Real Time Temporal Logic: Past, Present, Future

Oded Maler, Dejan Nickovic, Amir Pnueli

VERIMAG, NYU, Weizmann

2005

Technical Content

No new original technical results (the importance of “results” is exaggerated in certain circles)

Simple proofs of two folk theorems about the real-time temporal logic MITL:

1) All languages specified by Past MITL formulae are accepted by deterministic timed automata

2) Some languages specified by Future MITL formulae are not accepted by any deterministic timed automaton.

An explanation of why this is the case

Untimed Case: Summary

Future LTL denotes star-free (aperiodic) ω-regular sets (infinite words)

From φ to a non-deterministic Buchi automaton (NBA), either directly by tableau or indirectly via AFA and $\forall$-determinization

From NBA apply NcNaughton-Safra to obtain a deterministic Rabin automaton

Past LTL denotes star-free (aperiodic) regular sets over finite words

Admits a direct construction from a formula to a deterministic automaton

Every future LTL formula can be written as Boolean combination of $\Box\Diamond\varphi$ where φ is a past formula (normal form) [LichtensteinPnueliZuck85]

An algorithm to translate any counter-free automaton (or ω-automaton) into a past LTL (or normal form) formula [MalerPnueli90]

Dense/Metric Time

Machine: timed automaton [AlurDill], TPN, event-recording automaton, event-clock automaton,

Timed regular expressions [AsarinCaspiMaler97]

Logics: many were developed 80-90s

Modal: [Pnueli, Manna, Alur, Henzinger, ...]
First/second order: [Wilke, ... , Rabinovich, Hirshfeld, ... Lamport]

MITL [AlurFederHenzinger96], a restriction of MTL [Koymans90] to interval modalities

$\diamond_{[a,b]}\varphi$: φ will hold within $t \in [a, b]$ time from now

MITL is equivalent to event-clock logic [RaskinSchobbensHenzinger98]. MITL is decidable and admits a hierarchy based on alternation of past and future [AlurHenzinger92]

Determinism

Why the obsession with deterministic automata?

Classical untimed automata theory is very deterministic

Every regular set admits a deterministic finite acceptor

This acceptor is canonical for the language (Myhill-Nerode)

The theory of timed languages is still unclean compared to the classical theory [Trakhtenbrot95, Asarin03]

There is no agreement on what the analogue of regular/rational languages is

Our recent attempt: recognizable languages [MalerPnueli04] a kind of algebraic characterization that coincides with languages accepted by input-deterministic timed automata

Motivation and concise history for this work

Motivation: find a syntactic characterization of the recognizable/deterministic timed languages. Semi practical motivation: deterministic formalism are easier to monitor

1) Finding a proof of the determinism of Past MITL (source of optimism)

2) Proving that this does not hold for future MITL (blow to optimism)

3) Seeing that this does not hold also for star-free timed regular expressions (total despair)

4) Understanding why (some comfort)

Finitary Interpretation of LTL/MITL

Remove the asymmetry between finite past and infinite future so that we can focus on differences due to direction of modalities

We interpret future temporal logic over finite words/signals and get rid of all the ω-complications

Finitary interpretation have recently become popular due to runtime verification/monitoring/testing: decide whether a given ξ satisfies a property

Not easy (for mortals, computers included) to observe infinite inputs..

Finitary interpretations of LTL proposed by [EisnerFisman et al03]: “truncated paths”, “weak” interpretation

Main issue is how to define propositional satisfaction at (ξ, t) where t is outside the scope of ξ. Can be solved this way or another – we restrict to bounded modalities

The Logic

Interpreted over finite signals $\xi : \mathbb{T} \to \mathbb{B}^n$ defined over an interval $I = [0, r)$

Standard temporal logic definitions ...

Past modality: Since

$$(\xi, t) \models \varphi_1 \mathcal{S}_{[a,b]} \varphi_2 \leftrightarrow \exists t' \in t \ominus [a, b]\ (\xi, t') \models \varphi_2 \text{ and } \forall t'' \in [t', t], (\xi, t'') \models \varphi_1$$

Future modality: Until

$$(\xi, t) \models \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \leftrightarrow \exists\, t' \in t \oplus [a, b]\ (\xi, t') \models \varphi_2 \text{ and } \forall t'' \in [t, t'], (\xi, t'') \models \varphi_1$$

Derived operators: sometime/always in the past, eventually/always in the future $\Box$, $\Diamond$, $\boxminus$ $\diamondminus$

