Journal of Computer Science 3 (1): 1-8, 2007
ISSN 1549-3636

Deterministic Timed AFA: A New Class of Timed Alternating Finite Automata

[1]Abdelaziz Fellah, [2]Zachary Friggstad and [2]Soufiane Noureddine
[1]Department of Computer Science, University of Sharjah, Sharjah, United Arab Emirates
[2]Department of Math. and Computer Science, University of Lethbridge, Lethbridge, AB, Canada T1K 3M4

Abstract: Timed Alternating Finite Automata (TAFA), a natural generalization of Timed Finite Automata (TFA), are synchronous and powerful models for real-time computations. They become an effective and expressive model for developing embedded systems with real-time constraint computations which are required in many applications. We introduce Deterministic Timed Alternating Finite Automata (DTAFA), a new class of timed alternating finite automata, extended with a finite set of restricted and mutually exclusive real-valued clocks on events which trigger the state transitions of the automaton. We show how to transform deterministic *n*-state TFA into *log n*-state DTAFA and state some language properties between TFA, DTAFA, and deterministic TFA. We then show that, unlike TFA and TAFA, DTAFA are closed under all Boolean operations, including the complementation.

Key Words: Automata, formal languages, timed automata, timed alternating finite automata.

INTRODUCTION

Alternating finite automata (AFA) are a natural generalization of non-determinism 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, 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, [5, 6, 8, 9, 10, 12, 14].

Traditionally, 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 automata also called timed finite automata (TFA) have become a powerful canonical model for describing time 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 direction and several aspects of TFA have been investigated such as determinization, minimization, and power of clocks. 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[16], and KRONOS[15]. An extended version of timed automata with real time asynchronous processes has been also studied in[13], where each transition is annotated with a process that can be triggered by events as a model for embedded real-time systems. Moreover, timed automata are powerful and expressive models to describe synchronization and concurrency concepts.

The concept of *alternation* refers to the alternation of universal and existential quantifiers during the course of a computation. A formalization of this idea extended with a set of clocks and applied to finite state automata yields the definition of timed alternating finite automata [6]. 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"* in the sense that when the automaton reads an input symbol *a* in a given state *q* while the time constraints are satisfied, it will activate all states of the automaton to work in parallel on the remaining part of the input. Once the states have completed their tasks, *q* will evaluate their results using a Boolean function and pass the resulting value to the state by which it was activated. A state of a timed alternating finite automaton can be considered as a tuple containing the current state of the automaton and the current values of the clocks. Clocks are used to justify timed transitions and sequences in TAFA. Multiple clocks timed alternating finite automata would be particularly useful in model-

Corresponding Author : Abdelaziz Fellah, Department of Computer Science, University of Sharjah, Sharjah, U.A.E.

ing 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. A comprehensive analysis of the theory of TAFA based upon a hybrid combination of AFA and TFA models were proposed in[6]. In particular, an algebraic interpretation of TAFA which parallels that of timed regular expressions and language equations were developed and proved to be useful as a convenient way for representing TAFA. 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.

Event-clock automata (ECA) were introduced in[2] as the first determinizable subclass of timed automata by restricting the use of clocks. ECA are closed under Boolean operations. The key for the determinization of event clock automata is the property that each computation step, all clock values are determined only by the input word. That is, unlike timed automata model, where clock values depend on the path taken by the automaton and are determined by transition relations, event clock automata are characterized by a fixed, predefined association between the clocks and the symbols of the input alphabet. In this paper and along the lines of [2], we show 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 by the input sequence and doesn't depend on nondeterminism. The main property, which fails for arbitrary timed AFA, hold for all deterministic timed AFA. That is, deterministic timed alternating finite automata (DTAFA) are closed under all operations; in particular, for every DTAFA we can construct a DTAFA that defines the complement of a timed language.

In the timed setting and unlike AFA, timed AFA are closed under union and intersection operations, but not under complementation. The aim of this paper is to propose a formalism which sufficiently expressive to model TAFA for which all Boolean operations can be effectively defined. A solution to this is to introduce deterministic timed alternating finite automata (DTAFA), a class of timed AFA extended with a set of finite set of restricted real-valued clocks. The clocks are divided into mutually exclusive sets and a restricted association is predefined between the clocks and symbols of the input alphabet. Using the fact of mutual exclusive clocks justifies the determinization of TAFA, which, in turn, leads to the complementation of DTAFA. The underlying deterministic and mutually exclusive time property and the predefined association between input symbols and time-stamps define a subclass of timed AFA models which could be compared to event-recording and event-predicting in event timed automata models[2].

