Unambiguity in Timed Regular Languages: Automata and Logics

Paritosh K. Pandya[*] and Simoni S. Shah

Tata Institute of Fundamental Research, Colaba, Mumbai *400005*, India
`pandya@tifr.res.in`

Abstract. Unambiguous languages (UL), originally defined by Schutzenberger using unambiguous polynomials, are a robust subclass of regular languages. They have many diverse characterizations: they are recognized by partially-ordered two-way deterministic automata (*po2dfa*), they are definable by Unary Temporal Logic (UTL) as also by the two variable first-order logic over words ($FO^2[<]$).

In this paper, we consider the timed version of unambiguous languages. A subclass of the two-way deterministic timed automata (*2DTA*) of Alur and Henzinger, called *partially-ordered two-way deterministic automata* (*po2DTA*) are examined and we call the languages accepted by these as *Timed Unambiguous Languages* (TUL). This class has some interesting properties: we show that *po2DTA* are boolean closed and their non-emptiness is NP-Complete. We propose a deterministic and unary variant of MTL called $DUMTL$ and show that $DUMTL$ formulae can be reduced to language equivalent *po2DTA* in polynomial time, giving NP-complete satisfiability for the logic. Moreover, $DUMTL$ is shown to be expressively complete for *po2DTA*. Finally, we consider the unary fragments of well known logics MTL and MITL and we show that neither of these are expressively equivalent to *po2DTA*. Contrast this with the untimed case where unary temporal logic is equivalent to *po2dfa*.

1 Introduction

Regular languages and their automata are a well established formalism which have made considerable impact on techniques for modelling and verification of systems. The connection between temporal logics and finite automata provides the key to algorithmic analysis and reasoning about logical properties of reactive systems.

Attempts to extend the above paradigm to timed logics and timed automata has proved challenging. One impediment is that logics have boolean operations and the timed automata (of Alur-Dill) are not closed under complementation. Thus, one must consider some variant/subclass of the timed automata which are boolean closed. The classical timed logic, MTL (interpreted over timed words), is undecidable and has no automaton characterization. This logic uses time constrained until and since operators U_I and S_I where I is a time interval with

[*] Corresponding author.

integral endpoints: we denote this by $MTL[\mathsf{U}_I, \mathsf{S}_I]$. Various subclasses of MTL have been investigated for decidablity. Ouaknine and Worrell have shown [OW05] that over finite timed words the logic $MTL[\mathsf{U}_I]$ is decidable by reducing the logic to 1-clock alternating timed automata which are boolean closed. The emptiness of 1-clock ATA is decidable with non-primitive-recursive complexity.

An altogether different approach is followed by Alur, Feder and Henzinger who consider logic MTL with only non-punctual (non-singleton) intervals in the modalities U_I and S_I. The resulting logic is called $MITL$ [AFH96]. In their pioneering work called "Back to the Future", Alur and Henzinger have defined the notion of *two-way deterministic timed automata* (*2DTA*) and shown that the subclass of reversal-bounded *2DTA* is boolean closed. The emptiness of *2DTA* with bounded number of reversals is PSPACE-complete. Alur and Henzinger also give a reduction from logic $MITL[\mathsf{U}_I, \mathsf{S}_I]$ to reversal bounded *2DTA*. Using this they show that the satisfiability of the logic is EXPSPACE-complete. Real-time Temporal Logics with low decision complexities are a rarity.

In this paper, we consider a subclass of the reversal-bounded *2DTA* of Alur and Henzinger called *partially-ordered two-way deterministic timed automata* (*po2DTA*). The automaton is partially ordered in the sense that the underlying graph is acyclic upto self loops. By restricting to this subclass, we show that we can achieve better complexities for the decision problems for the automata and their logics. We call the languages accepted by these automata as *timed unambiguous languages* (TUL). In this nomenclature, we are motivated by the unambiguous languages (UL), originally proposed by Schutzenberger [Sch75]. Schwentick *et al* showed that UL is precisely the subclass of regular languages that can be recognized by partially-ordered two-way deterministic finite automata (*po2dfa*) [STV01]. Moreover, Etessami *et al* [EVW02] showed that UL are precisely the languages definable by the Unary Temporal Logic, UTL, which is linear temporal logic using only the modalities $\overrightarrow{\Diamond}$ and $\overleftarrow{\Diamond}$. In our recent work, we proposed an unambiguous interval temporal logic, *UITL*, which also specifies exactly the class UL [LPS08]. One feature of this deterministic (or unambiguous) logic is that every model (word) can be "uniquely parsed" to match the formula. The survey article by Diekert *et al* gives a comprehensive treatment of UL [DGK08].

In this paper, we explore the properties of *po2DTA* and show that these automata are closed under the operations of union, intersection and complementation. In fact, each of these operations only results in a linear blowup in the size of the automaton. We also show that non-emptiness of a *po2DTA* is decidable and is NP-Complete. Membership in NP is based on showing that every *po2DTA* with non-empty language has a "small-sized" timed word in its language which uses only a limited set of integral and fractional values in the time stamps. It is possible to guess this word non-deterministically and to simulate the *po2DTA* in time polynomial in the number of states to check for non-emptiness. Contrast this with k-reversal bounded *2DTA* for which emptiness is PSPACE-complete for fixed k. It follows that the language inclusion of *po2DTA* is Co-NP-Complete.

As our next key exploration, we consider logics for specifying timed unambiguous languages. We propose a variant of MTL called $DUMTL$ (Deterministic Unary MTL) where U_I and S_I are replaced by "deterministic" and "unary" operators $\xi \mathsf{U}_\theta^x$ and $\xi \mathsf{S}_\theta^x$. A guarded event θ has the form (a, g) where guard g puts constraints on the time of occurrence of a relative to other events in the word. The exact form of guards is defined in Section 2. Let ξ be a finite set of guarded events. Then formula $\xi\ \mathsf{U}_\theta^x\ \phi$ holds at position i in a timed word provided j is the *first* position strictly after i with letter a satisfying the time constraint g and the formula ϕ holds at j with a freeze variable x remembering the time stamp of j. Moreover, all the positions strictly inbetween i and j must have events matching the set ξ. Note that ξ is a set of guarded events and not a formula. In this sense, the operator $\xi \mathsf{U}_\theta^x$ is unary and deterministic (requiring match with first occurrence of θ). The operator $\xi \mathsf{S}_\theta^x$ is the past (mirror image) version of $\xi \mathsf{U}_\theta^x$ and defined similarly. We show that every formula of logic $DUMTL$ can be reduced to *po2DTA* in linear time. Hence, satisfiability of $DUMTL$ is NP-complete. We also show that language of every *po2DTA* can be specified by a $DUMTL$ formula. The logic $DUMTL$ is deterministic or unambiguous in the sense that every timed word can be uniquely parsed to match a formula.

