline{q_x^1}) \vee$$
$$(f^2(q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2) \wedge \overline{q_x^2})$$

where

$(q_1^1, q_2^1, \ldots, q_{|Q^1|}^1) \in Q^1$, $(q_1^2, q_2^2, \ldots, q_{|Q^2|}^2) \in Q^2$, $(x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1) \in \mathcal{X}^1$, $(x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2) \in \mathcal{X}^2$.

The operator $\vee$ is referred to as the "*critical binary operator*". The function g is defined as follows:

$g(u, a) = v, g^1(u^1, a) = v^1, g^2(u^2, a) = v^2$, where $u^1, v^1 \in (\mathbb{B}^{Q^1} \times \mathbb{R}^{\mathcal{X}^1})$; $u^2, v^2 \in (\mathbb{B}^{Q^2} \times \mathbb{R}^{\mathcal{X}^2})$; $u, v \in (\mathbb{B}^{Q} \times \mathbb{R}_{\geq 0}^{\mathcal{X}})$, $a \in \Delta$.

$v_q = v_q^1$	iff	$q \in Q^1$	and	$a \in \Delta^1$
$v_q = v_q^2$	iff	$q \in Q^2$	and	$a \in \Delta^2$
$v_q = 0$	iff	$(q \in Q^1$	and	$a \notin \Delta^1)$
	or	$(q \in Q^2$	and	$a \notin \Delta^2)$
$v_{q_x^1} = u_{q_x^1}$	iff	$a \in \Delta^1$		
$v_{q_x^1} = 1$	iff	$a \notin \Delta^2$		
$v_{q_x^2} = u_{q_x^2}$	iff	$a \in \Delta^2$		
$v_{q_x^1} = 1$	iff	$a \notin \Delta^2$		
$v_x = v_x^1$	iff	$x \in \mathcal{X}^1$	and	$a \in \Delta^1$
$v_x = v_x^2$	iff	$x \in \mathcal{X}^2$	and	$a \in \Delta^2$
$v_q = 0$	iff	$(x \in \mathcal{X}^1$	and	$a \notin \Delta^1)$
	or	$(x \in \mathcal{X}^2$	and	$a \notin \Delta^2)$

Note that:
$u = (q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, q_x^2, x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2)$
$u^1 = (q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, x_1^1, x_2^1, \ldots, x_{|\mathcal{X}^1|}^1)$
$u^2 = (q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, x_1^2, x_2^2, \ldots, x_{|\mathcal{X}^2|}^2)$
such that

$u_q = u_q^1$	iff	$q \in Q^1$
$u_q = u_q^2$	iff	$q \in Q^2$
$u_x = u_x^1$	iff	$x \in \mathcal{X}^1$
$u_x = u_x^2$	iff	$x \in \mathcal{X}^2$

□

The following Lemma is a direct result from the union and intersection algorithms.

Lemma 5.2 *For any integers $m, n \geq 1$, let A^1 be an m-state and A^2 be an n-state Ω-DTAFA. Then, $m + n + 2$ and $m + n + 1$ states are sufficient and necessary in the worst case* ($\Delta^1 \not\subseteq \Delta^2$, $\Delta^2 \not\subseteq \Delta^1$) for an Ω-DTAFA A to accept the languages $\mathcal{L}(A^1) \cup \mathcal{L}(A^2)$ and $\mathcal{L}(A^1) \cap \mathcal{L}(A^2)$, respectively. (Note that $A^1 \cup A^2$ and $A^1 \cap A^2$ are not reduced).

The critical binary operator can be redefined by a truth table to perform different logical language set operations. For example, $\overline{\wedge}$ (nand), $\overline{\vee}$ (*nor*), $\oplus$ (xor), $\Leftrightarrow$ (logical bi-conditional) and $\Rightarrow$ (logical implication).

Corollary 5.1 *The logical operators $\vee, \wedge, \overline{\vee}, \overline{\wedge}, \Rightarrow$, and $\Leftrightarrow$ generate the languages $\mathcal{L}(A^1) \vee \mathcal{L}(A^2)$, $\mathcal{L}(A^1) \wedge \mathcal{L}(A^2)$, $\overline{\mathcal{L}(A^1) \vee \mathcal{L}(A^2)}$, $\overline{\mathcal{L}(A^1) \wedge \mathcal{L}(A^2)}$, $\overline{\mathcal{L}(A^1)} \vee \mathcal{L}(A^2)$, and $\overline{\mathcal{L}(A^1 \oplus \mathcal{L}(A^2)}$, respectively.*

The logical bi-conditional ($a \Leftrightarrow b$) can describe and model the following language:

$$\begin{aligned}
& \overline{\mathcal{L}(A^1) \oplus \mathcal{L}(A^2)} \\
= \; & \overline{\mathcal{L}(A^1)/\mathcal{L}(A^2) \cup \mathcal{L}(A^2)/\mathcal{L}(A^1)} \\
= \; & \overline{\mathcal{L}(A^1) \cap \overline{\mathcal{L}(A^2)} \cup L(A^2) \cap \overline{\mathcal{L}(A^1)}} \\
= \; & \overline{\mathcal{L}(A^1) \cap \overline{\mathcal{L}(A^2)}} \cap \overline{\mathcal{L}(A^2) \cap \overline{\mathcal{L}(A^1)}} \\
= \; & \overline{\mathcal{L}(A^1)} \cup \mathcal{L}(A^2) \cap \overline{\mathcal{L}(A^2)} \cup \mathcal{L}(A^1)) \\
= \; & (\overline{\mathcal{L}(A^1)} \cap \overline{\mathcal{L}(A^2)}) \cup (\mathcal{L}(A^1) \cap \mathcal{L}(A^2))
\end{aligned}$$

Since Ω-DTAFA are closed under all Boolean operations, it is simple to prove that Ω-DTAFA are closed under difference and symmetric difference operations.