Satisfaction of a formula by a signal $\xi \models \varphi$ is defined as
$(\xi, 0) \models \varphi$ forward from zero for future formulae
$(\xi, |\xi|) \models \varphi$ backward from the end for past formulae

The Automata

Variation on “standard” timed automata:

Reads multi-dimensional dense-time Boolean signals. Alphabet letters are associated with states rather than with transitions

Acceptance conditions include constraints on clock values

Clock values may include the special symbol $\perp$ indicating that the clock is currently inactive

Transitions can be labeled by the usual resets of the form $x := 0$ or $x := \perp$ as well as by copy assignments of the form $x_i := x_j$

Determinism: two states associated with the same input letters have disjoint staying conditions. Every signal admits a unique run

From Past MITL to DTA

Automata are built compositionally like in [Pnueli03] for future LTL

The automaton for a formula observes the states of the automata that correspond to its immediate sub-formulae

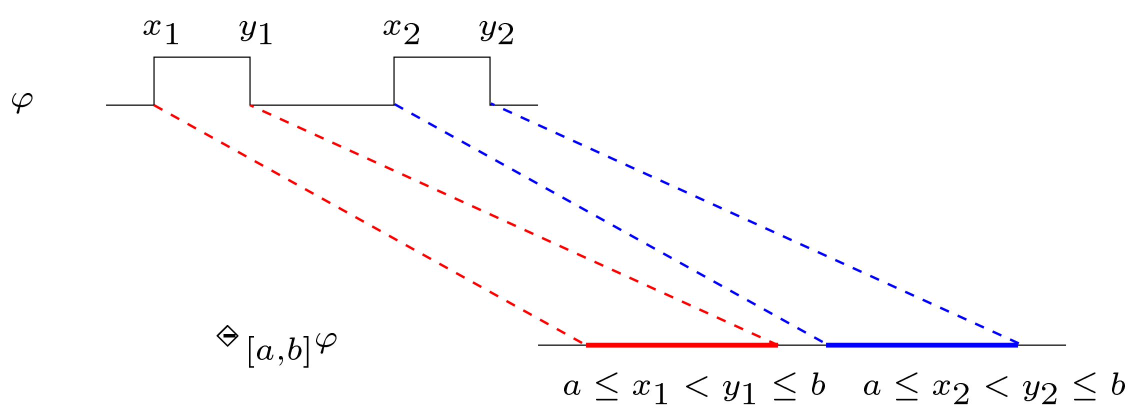
The automaton for a past formula φ is in an accepting state at time t exactly when the input signal read until t satisfies φ

The essence of the construction is the automaton for $\diamondminus_{[a,b]}\varphi$, the **event recorder**

The event recorder for φ observes the value of φ and outputs true exactly at every t such that φ was true in $t \ominus [a, b]$

The Basic Idea I

When φ become true in the i^{th} time we reset a clock x_i and when it becomes false we reset clock y_i. Formula $\diamondminus_{[a,b]}\varphi$ is true whenever $a \leq x_i < y_i \leq b$ for some i



How to reduce the number of clocks? When $y_1 > b$ we can kill both x_1 and y_1 and "shift" all clocks ($x_i := x_{i+1}$, $y_i := y_{i+1}$)

Now x_1 represents the oldest event still "alive" in the system

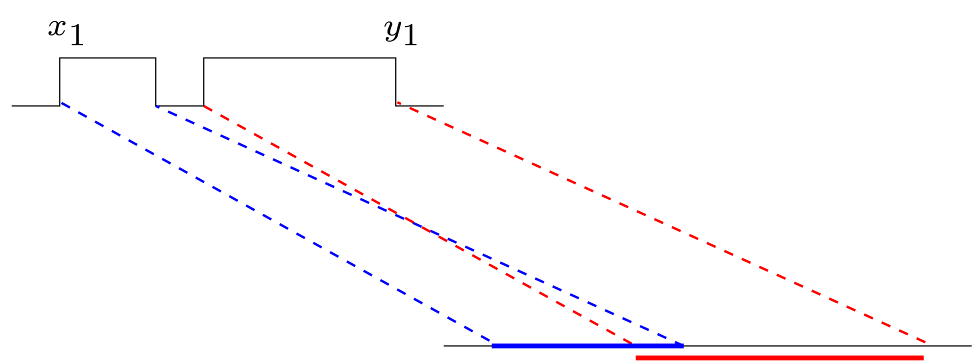
Not sufficient because φ can change unboundedly until $x_1 = b$