Next we explore expressiveness of various timed logics. The contrast between timed and untimed case should be noted. In untimed case, expressively, $UTL \equiv$ *UITL* $\equiv$ *po2dfa*. In order to complete the picture, we consider the unary version of the well known logic MTL and call it *UMTL*. By an example, we show that there are languages definable by *UMTL* which are not *po2DTA* recognizable. We also consider the unary fragment of logic MITL and call it *UMITL*. Again, by an example we show that there are *po2DTA* recognizable languages which are not definable using *UMITL*. Thus, unambiguity in timed regular languages seems to be captured only by the rather esoteric timed logic $DUMTL$ which is expressively distinct from the other timed logics listed above.

The rest of the paper is organized as follows. Section 2 defines the *po2DTA* and investigates their properties. Section 3 gives the unambiguous unary metric temporal logic, $DUMTL$ and shows its NP-complete satisfiability. Finally, Section 4 compares the expressiveness of unary fragments of MTL and $MITL$ with *po2DTA*. The paper ends with a short discussion.

2 Partially Ordered 2-Way Deterministic Timed Automata

Let Σ be a finite alphabet. A *timed word* w over Σ is a finite sequence $w = (\sigma_1, \tau_1), (\sigma_2, \tau_2)...(\sigma_m, \tau_m)$ of symbols $\sigma_i \in \Sigma$ paired with non-negative real numbers $\tau_i \in R$, such that the sequence $\tau_1, ...\tau_m$ is weakly monotonically increasing. We shall often represent w as $(\overline{\sigma}, \overline{\tau})$ with $\overline{\sigma} = \sigma_1, ...\sigma_m$ and $\overline{\tau} = \tau_1, ...\tau_m$ and let $untime((\overline{\sigma}, \overline{\tau})) = \overline{\sigma}$. For any real number τ let $Int(\tau)$ and $Fra(\tau)$ be the integral and fractional parts of τ. The set of all finite timed words over an alphabet Σ is denoted by $T\Sigma^*$. A timed language is a subset of $T\Sigma^*$. For 2-way automata it is convenient to enclose a timed word w between end markers. Let

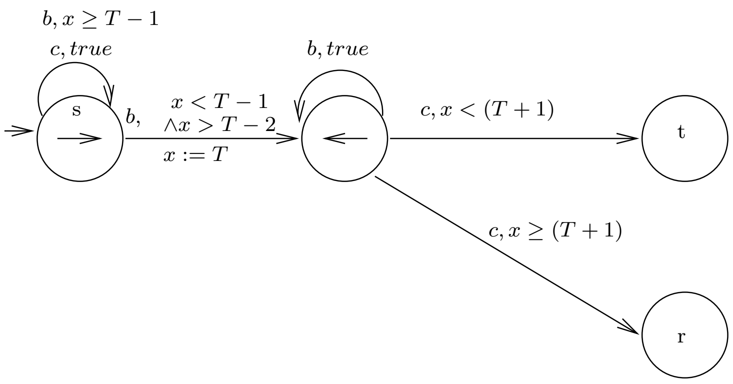

Fig. 1. Example of po2DTA

$\hat{w} = (\triangleright, 0)w(\triangleleft, \tau_{|w|})$ and let $\Sigma' = \Sigma \cup \{\triangleright, \triangleleft\}$. Also, let $dom(\hat{w}) = 0, 1, ...(|w|+1)$ be the set of all positions in the word $\hat{w}$. When clear from context, we shall abbreviate $\hat{w}$ by w.

A *two way finite state automaton* has a read-only head which scans the input word by reading the letter at its current head position i and the head can move in both directions (as in Turing machines). The textbook by Kozen [Koz97] provides a readable account of two way deterministic finite automata and their equivalence to 1-way DFA. Alur and Henzinger [AH92] generalized this to *2-way determinstic timed automata*, *2DTA*, which work over timed words. These timed automata are equipped with a finite set C of clocks (or registers) and in each transition a subset X of these can be reset to the value of the current time stamp under the head. The clocks retain their value till reset, thus they are like registers. As in timed automata, each transition is labelled with a letter and a guard. A guard g specifies a constraint on the values of the clocks and the current timestamp T. We label each state with a direction (left or right) and the head moves in direction specified by the *target* state of the transition. This form is more convenient for reduction to logics. (It can be shown that this form of automata are equivalent with a linear blowup in size to automata where the transitions carry the direction of head movement as in [AH92].) Each automaton is total, deterministic (see the formal definition below) and partially ordered. By *partial ordering of states* we mean that the only loops allowed are self-loops and once the automaton exits a state it can never return to it.

Example 1. Figure 1 shows an example po2DTA. Notation $x := T$ denotes that clock x is reset to current time stamp. This automaton accepts timed words with the following property: There is b in the interval $(1, 2)$ and c occurs before it. Moreover, if j is the position of the first b in the interval $(1, 2)$ and k is the position of the last c before it then $\tau_j - \tau_k < 1$.

We now give a formal definition of the syntax and semantics of *po2DTA*. Let G_C denote the set of guards over the clock set C. Let x range over C and c over integer constants. Let $\sim \in \{<, >, \leq, \geq, =\}$. The distinguished variable T denotes the current time. A guard g has the syntax:

$$g \in G_C \ ::= \ x \sim T + c \ | \ g_1 \wedge g_2 \ | \ \neg g$$

As usual, a clock valuation $\nu : C \rightarrow \Re_0$ assigns to each clock a non-negative real number. Let $\nu, \tau \models g$ denote that ν satisfies the guard g when T is assigned the real value τ. Also, let $\nu' = \nu \otimes (x \rightarrow \tau)$ denotes a valuation such that $\forall y \in C \ . \ y \neq x \Rightarrow \nu'(y) = \nu(y)$ and $\nu'(x) = \tau$. Two guards g_1 and g_2 are said to be disjoint if for all valuations ν and all τ, we have $\nu, r \models \neg(g_1 \wedge g_2)$. A special valuation ν_{init} maps all clocks to 0.

