Metric Interval Temporal Logic Revisited

MOVE seminar

Benjamin Monmege (LIF, Aix-Marseille Université)

Based on joint works with
Thomas Brihaye, Hsi-Ming Ho, Morgane Estiévenart (UMONS),
Gilles Geeraerts (ULB), and Nathalie Sznajder (LIP6)

30/03/2017

Timed systems

Timed systems

- Events (*MoveUp*, *MoveDown*, *OpenDoor*...)

Timed systems

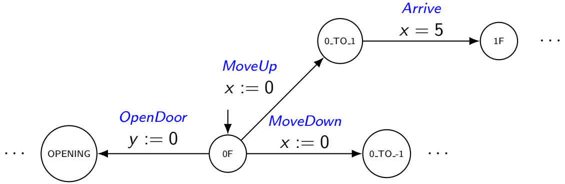
- Events (*MoveUp*, *MoveDown*, *OpenDoor*...)
- States (at which floor, opened/closed...)

Timed systems

- Events (*MoveUp*, *MoveDown*, *OpenDoor*...)
- States (at which floor, opened/closed...)
- *Timings* (operation time, latency...)

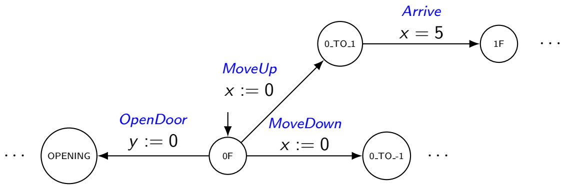
Timed systems

- Events (*MoveUp*, *MoveDown*, *OpenDoor*...)
- States (at which floor, opened/closed...)
- *Timings* (operation time, latency...)



Timed systems

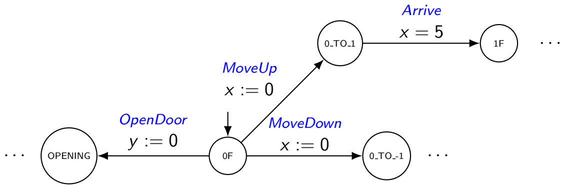
- Events (*MoveUp*, *MoveDown*, *OpenDoor*...)
- States (at which floor, opened/closed...)
- *Timings* (operation time, latency...)



This is a **timed automaton**.

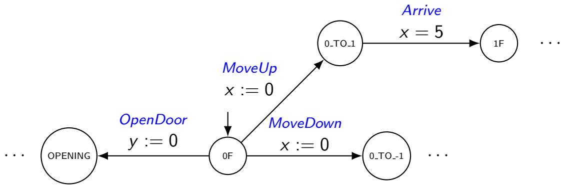
The two semantics

What we consider as a *behaviour* of the system?



The two semantics

What we consider as a *behaviour* of the system?

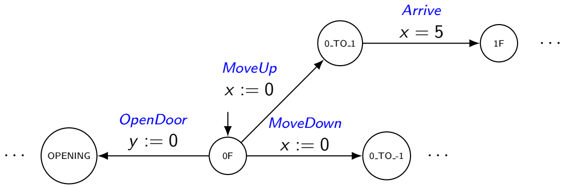


- **Pointwise** (event-based) view: **timed word**

$$(\textit{MoveUp}, 1)(\textit{Arrive}, 6)\ldots$$

The two semantics

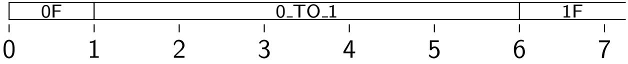
What we consider as a *behaviour* of the system?



- **Pointwise** (event-based) view: **timed word**

$$(\textit{MoveUp}, 1)(\textit{Arrive}, 6)\ldots$$

- **Continuous** (state-based) view: **signal** from $\mathbb{R}_{\geqslant 0}$ to states



Metric Temporal Logic (MTL)

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathsf{U}_I \, \varphi$$

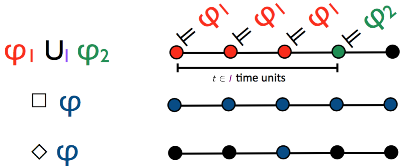
with $a \in \Sigma$, $I \subseteq [0, \infty)$ with bounds in $\mathbb{N} \cup \{+\infty\}$.