PRELIMINARIES

In this section we briefly recall the basic concepts and notations used in this paper. For a more detailed presentation on the formalisms of timed automata and alternating finite automata the reader may refer to[1,11].

We denote by $R_{\geq 0}$ and N the set of all nonnegative reals including 0 and the set of positive natural numbers, respectively. The cardinality of a finite set A is $|A|$. An *alphabet* Δ is a finite, nonempty set whose elements are called *symbols* or *letters*. A *timed word*, w over Δ a finite sequence $\rho = (a_1, t_1)(a_1, t_1) \ldots (a_i, t_i)$ where the a_i's are symbols of Δ and the t_i's are in $R_{\geq 0}$ 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, $t_1 \ldots t_i$ is a finite monotonically non decreasing time sequence of $R_{\geq 0}$. A *timed trace* (run) is a finite sequence $(a_1, t_1)(a_1, t_1) \ldots (a_i, t_i)$. 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 $R_{\geq 0}$ and both have the same length. The time language $(\Delta \times R_{\geq 0})^*$ is the set of all timed words over Δ where where λ denotes the empty timed word. Recall that classical words Δ the free monoid (Δ^*, o, λ) generated by Δ where "o" is the classical concatenation operator (we write ab rather than aob for the concatenation). Timed words are defined over the combination of the monoid (Δ, o, λ) and the time monoid $(R_{\geq 0, +}, 0)$. For any language $L \subseteq \Delta^*$, $\overline{L} = \Delta^* \backslash L$, is the complement of L with respect to Δ^*. For languages L_1 and L_2 over Δ, the union and intersection are denoted by $L_1 \cup L_2$ and $L_1 \cap L_2$, respectively.

DETERMINISTIC TIMED AFA

Let X be a set of clock variables, a *clock constraint* ψ over 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 X and $c \in R_{\geq 0}$ such that $c \geq 0$. The operators $\vee$ and $\wedge$ stands for the logical-or and logical-and, respectively.

A clock interpretation ν for X is a mapping from X to $R_{\geq 0}$ (*i.e.*, ν assigns to each clock $x \in X$ the value ν(x)). A clock interpretation represents the values of all clocks in 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 $\psi = 0 \leq x_1 \wedge 0 \leq x_2 \wedge \ldots \wedge \leq 0 \leq x_{|X|} \wedge \psi$ is implied for all $x_i \in X$ where $1 \leq i \leq |X|$. 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: *A deterministic timed alternating finite automaton* (DTAFA) is *a seven-tuple* $A = (Q, \Delta, S, g, h, X, f)$, *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) X *is a finite set, the set of clocks,*
(e) h *is a time transition function,*
$h: (B^Q \times R_{\geq 0}^X) \times (\Delta \times R_{\geq 0}) \rightarrow (B^Q \times (R_{\geq 0}^X \times R_{\geq 0})) \times \Delta$,
(f) g *is a letter symbol transition function from* Q *into the set of all functions from* $\Delta \times (B^Q \times R_{\geq 0}^X)$ into $B^Q \times R_{\geq 0}^X$, *that is,* $g: (B^Q \times R_{\geq 0}^X) \times \Delta \rightarrow B^Q \times R_{\geq 0}^X$,
(g) f *is a time accepting function,* $f: \rightarrow B^Q \times R_{\geq 0}^X \rightarrow B$.

We denote by the symbol B the two-element Boolean algebra $B = (\{0, 1\}, \vee, \wedge, \bar{}, 0, 1)$. B^Q is a vector with $|Q|$ elements referring to all the Boolean functions from Q to B, and $R_{\geq 0}^X$ is a vector with $|X|$ elements (all non-negative) which refers to all real functions from X to $R_{\geq 0}$. More specifically, the function h is defined as:

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

where $q_i \in Q$ for $1 \leq i \leq |Q|$, $x_j \in X$ for $1 \leq j \leq |X|$, $a \in \Delta$ and $t \in R_{\geq 0}$ such that $t \geq 0$.