Definition 1 (Syntax of *po2DTA***).** *A po2DTA over alphabet Σ is a tuple $\mathcal{M} = (Q, \leq, \delta, s, t, r, C)$ where $(Q, \leq)$ is a partially ordered and finite set of states such that r, t are the only minimal elements and s is the only maximal element. Here, s is the initial state, t the accept state and r the reject state. Set C is a finite set of clocks. The set $Q \setminus \{t, r\}$ is partitioned into Q_l and Q_r (making the head move resp. left and the right on transitions leading into them) with $s \in Q_r$. Function $\delta : ((Q_l \cup Q_r) \times \Sigma' \times G_C) \rightarrow Q \times 2^C)$ is the transition function which satisfies the following conditions: Let $\delta(q, a, g) = (q', X)$. Then, $X \subseteq 2^C$ specifies the subset of clocks to be reset to the current time stamp. Moreover,*

- *(Partial-order) $q' \leq q$.*
- *(Reset on progress) If $q' < q$ (i.e. $q \neq q'$) then the transition is called a* progress transition *(or progress edge). A transition with $q = q'$ is called a self-loop. We allow resets only on progress edges, i.e. $X = \emptyset$ if $q = q'$.*
- *If $a = \triangleleft$ then $q' \in Q_l$ and if $a = \triangleright$ then $q \in Q_r$. This prevents the head from falling off the end-markers.*
- *(Determinism) For all $q \in Q$ and $a \in \Sigma'$, if there exist distinct transitions $\delta(q, a, g_1) = (q_1, X_1)$ and $\delta(q, a, g_2) = (q_2, X_2)$, then g_1 and g_2 are disjoint.*

Definition 2 (Run). *Let $w = (\sigma_1, \tau_1), (\sigma_2, \tau_2)...(\sigma_m, \tau_m)$ be a given timed word. The automaton actually runs on the word $\hat{w}$ which is w enclosed in end markers, as defined before. The configuration of a po2DTA at any instant is given by (q, ν, l) where q is the current state, ν is the the current clock valuation and $l \in dom(\hat{w})$ is the current head position. In this configuration, the head reads the letter σ_l and the time stamp τ_l.*

The run ρ of a po2DTA on the timed word w with starting head position k and starting clock valuation ν is the (unique) sequence of configurations $(q_1, \nu_1, l_1), ...,$ (q_n, ν_n, l_n) such that

- *Initialization: $q_1 = s$, $l_1 = k$ and $\nu_1 = \nu$. The automaton always starts in the initial state s.*
- *If the automaton is in a configuration (q_i, ν_i, l_i) and there exists a (unique) transition $\delta(q_i, a, g) = (p, X)$ such that $\sigma_{l_i} = a$ and $\nu_i, \tau_{l_i} \models g$. Then,*
  - *$q_{i+1} = p$*
  - *$\nu_{i+1}(x) = \tau_{l_i}$ for all clocks $x \in X$, and $\nu_{i+1}(x) = \nu_i(x)$ otherwise.*

- $l_{i+1} = l_i - 1$ *if* $p \in Q_l$
  $l_{i+1} = l_i + 1$ *if* $p \in Q_r$
  $l_{i+1} = l_i$ *if* $p \in \{t, r\}$

– *Termination:* $q_n \in \{t, r\}$. *The run is accepting if* $q_n = t$ *and rejecting if* $q_n = r$. *The configuration* (q_n, l_n, ν_n) *is called the final configuration and the head position* l_n *is called the accepting/rejecting position.*

Let $\mathcal{F}_{\mathcal{M}}$ *be a function such that* $\mathcal{F}_{\mathcal{M}}(w, i, \nu)$ *gives the final configuration* (q_n, l_n, ν_n) *of the unique run of* $\mathcal{M}$ *on* w *starting with the configuration* (s, i, ν)*. The language accepted by an automaton* $\mathcal{M}$ *is given by:*

$$\mathcal{L}(\mathcal{M}) = \{w \mid \mathcal{F}_{\mathcal{M}}(w, 1, \nu_{init}) = (t, i, \nu), \textit{for some } i, \nu\}.$$

2.1 Properties of *po2DTA*

We now discuss the properties of *po2DTA*. One nice property of our *po2DTA* is that all runs on a given word are of bounded length and the automaton cannot loop for ever.

Reversal Bounding. A reversal refers to the change in the direction of the head movement within a run of the automaton. Recall that in *po2DTA*, the head movement direction is determined by the direction of the target state of a transition. Hence, for a *po2DTA*, a reversal corresponds to a progress transition from a Q_l to a Q_r state, or vice versa: reversal can occur only on progress transitions. Due to partial ordering, we can have at most $(n-1)$ progress transitions during the run where $|Q| = n$, the number of states. Hence the number of reversals is also at most $(n-1)$. Moreover, the run on a word $\hat{w}$ can be at most of length $|w| \times (n-1)$.

Number of Clocks. Due to the partial order on the states, each progress edge in a *po2DTA* is traversed at most once in a given run of the automaton. Since a clock may be reset only on progress edges, there can be only $n-1$ number of resets on any run of the automaton. Hence, the number of clocks of a *po2DTA* can upper-bounded by $n-1$. Any automaton with more clocks will have clocks which are always equal and these can be removed. We now assume that the automaton has at most $n-1$ clocks.

Composition of Automata. We define some useful constructions on *po2DTA*. These also establish the closure properties of the language class *po2DTA*.

– Let *Acc* be a *po2DTA* that immediately accepts without moving the head. Thus, $\mathcal{F}_{Acc}(w, i, \nu) = (t, i, \nu)$. Similarly, let *Rej* be a *po2DTA* that immediately rejects without moving the head. Hence, $\mathcal{F}_{Rej}(w, i, \nu) = (r, i, \nu)$.
– Let *Start* be a *po2DTA* that scans leftward for the start end marker and then accepts after moving one step to the right. Thus, $\forall i \in dom(w)$ we have $\mathcal{F}_{Start}(w, i, \nu) = (t, 1, \nu)$. Similarly, let *End* be the automaton that accepts at the end of the word. Thus, $\forall i \in dom(w)$, we have $\mathcal{F}_{End}(w, i, \nu) = (t, |w|, \nu)$.