Lemma 5.3 *Let A^1 and A^2 be two Ω-DTAFA. Then, $\mathcal{L}(A^1)/\mathcal{L}(A^2) = \mathcal{L}(A^1) \cap \overline{\mathcal{L}(A^2)}$ and $\mathcal{L}(A^1) \ominus \mathcal{L}(A^2) = (\mathcal{L}(A^1) \cap \overline{\mathcal{L}(A^2)} \cup (\mathcal{L}(A^2) \cap \overline{\mathcal{L}(A^1)})$*

Proof.
For the difference operation, an easy way to construct an Ω-DTAFA $A = A^1/A^2$ is to perform the union algorithm, except replace the critical binary operator with the binary operator "/". For the symmetric difference $\ominus$, which is also the set-theoretic equivalent of the exclusive-or operation in Boolean logic. The symmetric difference algorithm construction can be done with the union algorithm by replacing the critical binary operator $\vee$ with the binary *xor* operator "$\oplus$". □

Ω-DTAFA may contain less states than a corresponding DTAFA. For example, the operator $\Leftrightarrow$ can be simulated using only $\wedge$ $\vee$, and $^-$. Thus in general, $2(n+m)+3$ states are needed to simulate the $\Leftrightarrow$ operator using $\wedge$ $\vee$, and $^-$ as opposed to only $n+m+1$ states using directly the operator $\Leftrightarrow$. Therefore, Ω-DTAFA contain less states than a corresponding DTAFA would. Also, the algorithm needs only to be run once using the $\Leftrightarrow$ operator as opposed to three times in DTAFA using $\wedge, \vee$ and $^-$ operators.

6 Conclusion

Despite being very expressive for describing timed behaviors and presenting a powerful computational parallel model, both TFA and TAFA are neither determinizable nor closed under complementation. For Ω-DTAFA, closure under complement relies on the mutual exclusive clock construction. Thus, Ω-DTAFA is a determinizable class of TAFA closed under all the Boolean operations, including some of the most important operators. The addition of the concepts of alternation and timing to finite automata increase their expressive and descriptional power, as measured by the size of the automaton. This decrease in the state complexity of Ω-DTAFA could simplify proofs in languages and complexity theory. Moreover, the issue of state complexity and expressiveness have immediate practical application in software and real-time systems.

Acknowledgments: We would like to thank the anonymous reviewers for their valuable comments.

References:

[1] R. Alur, D. Dill, A Theory of Timed Automata, *Theoret. Comput. Sci.* 126 (2), 1994, pp. 183-235.

[2] R. Alur, L. Fix, T.A. Henzinger, Event-Clock Automata: A Determinizable Class of Timed Automata, *Theoret. Comput. Sci.*, 211 (1-2), 1999. pp. 253-273.

[3] J.A. Brzozowski, E. Leiss, On Equations for Regular Languages, Finite Automata, and Sequential Networks, *Theoret. Comput. Sci.* 10, 1980, pp. 19-35.

[4] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, Alternation, *J. Assoc. Comput*, 28, 1981, pp. 114-133.

[5] A. Fellah, Boolean Web-Service Automata: A Parallel Model for Distributed Web Service Operations, *IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, Victoria, Canada, August 2005, pp. 416-419.

[6] A. Fellah, C. Harding, Language Equations for Timed Alternating Finite Automata, *Internat. J. Comput. Math.* 80,2, 2003, pp. 1075-1091.

[7] A. Fellah, DTAFA: A Deterministic Class of Timed Alternating Finite Automata. *Technical Report*, Dept. of Computer Science, University of Sharjah, Sharjah, U.A.E, 2005.

[8] E. Leiss, Succinct Representation of Regular Languages by Boolean Automata II. *Theoret. Comput. Sci.*, 38, 1985, pp. 133-136.

[9] O. Kupferman, M. Vardi, Weak Alternating Automata are not that Weak. *ACM Trans. Comput. Log.*, 2(3), 2001, pp. 408-429.

[10] K. Salomaa, X. Wu, S. Yu, Efficient Implementation of Regular Languages using Reversed Alternating Finite Automata, *Theoret. Comput. Sci.*, 231, 2000, pp. 103-111.

[11] S. Yu, Regular Languages, in: G. Rozenberg, A. Salomaa, (Eds.), *Handbook of Formal Languages*, Vol. I., Springer, Berlin, 1997, pp. 41-110.

[12] C.L. Oding, W. Thomas, Alternating Automata and Logics over Infinite Words. *In IFIP TCS'00*, Vol. 1872 of LNCS, 2000, pp. 521-535.

[13] C. Daws, A. Olivero, S. Tripakis, S. Yovine, The Tool KRONOS, *Hybrid Systems III: Verification and Control*, Springer, Vol. 1066, 1996, pp. 208-219.

[14] K.G. Larsen, P. Pettersson, W. Yi, UPPAAL is a Nutshell, *Springer Inter. J. of Software Tools for Technology Transfer*, Vol. 1(2), 1997, pp. 134-152.