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

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

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

For each state $q \in Q$, $x \in X$ and $a \in \Delta$, we define $g_q(a)$ to be the Boolean function $(B^Q \times \Delta) \rightarrow B$ and $g_x(a)$ to be the function $(R_{\geq 0}^X \times \Delta) \rightarrow R_{\geq 0}$ such that
$g_q(u')(a) = g_q(u', a)$ and $g_x(v')(a) = g_x(v', a)$; $u' \in B^Q$ and $v' \in R_{\geq 0}^X$. 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 to be the function $(B^Q \times \Delta) \rightarrow B^Q$ by taking all the $|Q|$ functions $g_q : (B^Q \times \Delta \rightarrow B, q \in Q$. Similarly, we define $g_x(v', a)$ to be the real function $(R_{\geq 0}^X \times \Delta) \rightarrow R_{\geq 0}^X$ by taking all the $|X|$ functions $g_x: (B^Q \times R_{\geq 0}^X) \rightarrow R_{\geq 0}$, $x \in X$. For notational convenience, $g_Q(u', a)$ and $g_x(v', a)$ can be written as one mapping, $g(u, a)$, $u \in (B^Q \times R_{\geq 0}^X)$ and $a \in \Delta$.

Definition: Let $A = (Q, \Delta, S, g, h, X, f)$ be a DTAFA and $w \in (\Delta \times R_{\geq 0})^*$ be a timed word. w is accepted by A if and only if $f(g(h(s,w))) = 1$, where $s \in (B^Q \times R_{\geq 0}^X)$ 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 $x \in X$.

Example: Consider the following DTAFA $A = (Q, \Delta, S, g, h, X, f)$ where $Q = \{q_0, q_1, q_2\}$, $\Delta = \{a, b, c\}$, $S = \{q_0\}$, $X = \{x, y\}$, $f(q_0, q_1, q_2, x, y) = q_0 \wedge \overline{q_1} \wedge q_2$, 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 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 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 $u \in (B^Q \times R_{\geq 0}^X)$, $a \in (\Delta \times R_{\geq 0})$ for all clocks $x \in X$. If the Boolean value is 1, then we write $1 \,.\, x = x$ (or $x \,.\, 1 = x$) to indicate a no reset. The symbol "." is the *reset operator* and not the usual concatenation symbol. If the Boolean value is 0, then we write $0 \,.\, x = 0$ (or $x \,.\, 0$) to indicate a reset. In fact, the reset operator "." 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 $0 \,.\, x$ (or $x \,.\, 0$). Likewise, if an entry of the table contains an expression that is x, this implies the expression is $1 \cdot x$ (or $x \,.\, 1$). Moreover, $(q_1 \vee q_2) \,.\, x$ means that x is reset if $q_1 = 0$ and $q_2 = 0$. The following example traces the acceptance of a timed word w.

Example: Let $w = (b,2)\ (a,3)\ (c,3)$

$f(g(h(s, (b, 2)\ (a, 3)\ (c, 3))))$

$= f(g(h(g(h(s, (b, 2), (a, 3))), (c, 3))))$

$= f(g(h(g(h(g(h(s, (b, 2))), (a, 3))), (c, 3))))$

$= f(g(h(g(h(g(h((1, 0, 0, 0, 0),(b, 2))),(a ,3))),(c, 3))))$

$= f(g(h(g(h(g((1, 0, 0, 2, 2, b), (a, 3))), (c, 3))))$

$= f(g(h(g(h((0, 1, 0, 2, 2), (a, 3))), (c, 3))))$

$= f(g(h(g((0, 1, 0, 3, 3), a)), (c, 3))))$

$= f(g(h((1, 0, 0, 3, 0), (c, 3))))$

$= f(g((1, 0, 0, 3, 0), c))$

$= f(1, 0, 1, 0, 0)$

$= 1 \wedge \bar{0} \wedge 1 = 1.$

DTFA AND DTAFA

In this section, we show how to transform an *|L|-state deterministic timed finite automaton* (DTFA), into an equivalent *log |L|-state deterministic timed alternating finite automaton* (DTAFA). We also show the opposite construction of a DTFA from a given DTAFA.