– Let $Freeze(x)$ be a *po2DTA* which accepts after assigning the current time to the clock x. Hence, $\mathcal{F}_{Freeze(x)}(w,i,\nu) = (t,i,\nu')$, where $\nu' = \nu \otimes (x \to \tau_i)$.
– Let $\mathcal{M}_1?\mathcal{M}_2 : \mathcal{M}_3$ be an automaton that executes M_1 first. If M_1 accepts then the execution continues with the run of M_2 from the final configuration of M_1. If M_1 rejects then the execution continues with the run of M_3. We omit the details of this simple construction. Let $\mathcal{M} = \mathcal{M}_1?\mathcal{M}_2 : \mathcal{M}_3$. Then the size (number of states) of $\mathcal{M}$ is equal to the sum of the sizes of $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$. Also, $\mathcal{F}_{\mathcal{M}}(w,i,\nu) = \mathcal{F}_{\mathcal{M}_2}(w,j,\nu')$ if $\mathcal{F}_{\mathcal{M}_1}(w,i,\nu) = (t,j,\nu')$ and $\mathcal{F}_{\mathcal{M}}(w,i,\nu) = \mathcal{F}_{\mathcal{M}_3}(w,j,\nu')$ if $\mathcal{F}_{\mathcal{M}_1}(w,i,\nu) = (r,j,\nu')$.
– Let $\mathcal{M}_1;\mathcal{M}_2 = \mathcal{M}_1?\mathcal{M}_2 : Rej$.

We state the following lemma without proof.

Lemma 1 (Boolean Closure). *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be po2DTA with disjoint clock sets.*

– *Let $\mathcal{M}_1 \vee \mathcal{M}_2 = \mathcal{M}_1?Acc : (Start;\mathcal{M}_2)$.*
  Then, $\mathcal{L}(\mathcal{M}_1 \vee \mathcal{M}_2) = \mathcal{L}(\mathcal{M}_1) \cup \mathcal{L}(\mathcal{M}_2)$.
– *Let $\mathcal{M}_1 \wedge \mathcal{M}_2 = \mathcal{M}_1?(Start;\mathcal{M}_2) : Rej$.*
  Then, $\mathcal{L}(\mathcal{M}_1 \wedge \mathcal{M}_2) = \mathcal{L}(\mathcal{M}_1) \cap \mathcal{L}(\mathcal{M}_2)$.
– *Let $\neg\mathcal{M}_1 = \mathcal{M}_1?Rej : Acc$. Then, $\mathcal{L}(\neg\mathcal{M}_1) = T\Sigma^* \setminus \mathcal{L}(\mathcal{M}_1)$.*

Note that in each case, the size of the constructed automaton exceeds the sum of sizes the component automata by only a constant factor. The construction can be carried out in time polynomial in the input. □

2.2 po2DTA: Decision Problems

Membership Since *po2DTA* are total and deterministic, there is a unique run ρ of the automaton on any word w. As stated before, the size of the run is upperbounded by $|w| \times (n-1)$ where n is the number of states. For every timed word w with rational timestamps, we may simulate this run on the automaton in time polynomial in the size of the word and number of states of the automaton, and decide whether or not the automaton accepts w. Thus, the membership problem for *po2DTA* is in P.

Non-emptiness. As discussed earlier, recollect that a *po2DTA* with n states has at most $(n-1)$ clocks. Given positive integers n, k, we define an equivalence $\approx_k^n$, using ideas similar to the region equivalence found in literature. Consider two sequences of timestamps $\overline{u} = u_1...u_n$ and $\overline{t} = t_1...t_n$. Define $\overline{u} \approx_k^n \overline{t}$ iff $\forall 0 \leq i,j < n$

– $t_i - t_j > k \Leftrightarrow u_i - u_j > k$
– $(u_i - u_j) \leq k \Rightarrow Int(u_i - u_j) = Int(t_i - t_j)$
– $Fra(u_i) < Fra(u_j) \Leftrightarrow Fra(t_i) < Fra(t_j)$
  $Fra(u_i) = Fra(u_j) \Leftrightarrow Fra(t_i) = Fra(t_j)$

Lemma 2. *Consider po2DTA $\mathcal{M}$ k_{max} as the maximum integer constant appearing on the guards. Consider two words $w = (\overline{\sigma}, \overline{u})$ and $w' = (\overline{\sigma}, \overline{t})$ of length p, with the same untimed word but possibly different timestamps. If $\overline{u} \approx^p_{k_{max}} \overline{t}$, then $w \in \mathcal{L}(\mathcal{M})$ iff $w' \in \mathcal{L}(\mathcal{M})$.*

Proof. (Outline) We can inductively observe that the run of $\mathcal{M}$ on both w and w' takes the same sequence of edges (transitions). Essentially, the guards compare the time distance between two timestamps with integer constants $\leq k_{max}$. Hence, if the configurations after i steps in runs on w and w' are (q, ν, l) and (q, ν', l) respectively, then for any guard g we have $\nu, u_l \models g$ iff $\nu', t_l \models g$ due to $\overline{u} \approx^p_{k_{max}} \overline{t}$. Hence, the same edge e is taken as $i + 1$'th transition in both runs. □

Lemma 3. *Given positive integers n, k_{max}, for all $\overline{u}$ of length $n-1$, there exists $\overline{t}$ of length $n-1$ such that $\overline{u} \approx^{n-1}_{k_{max}} \overline{t}$ and $\forall i$*

- *$Int(t_i) \leq (k_{max} + 1)(n - 1)$*
- *$Fra(t_i)$ is a multiple of $1/n$*

Proof. (Outline) The equivalence definition states that once difference between successive numbers in list $\overline{u}$ is greater than k_{max} their magnitude does not affect the equivalence. Hence, we can construct an equivalent sequence $\overline{t}$ where difference in timestamps between successive elements is at most $k_{max} + 1$. Also, only the relative ordering of fractional parts is important for equivalence. Since in each sequence there are at most $n - 1$ different numbers, the relative ordering between their fractional parts can be faithfully recorded using fractions which are multiple of $1/n$. □

Theorem 1 (Small-model property). *Given a po2DTA $\mathcal{M}$ with n number of states and k_{max} being the maximum integer appearing on the guards, if $\mathcal{L}(\mathcal{M})$ is non-empty, then there exists a timed word $w = (\sigma_1, \tau_1)...(\sigma_m, \tau_m) \in \mathcal{L}(\mathcal{M})$ such that*