Metric Temporal Logic (MTL)

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathsf{U}_I \, \varphi$$

with $a \in \Sigma$, $I \subseteq [0, \infty)$ with bounds in $\mathbb{N} \cup \{+\infty\}$.

In the pointwise semantics:

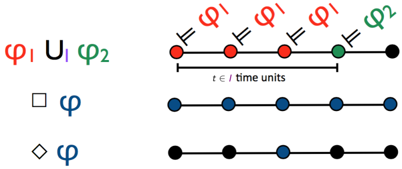


Metric Temporal Logic (MTL)

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \varphi$$

with $a \in \Sigma$, $I \subseteq [0, \infty)$ with bounds in $\mathbb{N} \cup \{+\infty\}$.

In the pointwise semantics:



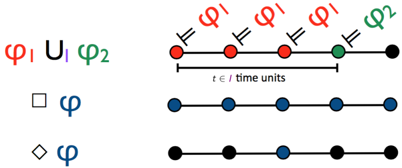
'There is a *MoveUp* followed by an *Arrive* after 5 t.u.'

Metric Temporal Logic (MTL)

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathbf{U}_I \, \varphi$$

with $a \in \Sigma$, $I \subseteq [0, \infty)$ with bounds in $\mathbb{N} \cup \{+\infty\}$.

In the pointwise semantics:



'There is a *MoveUp* followed by an *Arrive* after 5 t.u.'

$$\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]} \mathit{Arrive})$$

What do we want to do?

1. Satisfiability of an MTL formula
   - check whether a specification is consistent

What do we want to do?

1. Satisfiability of an MTL formula
   - check whether a specification is consistent
2. Model-check a timed model against an MTL formula
   - verification of the system

What do we want to do?

1. Satisfiability of an MTL formula
   - check whether a specification is consistent
2. Model-check a timed model against an MTL formula
   - verification of the system
3. Synthesise a valid system from an MTL specification, under certain restrictions on the environment
   - reactive synthesis task

Part 1: satisfiability and model-checking

Based on a joint work with Thomas Brihaye (UMONS),
Gilles Geeraerts (ULB), and Hsi-Ming Ho (UMONS)



Submitted at CAV 2017 @ Heidelberg

State of the art: fundamental difficulties

State of the art: fundamental difficulties

Theorem: [Alur and Dill, 1994]

Timed automata are not closed under complementation.

State of the art: fundamental difficulties

Theorem: [Alur and Dill, 1994]

Timed automata are not closed under complementation.

Theorem: [Alur and Dill, 1994]

Universality and language inclusion are undecidable for timed automata.

State of the art: fundamental difficulties

Theorem: [Alur and Dill, 1994]

Timed automata are not closed under complementation.

Theorem: [Alur and Dill, 1994]

Universality and language inclusion are undecidable for timed automata.

Theorem: [Alur and Dill, 1994, Ouaknine and Worrell, 2006]

Satisfiability and model checking for MTL are undecidable (over infinite words).

State of the art: fundamental difficulties

Theorem: [Alur and Dill, 1994]

Timed automata are not closed under complementation.

Theorem: [Alur and Dill, 1994]

Universality and language inclusion are undecidable for timed automata.

Theorem: [Alur and Dill, 1994, Ouaknine and Worrell, 2006]

Satisfiability and model checking for MTL are undecidable (over infinite words).

Can we find a *fully decidable* subclass?