The Basic Idea II

If φ is false for less than $b-a$ time then

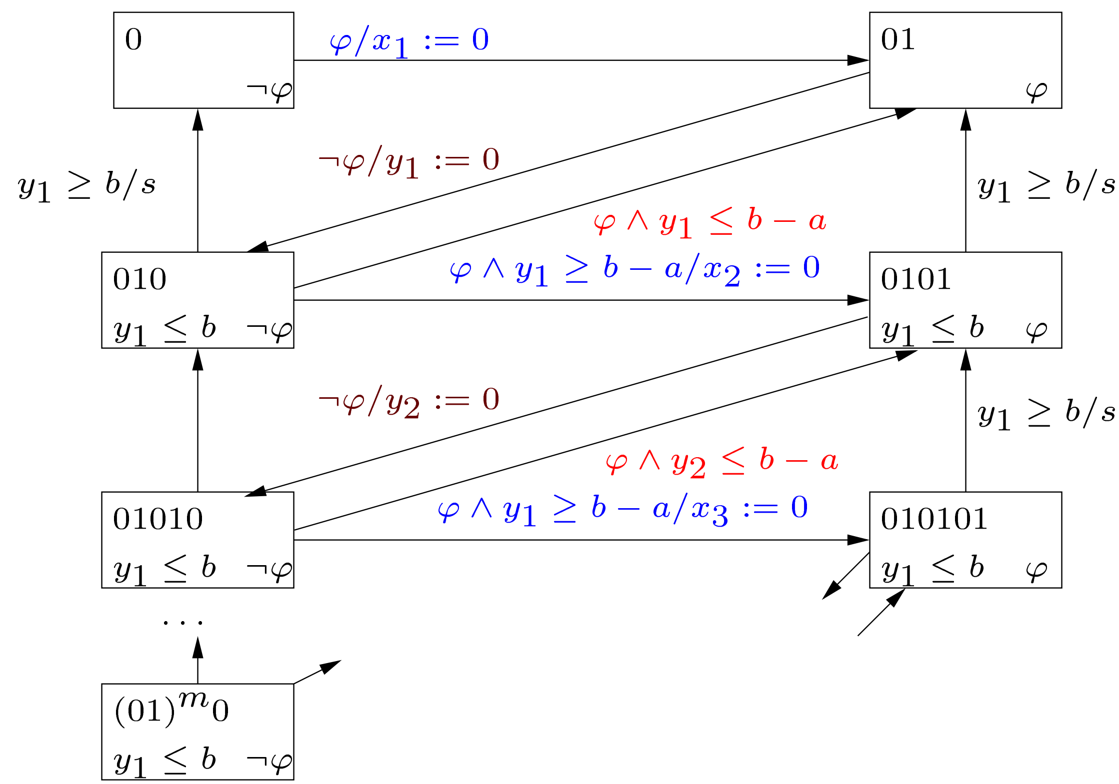
$(x_1 \geq a \wedge y_1 \leq b) \vee (x_2 \geq a \wedge y_2 \leq b)$ iff $x_1 \geq a \wedge y_2 \leq b$

We can kill y_1 and x_2 which is like ignoring/forgetting the short false episode



At most $m = b/(b-a)-1$ true-episodes should be recorded before x_1 reaches b and $2m$ clocks suffice to memorize their timing

The Event Recorder

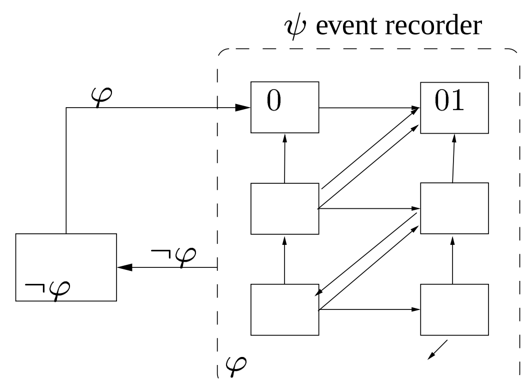


Acceptance: $a \leq x_1$

Automaton for $\varphi \mathcal{S}_{[a,b]} \psi$

Formula $\varphi \mathcal{S} \psi$ is like $\diamondminus \psi$ and φ holds continuously since then

The automaton for $\varphi \mathcal{S}_{[a,b]} \psi$ is an event recorder for ψ with an additional state for $\neg\varphi$