- *$m < n$*
- *$\tau_m \leq (n - 1)(k_{max} + 1)$ and*
- *$\forall i \in dom(w)$, the fractional part of τ_i is a multiple of $1/n$*

Proof. Consider any word $w \in \mathcal{L}(\mathcal{M})$ (since $\mathcal{L}(\mathcal{M})$ is non-empty). Let w_1 be the subword of w consisting of only those positions which correspond to the progress transitions of $\mathcal{M}$. Since resets only occur at the progress transitions, it is easy to see that $w \in \mathcal{L}(\mathcal{M}) \Leftrightarrow w_1 \in \mathcal{L}(\mathcal{M})$. Since the number of progress transitions on any path from start to accept state is $< n$, we know that length of w_1 is $< n$. Let $w_1 = (\overline{\sigma}, \overline{u})$ and $w_2 = (\overline{\sigma}, \overline{t})$ such that $u \approx^{n-1}_{k_{max}} \overline{t}$ and $\forall i$

- $Int(t_i) \leq (k_{max} + 1)(n - 1)$
- $Fra(t_i)$ is a multiple of $1/n$

From Lemma 2, we know that such a word w_2 exists, and from Lemma 3, we know that $w_1 \in \mathcal{L}(\mathcal{M}) \Leftrightarrow w_2 \in \mathcal{L}(\mathcal{M})$. Hence, we have the word w_2 with the desired properties. □

Corollary 1 (Non-emptiness and Language Inclusion)

- *Non-emptiness of po2DTA is decidable with NP-Complete complexity.*
- *Language inclusion of po2DTA is decidable with Co-NP-Complete complexity.*

Proof. From Theorem 1, we know that a polynomial-sized word is included in the language of any *po2DTA* with non-empty language. This word can be guessed non-deterministically and checked for membership using the PTIME membership checking. Hence, non-emptiness of *po2DTA* is in NP. The non-emptiness of the (untimed) subclass *po2dfa* of *po2DTA* is already shown to be NP-complete [SP09] by giving a log-space reduction of satisfiability of CNF boolean formulae to non-emptiness of *po2dfa*. This immediately implies the NP-hardness of *po2DTA* non-emptiness.

Note that $\mathcal{L}(\mathcal{M}_1) \subseteq \mathcal{L}(\mathcal{M}_2)$ iff $\mathcal{L}(\neg(\mathcal{M}_2) \wedge \mathcal{M}_1) = \emptyset$. Since boolean operations on *po2DTA* automata are achieved in PTIME with linear increase in size by Lemma 1, and non-emptiness checking is NP-Complete, we have that language inclusion of *po2DTA* is in co-NP-Complete. □

3 *DUMTL*

In real-time logics, Freeze quantifiers are typically used in order to bind the time of occurrence of some events to variables[AH91]. Let Σ be the finite alphabet and X the finite set of freeze variables. We use the same syntax of guards G_X as in *po2DTA* to express time constraints. A *Guarded Event* over Σ, X is a pair $\theta = (a, g)$ such that $a \in \Sigma$ and $g \in G_X$ and a Guarded-Event-Set ξ over Σ, X is a finite set of guarded events. (A guarded event is just a singleton Guarded-Event-Set). A Guarded-Event-Function is a function in $\Sigma \rightarrow G_X$. For each Guarded-Event-Set ξ we define equivalent Guarded-Event-Function χ_ξ such that $\chi_\xi(a) = \vee\{g_i \mid (a, g_i) \in \xi\}$. Also, for a Guarded-Event-Function χ we can define a Guarded-Event-Set $\xi_\chi = \{(a, \chi(a)) \mid a \in \Sigma\}$. Thus, Guarded-Event-Set and Guarded-Event-Function are equivalent representations. Let $\top$ denote Guarded-Event-Set such that $\chi_\top(a) = true$. Given two Guarded-Event-Function χ_1 and χ_2, we define boolean operation on them in pointwise manner. E.g. $\forall a \in \Sigma.\ (\chi_1 \wedge \chi_2)(a) = \chi_1(a) \wedge \chi_2(a)$. This also allows us to carry out boolean operations on Guarded-Event-Set. This notational facility will be useful in constructing automata.

The logic *DUMTL* over Σ, X specifies properties of finite timed words using guarded events in its formulae. Let ϕ, ϕ_1, ϕ_2 range over *DUMTL* formulas, θ be any guarded event and ξ be any Guarded-Event-Set. The syntax of *DUMTL* formulas is as follows:

$$\phi := true \mid \theta \mid \xi \mathsf{U}^x_\theta \phi_1 \mid \xi \mathsf{S}^x_\theta \phi_1 \mid \neg\phi_1 \mid \phi_1 \vee \phi_2$$

Semantics. Let $\theta = (a, g)$, then $w, i \models_\nu \theta$ iff $(\sigma_i = a) \wedge (\nu, \tau_i \models g)$. Similarly, $w, i \models_\nu \xi$ iff $\nu, \tau_i \models \chi_\xi(\sigma_i)$. The remaining cases are defined inductively:

$w, i \models_\nu \xi \mathsf{U}^x_\theta \phi$ iff $\exists j > i. w, j \models \theta \wedge \forall i < k < j. w, k \models (\xi \wedge \neg\theta) \wedge w, j \models_{\nu'} \phi$ *where* $\nu' = \nu \otimes x \to \tau_j$.
$w, i \models_\nu \xi \mathsf{S}^x_\theta \phi$ iff $\exists j < i. w, j \models \theta \wedge \forall j < k < i. w, k \models (\xi \wedge \neg\theta) \wedge w, j \models_{\nu'} \phi$ *where* $\nu' = \nu \otimes x \to \tau_j$.
$w, i \models_\nu \phi_1 \vee \phi_2$ iff $w, i \models_\nu \phi_1$ or $w, i \models_\nu \phi_2$
$w, i \models_\nu \neg\phi_1$ iff $w, i \not\models_\nu \phi_1$

Notice that the U and S modalities are strict and no next/previous modalities are needed. The language of formula ϕ is given by $\mathcal{L}_\phi = \{w \mid w, 0 \models_{\nu_{init}} \phi\}$ where ν_{init} assigns the value 0 to each freeze variable.