Definition: *A* timed finite automaton (TFA) is a septtuple $A = (L,\Sigma,L_0,X^L,I,E,L^f)$, where (a) *L is a finite set of locations*, *the set of states*, (b) Σ is an alphabet, *the input alphabet*, (c) $L_0 \subseteq L$ is the set of all starting locations (*starting states*), (d) X^L is a finite set, the set of clocks, (e) I is a set of invariants of clocks on a state, (f) E is a set of edges between states, (g) L^f is a set of final locations.

A deterministic timed finite automaton (DTFA) is a timed finite automata with the following key properties:

i) $|L_0| = 1$.

(*ii*) $I = \emptyset$.

(*iii*) All clock constraints on a given input symbol from a given state must be mutually exclusive and must span all $R_{\geq 0}^X$.

(*iv*) All clock zones[1] on a given symbol must be defined. Clock zones are represented as conjunctions of timing constraints.

Theorem: *Let* $A = (L,\Sigma,L_0,X^L,I,E,L^f)$ *be an |L|-state DTFA there exists an equivalent* log *|L|-state DTAFA* $A' = (Q,\Delta,S,g,h,X,f)$ *such that* $L(A) = L(A')$.

Proof. Without loss of generality, number the locations of L from 0 to $|L|$ - 1. Now, let N_i denote the location numbering where $0 \leq i \leq |L|$ - 1 and $Bin(N_i)$ be the equivalent binary representation of N_i with exactly m bits where $m = \lceil d \log_2 |L| \rceil$. Define $Q = \{q_1,\ldots,q_m\}$ to be the state of the DTAFA, and set $\Delta = \Sigma$ and $X = X^L$. Define the function h for A':

$$h((q_1, \ldots, q_m, x_1, x_2,\ldots, x_{|X|},(a,t)) =$$
$$((q_1, \ldots, q_m, x_1+t, x_2+t,\ldots, x_{|X|}+t),a)$$

where $a \in \Delta$ and $t \in R_{\geq 0}$.

Let $Bin(N_i) = b_1 \ldots b_m$ denote the equivalent binary representation of N_i where b_1, b_2 and b_m indicate the least significant bit (LSB), the next LSB, and the most significant bit (MSB), respectively. Identify the least significant bit side of $Bin(N_i)$ and pad it with as many zeros as required to make the total number of bits of $Bin(N_i)$ equals to m. Denote by $\mathcal{B} = b_1 \ldots b_m$. Then, for each bit $b \in \mathcal{B}$, define a state association mapping $\varphi(b)$ from $\mathcal{B} \rightarrow Q : (b_1 \ldots b_m) \rightarrow (q_1,\ldots,q_m)$. That is, we identify the bits b_1 with state q_1, b_2 with state q_2 and we proceed with this until all bits have a state association (the MSB should be associated with state q_m). Consider the binary number associated with the state in L_0, denoted by $\mathcal{B}^0 = b_1^0 \ldots b_m^0$. Then for all $i = 1, \ldots, m$, the following conditions hold:

(*i*) If $b_i^0 = 0$, then $\varphi(b_i^0) \notin S$.

(ii) If $b_i^0 = 0$, then $\varphi(b_i^0) \in S$.

Consider the set of final locations, $L^f = \{L_1^f, L_2^f, \ldots, L_k^f\}$, where k is the number of final locations of A. Let

$$\mathcal{B}_1^f = b_1^1 \ldots b_m^1$$
$$\mathcal{B}_2^f = b_1^2 \ldots b_m^2$$
$$\ldots$$
$$\mathcal{B}_k^f = b_k^f \ldots b_m^f$$

denote the binary representations associated with each location in L^f and let $\varphi(b^1), \ldots, \varphi(b^k)$ be their respective state association mappings. For each location in L^f, derive the expression:

$$E_i^f = \bigvee_{i=1}^{k-1} \left(\bigwedge_{j=1}^{m} s_j^i \right)$$

Where

$$s_j^i = \begin{cases} \overline{q_j^i} & \text{iff } b_j^i = 0 \\ q_j^i & \text{otherwise} \end{cases}$$

and where $q_j^i \in Q$, i=1,…, k-1, j =1,…,m, and the operators $\vee$ and $\wedge$ indicate the bit-wise operations *or* and *and*, respectively. Set $f(q_1,\ldots,q_m, x_1, x_2, \ldots, x_{|X|}) = e^f$ where e^f is the reduced Boolean expression of E_i^f obtained by any known Boolean expression simplification algorithm.