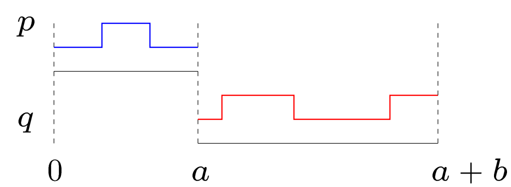
Corollary: we can build a deterministic timed automaton for any past MITL formula

And now to the Sad Part

We demonstrate a timed language L, definable in future MITL, not accepted by any deterministic automaton. Consider the formula

$$\varphi : \quad \Box_{[0,a]}(p \Rightarrow \Diamond_{[a,b]} q)$$

Let L consist of all $p-q$-signals of length $a+b$ that satisfy φ, that is, maintain some relation between the times p holds in $[0,a]$ and times when q holds in $[a, a+b]$



The automaton reads first the p part and memorizes what is required in order to determine whether the q part is accepted

How to Prove Non-Detrminizability

The syntactic (Nerode) right-congruence $\sim$ associated with a language L is:

$$u \sim v \text{ iff } \forall w \; u \cdot w \in L \Leftrightarrow v \cdot w \in L$$

Two prefixes are equivalent if they “accept” the same suffixes

For untimed languages, regularity (and acceptance by a deterministic finite automaton) is equivalent to $\sim$ having a finite index

For timed languages [MalerPnueli04] replace finiteness by some kind of boundedness which implies:

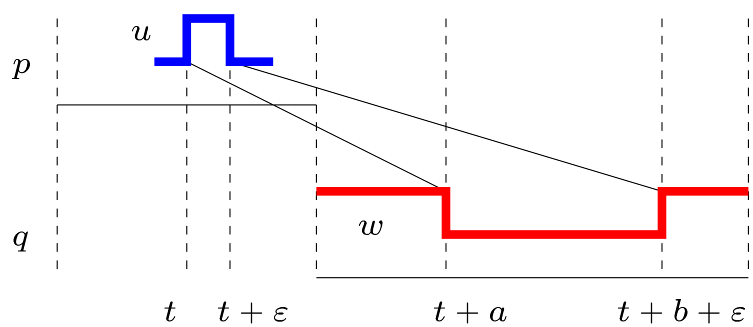
If a timed language is deterministic then there is some n such that every signal with n changes is $\sim$-equivalent to a signal with less than n changes

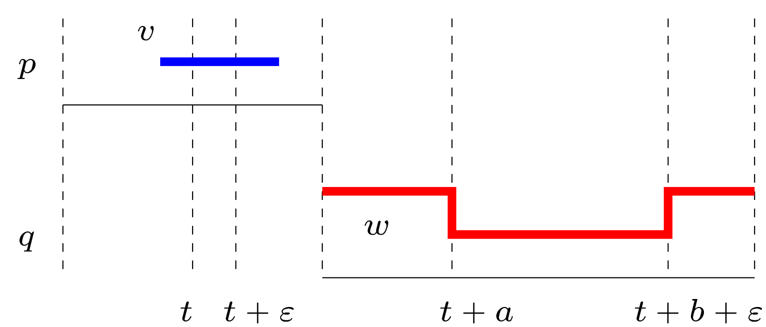
Demonstration

We show that L does not have that property and every two different p-signals are not Nerode-equivalent

Let u and v be two different p-signals and assume p is true on $[t, t+\varepsilon]$ in u and false in v

We construct a q-signal w such that $u \cdot w \notin L$ and $v \cdot w \in L$





For this formula you need to remember everything

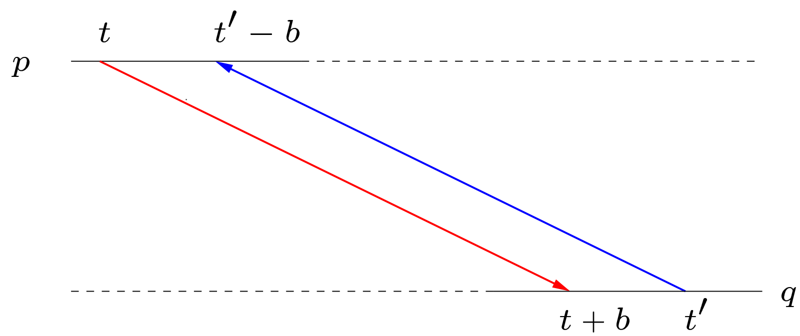
Why?

Why a past formula can forget short episodes and a future formula cannot?