Example 2. Let $\theta_1 = (b, x < (T-1))$ and $\theta_2 = (c, y = T+1)$. Then the formula $\top \mathsf{U}^y_{\theta_1}(\top \mathsf{S}_{\theta_2} true)$ defines the language of all timed words where there is a b after time 1, and there is a c exactly one time unit before the first occurrence of b after time 1.

3.1 From *DUMTL* to *po2DTA*

Consider a *DUMTL* formula ϕ. Our aim is to construct a *po2DTA* $\mathcal{M}(\phi)$ recognizing the language $L(\phi)$. The logic is deterministic. Hence, given a timed word w, in evaluating ϕ any subformula ψ of ϕ can be uniquely determined to be relevant or irrelevant. Moreover, if relevant, its value is needed only at a determined position in the word. We denote this position by pos^ψ_w. We construct a *po2dfa* $\mathcal{R}(\psi)$, called the ranker of ψ such that, starting with any position, $\mathcal{R}(\psi)$ accepts at position pos^ψ_w if the formula is relevant and rejects if it is irrelevant. The ranker actually depends not on the subformula ψ but the context λ such that $\phi = \lambda(\psi)$. Its inductive construction is given below. Let $\mathcal{A}_{\mathsf{U}}(\xi, \theta, x)$ and $\mathcal{A}_{\mathsf{S}}(\xi, \theta, x)$ be *po2DTA* as shown in Figure 2 which scan forwards (or backwards respectively) accepting at the first (or last) θ provided all intermediate positions are events in ξ.

- $\mathcal{R}(\phi) = Start$
- If $\psi = \psi_1 \vee \psi_2$ then $\mathcal{R}(\psi_1) = \mathcal{R}(\psi_2) = \mathcal{R}(\psi)$.
  Also, if $\psi = \neg\psi_1$ then $\mathcal{R}(\psi_1) = \mathcal{R}(\psi)$.
- $\psi = \xi \mathsf{U}^x_\theta \psi_1$ then $\mathcal{R}(\psi_1) = \mathcal{R}(\psi); \mathcal{A}_{\mathsf{U}}(\xi, \theta, x)$
- $\psi = \xi \mathsf{S}^x_\theta \psi_1$ then $\mathcal{R}(\psi_1) = \mathcal{R}(\psi); \mathcal{A}_{\mathsf{S}}(\xi, \theta, x)$

Given rankers $R(\psi)$ for each subformula ψ, we can construct *po2DTA* automaton $\mathcal{M}(\psi)$ in a bottom-up fashion so that the automaton accepts if ψ is relevant and evaluates to true at its relevant position. A fresh clock is introduced each time we assign a freeze quantifier.

- If $\psi = \theta$ then $\mathcal{M}(\psi) = \mathcal{R}(\psi); \mathcal{A}(\theta)$ where $\mathcal{A}(\theta)$ accepts immediately if θ evaluates to true at current position and rejects immediately otherwise.
- If $\psi = \xi \mathsf{U}^x_\theta \psi'$ then $\mathcal{M}(\psi) = \mathcal{R}(\psi); \mathcal{A}_{\mathsf{U}}(\xi, \theta, x); \mathcal{M}(\psi')$
- If $\psi = \xi \mathsf{S}^x_\theta \psi'$ then $\mathcal{M}(\psi) = \mathcal{R}(\psi); \mathcal{A}_{\mathsf{S}}(\xi, \theta, x); \mathcal{M}(\psi')$
- If $\psi = \psi_1 \vee \psi_2$ then $\mathcal{M}(\psi) = \mathcal{M}(\psi_1) \vee \mathcal{M}(\psi_2)$
- If $\psi = \neg\phi$ then $\mathcal{M}(\psi) = \neg(\mathcal{M}(\phi))$

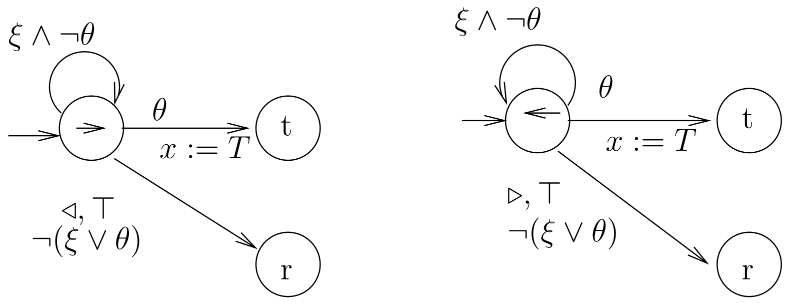

Fig. 2. Automata for $\mathcal{A}_{\mathsf{U}}(\xi, \theta, x)$ and $\mathcal{A}_{\mathsf{S}}(\xi, \theta, x)$

We state the following lemma without proof.

Lemma 4. *Given a DUMTL formula ϕ, the above construction gives an equivalent po2DTA $\mathcal{M}(\phi)$ such that $\forall w \in T\Sigma^* . w \in \mathcal{L}(\phi)$ iff $w \in \mathcal{L}(\mathcal{M}(\phi))$.*

3.2 From *po2DTA* to *DUMTL*

We now give a translation from an automaton $\mathcal{M}$ to a language equivalent *DUMTL* formula $\phi_{\mathcal{M}}$. Since the automaton works on words $\hat{w} = (\rhd, 0)w(\lhd, \tau_{|w|})$, we shall extend the alphabet of the *DUMTL* formulas to include end-markers: $\Sigma' = \Sigma \cup \{\rhd, \lhd\}$. For each clock in the automaton, we have a freeze variable in

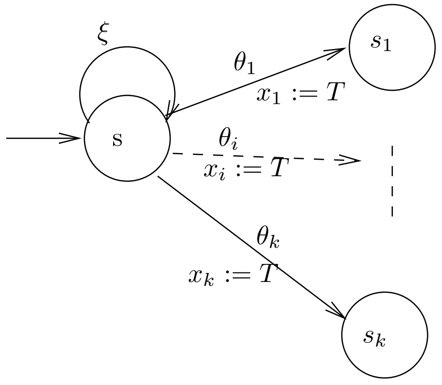

Fig. 3. A typical state of *po2DTA*with incoming and outgoing transitions

the formula. Consider a state s in $\mathcal{M}$ as shown in Figure 3. We will construct a formula ψ_s satisfying the following lemma which we give here without proof.