For each input symbol $a \in \Sigma$, consider all clock constraints associated with this symbol. If any two 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.

Consider each constraint for each input symbol $a \in \Sigma$. For each location in L, consider its binary equivalent representation and the binary equivalent of the state association φ it goes to upon the considered input symbol and clock constraint. We perform the state association mapping for each state, then derive a Boolean expression of the $q_m,\ldots, q_1$ bits for each bit of the resulting state list. This will result in obtaining the expression for the bit in the state list upon reading an input symbol within the given time constraint.

Finally, for the clock resets, for each a in Σ, consider each clock constraint for each symbol. For each location in L and each clock in X, consider its binary equivalent representation. If its edge on the given input symbol and constraint resets the clock, then store it as a resulting Boolean false function 0, or else store it as a resulting Boolean true function 1. For each clock, derive a Boolean expression, given the binary values of the states, such that the expression is 1 if and only if the binary representation of the state does not reset its clock on the given transition. Finally, "*dot*" the expression by the interpretation of the clock X so if the transition resets the clock, then $x = 0 \,.\, x$, and if it doesn't reset the clock, then $x = 1 \,.\, x$.

Theorem: *Let* $A = (Q,\Delta,S,g,h,X,f)$ *be a DTAFA there exists an equivalent DTFA* $A' = (L,\Sigma,L_0\ X^L,\ I,E,L^f)$ *such that* $L(A) = L(A')$.

Proof. Initially, we set $\Sigma = \Delta$, $X^L = X$, and $I = \emptyset$. Associate each state in Q with a bit in any $|Q|$-bit binary number. Now, without loss of generality, arrange the states in the order as $q_{|Q|}, q_{|Q|-1}, \ldots, q_1$. We denote a binary number formed by these states and bits association as $q_{|Q|}, q_{|Q|-1}, \ldots, q_1$.

Let L be a set of $2^{|Q|}$ states, numbered from 0 to $|Q|$ - 1. Let the state numbered $q_{|Q|}, q_{|Q|-1}, \ldots, q_1$, where where $q_i = 1$ if and only if $q_i \in S$, be the state in L_0 where $1 \le i \le Q$. The set of L^f is the set of all numbered states such that in their binary representation $q_{|Q|}$, $q_{|Q|-1},\ldots, q_1$, the value of all the states satisfies the function f.

Consider a location $q_{|Q|}, q_{|Q|-1},\ldots, q_1$. Derive the next state on a given symbol and time constraints by forming the number $q'_{|Q|}, q'_{|Q|-1}, \ldots, q'_1$, where q'_i is the binary value of that state after the transition where $1 \le i \le Q$. Define an edge from $q_{|Q|}, q_{|Q|-1},\ldots, q_1$ to $q'_{|Q|}, q'_{|Q|-1}, \ldots, q'_1$ with the considered input symbol and time constraint. Then, include all clocks x_j that are reset within this transition by determining if the function $g(q_1, q_2,\ldots, q_{|Q|}, x_1, x_2, \ldots, x_{|X|})$ results in resetting x_j where $j = 1, \ldots, |X|$. The resulting DTFA may not be reduced, however, any standard reduction algorithm may be used to reduce the number of states.

DTAFA: PROPERTIES

COMPLEMENTATION

Theorem: *Timed regular languages accepted by DTAFA are closed under complementation.*

Proof. Given a DTAFA $A = (Q,\Delta,S,g,h,X,f)$, the language accepted by A is $L(A)$. The complement of this language $\overline{(L(A))}$ is accepted by a DTAFA $A' = (Q, \Delta,S,g,h,X,f')$, where $f' = \overline{f}$ ($\overline{f}$ is the logical negation of f).

UNION

Theorem:
Given two DTAFA $A^1 = (Q^1, \Delta^1, S^1, g^1, h^1, X^1, f^1)$, *and* $A^2 = (Q^2, \Delta^2, S^2, g^2, h^2, X^2, f^2)$ *we can construct a DTAFA* $A = (Q, \Delta,S,g,h,X,f)$ *such that* $L(A) = L(A^1) \cup L(A^2)$.