Consider first a "punctual" version of the bad formula, where q should follow exactly b time after p, and its past "dual"

$$\Box_{[0,a]}(p \Rightarrow \Diamond_b q) \quad = \quad \boxminus_{[0,a]}(\neg q \Rightarrow \diamondminus_b \neg p)$$

$$\forall t \in [0,a]\ p[t] \Rightarrow q[t+b] \qquad \forall t' \in [b, b+a]\ \neg q[t'] \Rightarrow \neg p[t'-b]$$

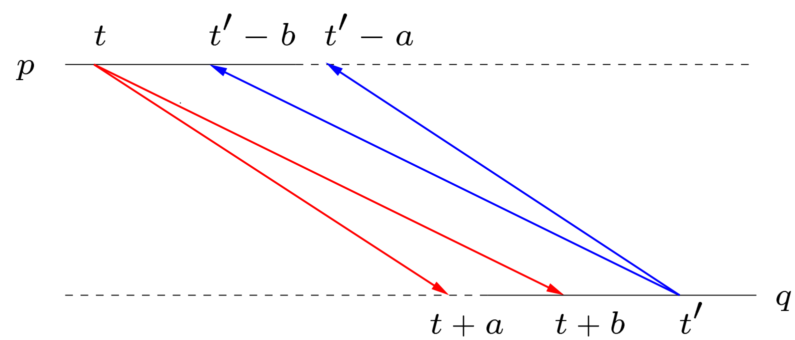


Relaxing Punctuality

When we use interval modalities we create an asymmetry:

Future MITL: a small-duration event in the past creates obligations for a large time interval in the future

Past LTL: a small-duration event in the future is implied by something that happened somewhere inside a large past interval



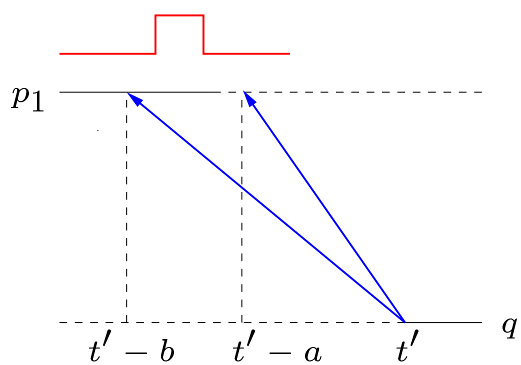
Logically Speaking

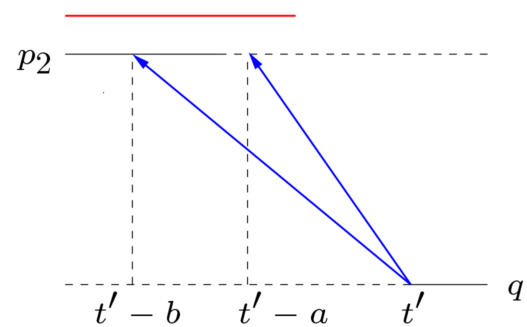
The “equivalent” past formula $\boxminus_{[0,a]}(\neg q \Rightarrow \Diamond\!\!\!-_{[a,b]} \neg p) \quad =$

$\forall t' \in [b, a+b]\ (\neg q[t'] \Rightarrow \exists t \in t' \ominus [a,b]\ \neg p[t]) \quad =$

$\forall t' \in [b, a+b]\ ((\forall t \in t' \ominus [a,b]\ p[t]) \Rightarrow q[t'])$

Cannot distinguish between p_1 with a short positive episode and p_2 without





And that’s it

Conclusions (and Future)

We hopefully explained an intriguing phenomenon which turns out to be a result of a syntactical accident

It is worth mentioning the inertial delay operator used in hardware timing, and formalized using timed automata by [MalerPnueli95]

This operator also "filters" small fluctuations in the signal

We can require events that imply toward the future to persist some minimal duration

The following "inertial" version of the bad formula, is deterministic

$$\square_{[0,a]}((\square_{[0,b-a]}p) \Rightarrow (\Diamond_{[a,b]}q))$$

Bonus: Results on Star-free Timed Regular Expressions

Theorem: some (but unfortunately not all) timed languages denoted by timed star-free expressions are deterministic

The future language:

$$\neg(U \cdot (p \cdot U \wedge \neg(\langle U \cdot q\rangle_{[a,b]} \cdot U)))$$

The past language:

$$\neg(U \cdot (U \cdot p \wedge \neg(U \cdot \langle q \cdot U\rangle_{[a,b]})) \cdot U)$$

U is a special symbol denoting the universal timed language