Lemma 5. *For any state s of $\mathcal{M}$ and for any word w if $\mathcal{M}$ in its partial run executes a progress transition e with head at position p and enters the configuration (s, p', ν) then $w, p \models_\nu \psi_s$ iff there is an accepting run of $\mathcal{M}$ on w from the configuration (s, p', ν)*

We inductively construct ψ_s for each state as follows:

- $\psi_t = \top$ and $\psi_r = \bot$
- Given ψ_{s_i} for all $1 \leq i \leq k$ (as in the figure), then
  - If $s \in Q_r$, $\psi_s := \vee_{1 \leq i \leq k}(\xi \mathsf{U}^{x_i}_{\theta_i}(\psi_{s_i}))$
  - If $s \in Q_l$, $\psi_s := \vee_{1 \leq i \leq k}(\xi \mathsf{S}^{x_i}_{\theta_i}(\psi_{s_i}))$

For the formula ψ_s corresponding to the start state s, we assume that there is a dummy incoming edge on the left end-marker, for which s is the target state. Hence, the *DUMTL* formula ψ_s must be evaluated from the position 0.

Theorem 2. *Given a po2DTA $\mathcal{M}$, there is an equivalent DUMTL formula $\psi_{\mathcal{M}} = \psi_s$ (where s is the start state) such that $\forall w \in T\Sigma^*$, $\mathcal{M}$ accepts w iff $\hat{w}, 0 \models_{\nu_{init}} \psi_{\mathcal{M}}$, where ν_{init} assigns 0 to each freeze variable.*

4 Unary Metric Temporal Logics and Expressiveness

For untimed words, it was shown by [EVW02] that Unary Temporal Logic (UTL) is expressively equivalent to po2DFA. Here, we add timing constraints to the temporal modalities of UTL to get logic *UMTL*.

Consider finite timed words over a finite alphabet, Σ. Let ϕ, ϕ_1, ϕ_2 range over *UMTL* formulae and $a \in \Sigma$. An interval I is a convex subset of non-negative reals with integral end-points. It may be open, half-open or closed. Such an interval may be represented as $I =< i, j >$ in general. Here j can even be ∞ specifying infinite right open interval. If $i = j$, then the interval is said to be singular. It is non-singular if $i < j$. The syntax of *UMTL* is as follows:
$\phi := a \mid \overrightarrow{\Diamond}_I \phi \mid \overleftarrow{\Diamond}_I \phi \mid \phi_1 \vee \phi_2 \mid \neg\phi$
Let $w = (\sigma_1, \tau_1), ...(\sigma_m, \tau_m)$ be a finite timed word over Σ. *UMTL* formulas are interpreted over a position in the timed word. We give the semantics of *UMTL* formulas as follows.

$$\begin{aligned}
&w, i \models a \text{ iff } \sigma_i = a\\
&w, i \models \overrightarrow{\Diamond}_I \phi \text{ iff } \exists j > i. \tau_j \in \tau_i + I \wedge w, j \models \phi\\
&w, i \models \overleftarrow{\Diamond}_I \phi \text{ iff } \exists j < i. \tau_j \in \tau_i - I \wedge w, j \models \phi\\
&w, i \models \phi_1 \vee \phi_2 \text{ iff } w, i \models \phi_1 \vee w, i \models \phi_2\\
&w, i \models \neg\phi \text{ iff } w, i \not\models \phi
\end{aligned}$$

4.1 Punctuality and Expressiveness

The logic *UMTL* allows singular or punctual intervals in its formulae. Let *UMITL* be the subclass of *UMTL* where all intervals are syntactically required to be non-punctual. In context of MITL, Alur and Henziger [AH92] have shown with examples that punctuality is a property that cannot be expressed by two-way deterministic TA even without any reversal bounds.

Theorem 3. *There is no po2DTA which recognizes the language defined by the UMTL formula $\phi := \overrightarrow{\Diamond}_{(0,1)}(a \wedge \overrightarrow{\Diamond}_{[3,3]} c)$.*

Proof. Let us assume $\mathcal{M}_\phi$ is a *po2DTA* which recognizes $\mathcal{L}(\phi)$, and $\mathcal{M}_\phi$ has n states.
Consider a word w such that $untime(w) = a^{2n+1}c^{2n+1}$ such that all the a's are in the interval $(0, 1)$ and $\overrightarrow{\Diamond}_{[3,3]}c$ holds only for the $(n + 1)^{th}$ a. In order to recognize the timestamp of this a, there must be a progress transition on this a (because resets are allowed only on progress transitions). However, if the $(n+1)^{th}$ a within the unit interval $(0, 1)$ is on a progress transition, it must imply that the other n a's before (or after) are also on progress transitions since the automaton is deterministic. But the automaton $\mathcal{M}_\phi$ may have at most $(n - 1)$ progress transitions on a given run. Hence, the assumption that $\mathcal{M}_\phi$ recognizes the language is false. □

Theorem 4. *The automaton in the figure below does not have an equivalent UMITL formula.*

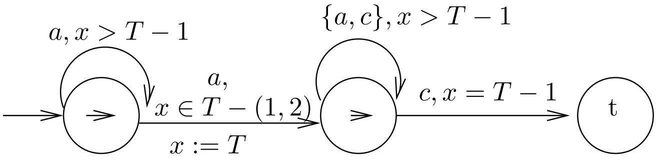

Fig. 4. Expressiveness of *UMITL*

Proof (Outline). The following property holds for *UMITL* formulas. We omit its detailed proof. Given a *UMITL* formula ϕ there exists $\delta > 0$ such that for any word $w = (\overline{\sigma}, \overline{t}) \in \mathcal{L}(\phi)$, there exists a sequence of open intervals $\overline{I}$ such that each I_i is of width δ and $t_i \in I_i$ and for any sequence $\overline{u}$ formed with $u_i \in I_i$, we have $w' = (\overline{\sigma}, \overline{u}) \in \mathcal{L}(\phi)$. Informally, it states that time stamps of models of a *UMITL* formula can be perturbed in a small neighbourhood preserving truth. Now, the automaton ($\mathcal{M}$) above accepts all words such that for the first a in the interval $(1, 2)$, there is a c exactly one time unit after it. A word in $\mathcal{L}(\mathcal{M})$ with only one c in it, will not satisfy the perturbation property of *UMITL* formulas stated above. Hence, there is no *UMITL* formula equivalent to $\mathcal{M}$. □

5 Discussion

Unambiguous languages [Sch75] are a robust subclass of regular languages with diverse characterizations. The *po2dfa* give an automaton based characterization of this class[STV01] and the unary temporal logic, UTL, which is a temporal logic using only the modalities $\overrightarrow{\Diamond}$ and $\overleftarrow{\Diamond}$ provides a logical characterization of this class [EVW02]. In our past work [LPS08, LPS10] we have shown that deterministic temporal logics admitting unique parsability can be defined for unambiguous languages. The deterministic logics have intimate connections with the automata for unambiguous languages and this results into efficient satisfiability checking of determinstic logic formulae. In this paper, we carry over the same approach to much more challenging setting of timed languages.