Proof: We assume that $Q^1 \cap Q^2 = \emptyset$, it follows that $S^1 \cap S^2 = \emptyset$. $X^1 \cap X^2 = \emptyset$.
Let $Q = Q^1 \cup Q^2 \cup \{q_x^1, q_x^2\}$ such that $q_x^1, q_x^2 \notin$

$Q^1 \cup Q^2$, $\Delta = \Delta^1 \cup \Delta^2$, $S = S^1 \cup S^2$, $X = X^1 \cup X^2$.

Given Q, Δ, and X, h is defined as:

$$h((q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, q_x^2, x_1^1, x_2^1, \ldots, x_{|X^1|}^1, x_1^2, x_2^2, \ldots, x_{|X^2|}^2), (a,t))$$

$$= ((q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, q_x^2, x_1^1 + t, x_2^1 + t, \ldots, x_{|X^1|}^1 + t, x_1^2 + t, x_2^2 + t, \ldots, x_{|X^2|}^2 + t), a)$$

The function f is defined as follows:

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

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

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

where

$q_1^1, q_2^1, \ldots, q_{|Q^1|}^1 \in Q^1$; $q_1^2, q_2^2, \ldots, q_{|Q^1|}^2 \in Q^2$; $x_1^1, x_2^1, \ldots, x_{|X^1|}^1 \in X^1$; $x_1^2, x_2^2, \ldots, x_{|X^2|}^2 \in X^2$;

and the logical operator "$\vee$" is referred to as the "*critical binary operator*". The function g is defined in the following manner:

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

$v_q = v_q^1$ 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 $a \in \Delta^2$

$v_x = v_x^1$ iff ($x \in X^1$ and $a \in \Delta^1$)

$v_x = v_x^2$ iff ($x \in X^2$ and $a \in \Delta^2$)

$v_q = 0$ iff ($x \in X^1$ and $a \notin \Delta^1$)
or ($x \in X^2$ and $a \notin \Delta^2$)

Note that:

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

$$u^1 = (q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, x_1^1, x_2^1, \ldots, x_{|X^1|}^1)$$

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

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 $x \in X^1$,

$u_x = u_x^2$ iff $x \in X^2$.

INTERSECTION

Theorem:

Given two DTAFA $A^1 = (Q^1, \Delta^1, S^1, g^1, h^1, X^1, f^1)$, *and* $A^2 = (Q^2, \Delta^2, S^2, g^2, h^2, X^2, f^2)$ *we can construct a DTAFA* $A = (Q, \Delta, S, g, h, X, f)$ *such that* $L(A) = L(A^1) \cap L(A^2)$.

Proof. We can adapt the proof of the previous theorem to construct a DTAFA $A = (Q, \Delta, S, g, h, X, f)$, with the same preconditions, as we construct the union of two DTAFA. The only difference is that the operator we refer to as the critical binary operator is changed to $\wedge$ (logical and), instead of $\vee$ (logical or). It can be noted that the intersection algorithm can be constructed with a single $q_x \notin (Q^1 \cup Q^2)$ instead of q_x^1 and q_x^2. In addition, the following holds in both the intersection and union operations:

If $\Delta^1 = \Delta^2$, then q_x^1, q_x^2 are not required.
If $\Delta^1 = \Delta^2$, then q_x^2 is not required.
If $\Delta^1 = \Delta^2$, then q_x^1 is not required.

Corollary: *For any integers* $m, n \geq 1$ *let* A^1 *be an m-state and* A^2 *be an n-state DTAFA. Then* $m + n + 2$ *and* $m + n + 1$ *states are sufficient and necessary in the worst case for a DTAFA A to accept the languages* $L(A^1) \cup L(A^2)$ *and* $L(A^1) \cap L(A^2)$, *respectively.*

COMPOSITION OF DTAFA

Theorem:

Given two DTAFA $A^1 = (Q^1, \Delta^1, S^1, g^1, h^1, X^1, f^1)$, *and* $A^2 = (Q^2, \Delta^2, S^2, g^2, h^2, X^2, f^2)$ *we can construct a DTAFA* $A = (Q, \Delta, S, g, h, X, f)$ *such that* $L(A) = L(A^1) \otimes L(A^2)$ *where* $\otimes$ *is the composition operation.*

Proof. The steps of the algorithm are described in the following proof. We assume the following preconditions:

$Q^1 \cap Q^2 = \emptyset$, it follows that $S^1 \cap S^2 = \emptyset$, $X^1 \cap X^2 = \emptyset$. Let $Q = \{Q^1 \cup Q^2\}$, S = $\{S^1 \cup S^2\}$, $\Delta = \{\Delta^1 \cup \Delta^2\}$, $X = \{X^1 \cup X^2\}$.