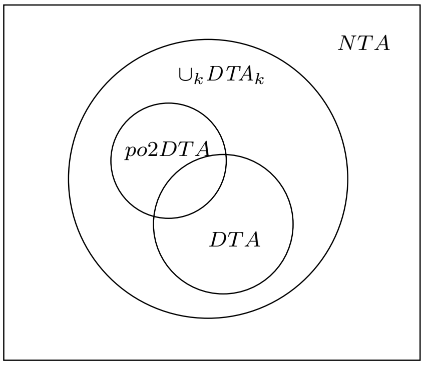

Fig. 5. Showing Expressiveness of po2DTA

In this paper, we have initiated the study of timed unambiguous languages (TUL). We have chosen to specify this class by defining partially ordered two way deterministic timed automata, *po2DTA*, which recognize TUL. We have also come up with a temporal logic called deterministic, unary metric temporal logic, *DUMTL*, which is expressvily complete for *po2DTA*. Using the reduction from logic to automata we have shown that the satisfiability of *DUMTL* is NP-complete.

Alur and Henzinger [AH92] defined *2DTA*. The subclass of *2DTA* with at most k reversals is called DTA_k. The subclass $\cup_k DTA_k$ of reversal bounded 2DTA is boolean closed and the emptiness of DTA_k with fixed reversal bound k is PSPACE-complete. Alur and Henzinger also showed that logic $MITL[\mathsf{U}_I, \mathsf{S}_I]$ can be reduced to $\cup_k DTA_k$ giving EXPSPACE satisfiability for the logic.

Our *po2DTA* are a subclass of $\cup_k DTA_k$. They have many pleasant properties: they are boolean closed (with only polynomial blowup), their non-emptiness is NP-Complete and the language inclusion between automata is Co-NP-Complete. Moreover, an automaton with n states and arbitrarily many clocks can be reduced to an automaton with at most n-1 clocks. Expressively, *po2DTA* are a strict subset of $\cup_k DTA_k$ and unrelated to the deterministic timed automata (DTA). The exact expressiveness is depicted diagrammatically in Figure 5.

For untimed case, unambiguous languages correspond to unary temporal logic [EVW02]. The generalization of this to timed case turns out to be tricky. The unary temporal logic is obtained by replacing the U and the S by unary operators $\overrightarrow{\Diamond}$ and $\overleftarrow{\Diamond}$. Unfortunately, this method does not yield timed unambiguous languages. In the paper, we have considered the unary version of logic MTL and called it *UMTL*. We also considered the unary fragment of MITL and called it *UMITL*. In Section 4, we were able to show with examples that *UMTL* is not contained within *po2DTA* and that *po2DTA* are not contained within *UMITL*. Hence, neither of these logics characterize the class of timed unambiguous languages. Our deterministic and unary logic *DUMTL* seems to adequately capture the essence of unambiguity in timed regular languages. In our opinion, the efficient satisfiability checking of *DUMTL* is a direct consequence of this connection. The exact relationship between the expressive powers of logics *DUMTL*, *UMTL* and *UMITL* is a topic of our ongoing investigation.

References

[AD91] Alur, R., Dill, D.: A Theory of Timed Automata. Theo. Comp. Sc. 126(2), 183–235 (1994)

[AH91] Alur, R., Henzinger, T.A.: Logics and Models of Real Time: A Survey. In: Huizing, C., de Bakker, J.W., Rozenberg, G., de Roever, W.-P. (eds.) REX 1991. LNCS, vol. 600, pp. 74–106. Springer, Heidelberg (1992)

[AFH96] Alur, R., Feder, T., Henzinger, T.A.: The Benefits of Relaxing Punctuality. J. ACM 43(1), 116–146 (1996)

[AH92] Alur, R., Henzinger, T.: Back to the Future: Towards a Theory of Timed Regular Languages. In: FOCS 1992, pp. 177–186 (1992)

[Koz97] Kozen, D.: Automata and Computability. Springer, Heidelberg (1997)

[DGK08] Diekert, V., Gastin, P., Kufleitner, M.: A survey on small fragments of first-order logic over finite words. Int. J. Found. Comp. Sci. 19(3), 513–548 (2008)

[EVW02] Etessami, K., Vardi, M.Y., Wilke, T.: First-order logic with two variables and unary temporal logic. Inf. Comput. 179, 279–295 (2002)

[LPS08] Lodaya, K., Pandya, P.K., Shah, S.S.: Marking the chops: an unambiguous temporal logic. In: Ausiello, G., Karhumäki, J., Mauri, G., Ong, L. (eds.) Proc. 5th IFIP TCS, Milano. IFIP Series, vol. 273, pp. 461–476 (2008)

[LPS10] Lodaya, K., Pandya, P.K., Shah, S.S.: Around dot-depth two. In: Yu, S. (ed.) Proc. 14th DLT, London, Canada. LNCS (2010)

[OW05] Ouaknine, J., Worrell, J.: On the Decidability of Metric Temporal Logic over finite words. LMCS 3(1) (2007)

[Sch75] Schützenberger, M.-P.: Sur le produit de concaténation non ambigu. Semigroup Forum 13, 47–75 (1976)

[STV01] Schwentick, T., Thérien, D., Vollmer, H.: Partially-ordered two-way automata: a new characterization of DA. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 239–250. Springer, Heidelberg (2002)

[SP09] Shah, S.S., Pandya, P.K.: An automaton normal form for UITL, Technical Report STCS-TR-SP-2009/1. Computer Science Group, TIFR (2009)

[TW98] Thérien, D., Wilke, T.: Over words, two variables are as powerful as one quantifier alternation: $FO^2 = \Sigma_2 \cap \Pi_2$. In: Proc. STOC, Dallas, pp. 41–47 (1998)