The function h is defined as follows:

$$h((q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, q_x^2, x_1^1, x_2^1, \ldots, x_{|X^1|}^1, x_1^2, x_2^2, \ldots, x_{|X^2|}^2), (a,t)$$

$$= ((q_1^1, q_2^1, \ldots, q_{|Q^1|}^1, q_x^1, q_1^2, q_2^2, \ldots, q_{|Q^2|}^2, q_x^2, x_1^1 + t, x_2^1 + t, \ldots, x_{|X^1|}^1 + t, x_1^2 + t, x_2^2 + t, \ldots, x_{|X^2|}^2 + t), a)$$

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

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

We define g as:

$g(u, a) = v$ where $u_q = u_q^1$ iff $q \in Q^1$

$g^1(u^1, a) = v^1$ $u_q = u_q^2$ iff $q \in Q^2$

$g^2(u^2, a) = v^2$ $u_x = u_x^1$ iff $x \in X^1$

$u_x = u_x^2$ iff $q \in X^2$

$v_q = v_q^1$ iff ($q \in Q^1$ and $a \in \Delta^1$)

$v_q = u_q$ iff ($q \in Q^1$ and $a \notin \Delta^1$)

or ($q \in Q^2$ and $a \notin \Delta^2$)

$v_q = v_q^2$ iff ($q \in Q^2$ and $a \in \Delta^2$)

$v_x = v_x^1$ iff ($x \in X^1$ and $a \in \Delta^1$)

$v_x = v_x^2$ iff ($q \in X^2$ and $a \in \Delta^2$)

$v_x = u_x$ iff ($x \in X^1$ and $a \notin \Delta^1$)

or ($x \in X^2$ and $a \notin \Delta^2$)

One can define the time-concatenation of two timed words w_1 and w_2, as $w_1 \Theta w_2 = (a^1,t^1) \Theta (a^2,t^2)$ where $a^1, a^2 \in \Delta^*$, and t^1, t^2 are time sequences of the same length as a^1 and a^2, respectively.

Let $a^1 = a_1^1 a_2^1 \ldots a_n^1$ and $a^2 = a_1^2 a_2^2 \ldots a_m^2$ where $|a^1| = n \in N$, and $|a^2| = m \in N$; $a_i^1, a_j^2 \in \Delta$ where $1 \le i \le n$ and $1 \le j \le m$.

Let $t^1 = t_1^1 t_2^1 \ldots t_n^1$ and $t^2 = t_1^2 t_2^2 \ldots t_m^2$ be the corresponding time sequences. Then

$(a^1,t^1) \Theta (a^2,t^2) =$

$(a_1^1 a_2^1 \ldots a_n^1, t_1^1 t_2^1 \ldots t_n^1) \Theta (a_1^2 a_2^2 \ldots a_m^2, t_1^2 t_2^2 \ldots t_m^2) = (a_1^1 a_2^1 \ldots a_n^1 a_1^2 a_2^2 \ldots a_m^2, t_1^1 t_2^1 \ldots t_n^1 (t_1^2 + t_n^1)(t_2^2 + t_n^1) \ldots (t_m^2 + t_n^1) = (a^1 a^2, t)$

Proposition: *The time-concatenation operation* Θ *is:* (1) associative, (2) right and left distributive over + ,and (3) can be used with Δ to form a non-commutative monoid (Δ^*, Θ).

Let w_1 and w_2 be two timed words. Then we can define the operation of timed concatenation of L_1 with L_2 to be $L_1 \Theta L_2 = \{w_1 \Theta w_2 \mid w_1 \in L_1$ and $w_2 \in L_2\}$. Similarly, other time operations can be defined. For example, union $L_1 \cup L_2$, intersection $L_1 \cap L_2$, power $(L)^n$, and star operations $(L_1)^* = \bigcup_{i \in N} (L_1)^i$, $(L)_1^{\Theta} = \bigcup_{i \in N} (L_1)^i$ where $(L_1)^i$ denotes occurrences of L_1 (*i.e.*, finitely iterated time-concatenation of the enclosed timed regular language). The operation Θ denotes the timed version of the Kleene star.

Proposition: The family of timed regular languages is closed under the time-concatenation operation.

CONCLUSION

Alternation, timing and determinism add perfect features to automata expressiveness, parallelism, and succinctness, which have practical applications in software and real-time systems. It also could potentially lead to answer several open problems in formal languages and complexity. DTAFA, a class of timed alternating finite automata (TAFA), have been proposed in this paper. Despite being very expressive for describing timed behaviors, both timed finite automata (TFA) and TAFA are neither determinizable nor closed under complementation. However, these limitations could be considered as drawbacks in several model checking applications and real-time systems. Also, the lack of complementation of TAFA and TFA could give a weak theoretical endorsement of these machines from the point of view languages. We show that DTAFA are closed under all Boolean operations, including the complementation. Transformation algorithms between DTFA and DTAFA were also given. Currently, we are investigating a representation of DTAFA by systems of language equations.

REFERENCES

1. Alur, R. and Dill, D., 1994. A Theory of Timed Automata: Theoret. Comput. Sci., 126 (2): 183-235.
2. Alur, R., Fix, L. and Henzinger, T.A., 1999. Event-Clock Automata: A Determinizable Class of Timed Automata: Theoret. Comput. Sci.,211(1-2): 253-273.

3. Brzozowski, J.A. and Leiss, E., 1980. On Equations for Regular Languages, Finite Automata, and Sequential Networks: Theoret. Comput. Sci., 10: 19-35.
4. Chandra, A.K., Kozen, D.C., and Stockmeyer, L.J., 1981. Alternation: J. Assoc. Comput., 28: 114-133
5. Fellah, A., Jurgensen, H., and Yu, S., 1990. Constructions for Alternating Finite Automata: Internat. J. Comput. Math., 35: 117-132
6. Fellah, A. and Harding, C., 2003. Language Equations for Timed Alternating Finite Automata: Internat. J. Comput. Math., 80(2): 1075-1091.
7. Fellah, A., 2005. On Deterministic Timed AFA: Technical Report, Dept. of Computer Science, University of Sharjah, Sharjah, U.A.E.
8. Leiss, E., 1985. Succinct Representation of Regular Languages by Boolean automata II: Theoret. Comput. Sci., 38: 133-136.
9. Kupferman, O. and Vardi, M., 2001. Weak Alternating Automata are not that Weak: ACM Trans. Comput. Log., 2(3): 408-429.
10. Salomaa, K., Wu, X., and Yu, S., 2000. Efficient Implementation of Regular Languages using Reversed Alternating Finite Automata: Theoret. Comput. Sci., 231: 103-111.
11. Yu, S., 1997. Regular Languages, in: G. Rozenberg, A. Salomaa, (Eds.), Handbook of Formal Languages, vol. I.,: Springer, Berlin
12. Henzinger, T.A., B. Horowitz, B., and Kirsch, C.M., 2001. Giotto: A Time-Triggered Language for Embedded Programming: Proc. of EMSOFT, 166-184.
13. Fersman, E., Pettersson, P., and Yi, W., 2002, Timed Automata with Asynchronous Processes: Schedulability and Decidability: Lect. Notes in Computer Sci., 2280, Springer-Verlag.
14. Oding, C.L., and Thomas, W., 2000, Alternating Automata and Logics over Infinite Words: IFIP TCS, 1872 of LNCS-2000, 521-535.
15. Daws, C., Olivero, A., Tripakis, S., and Yovine, S., 1996. The Tool KRONOS: Hybrid Systems III: Verification and Control, Springer, 1066: 208-219.
16. Larsen, K.G., Pettersson, P., and Yi,W. 1997. UPPAAL is a Nutshell: Springer Inter. J. of Software Tools for Technology Transfer, 1(2): 134-152.