A brief history of time (in automata and logics)

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]} \mathit{Arrive})$

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]} \mathit{Arrive})$
- TA [Alur and Dill, 1994]

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]} \mathit{Arrive})$
- TA [Alur and Dill, 1994]
- TCTL [Alur, Courcoubetis, and Dill, 1990]
  - $A\Diamond(\mathit{MoveUp} \wedge A\Diamond_{[5,5]} \mathit{Arrive})$

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]}\mathit{Arrive})$
- TA [Alur and Dill, 1994]
- TCTL [Alur, Courcoubetis, and Dill, 1990]
  - $A\Diamond(\mathit{MoveUp} \wedge A\Diamond_{[5,5]}\mathit{Arrive})$
- MITL [Alur, Feder, and Henzinger, 1996]
  - Same as MTL except that the bounding interval ***I*** must be **non-singular**

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]}\mathit{Arrive})$
- TA [Alur and Dill, 1994]
- TCTL [Alur, Courcoubetis, and Dill, 1990]
  - $A\Diamond(\mathit{MoveUp} \wedge A\Diamond_{[5,5]}\mathit{Arrive})$
- MITL [Alur, Feder, and Henzinger, 1996]
  - Same as MTL except that the bounding interval I must be **non-singular**
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[4,6]}\mathit{Arrive})$

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]} \mathit{Arrive})$
- TA [Alur and Dill, 1994]
- TCTL [Alur, Courcoubetis, and Dill, 1990]
  - $A\Diamond(\mathit{MoveUp} \wedge A\Diamond_{[5,5]} \mathit{Arrive})$
- MITL [Alur, Feder, and Henzinger, 1996]
  - Same as MTL except that the bounding interval I must be **non-singular**
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[4,6]} \mathit{Arrive})$
- ECA [Alur, Fix, and Henzinger, 1999, Henzinger, Raskin, and Schobbens, 1998]

A brief history of time (in automata and logics)

- TPTL [Alur and Henzinger, 1989]
  - $\Diamond x.\big(\mathit{MoveUp} \wedge \Diamond y.(\mathit{Arrive} \wedge y = x + 5)\big)$
- MTL [Koymans, 1990]
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[5,5]} \mathit{Arrive})$
- TA [Alur and Dill, 1994]
- TCTL [Alur, Courcoubetis, and Dill, 1990]
  - $A\Diamond(\mathit{MoveUp} \wedge A\Diamond_{[5,5]} \mathit{Arrive})$
- MITL [Alur, Feder, and Henzinger, 1996]
  - Same as MTL except that the bounding interval I must be **non-singular**
  - $\Diamond(\mathit{MoveUp} \wedge \Diamond_{[4,6]} \mathit{Arrive})$
- ECA [Alur, Fix, and Henzinger, 1999, Henzinger, Raskin, and Schobbens, 1998]
- ECL [Raskin and Schobbens, 1999]
  - $\Diamond(\mathit{MoveUp} \wedge \triangleright_{[5,5]} \mathit{Arrive})$

Metric Interval Temporal Logic (MITL)

In real world there is no infinite precision!

Metric Interval Temporal Logic (MITL)

In real world there is no infinite precision!

Theorem: [Alur et al., 1996]

MITL can be translated into timed automata.

Metric Interval Temporal Logic (MITL)

In real world there is no infinite precision!

Theorem: [Alur et al., 1996]

MITL can be translated into timed automata.

Theorem: [Alur et al., 1996]

Satisfiability and model checking for MITL are EXPSPACE-complete.

Metric Interval Temporal Logic (MITL)

In real world there is no infinite precision!

Theorem: [Alur et al., 1996]

MITL can be translated into timed automata.

Theorem: [Alur et al., 1996]

Satisfiability and model checking for MITL are EXPSPACE-complete.

Too expensive?

Metric Interval Temporal Logic (MITL)

In real world there is no infinite precision!

Theorem: [Alur et al., 1996]

MITL can be translated into timed automata.

Theorem: [Alur et al., 1996]

Satisfiability and model checking for MITL are EXPSPACE-complete.

Too expensive?

Theorem: [Raskin and Schobbens, 1999]

Satisfiability and model checking for ECL are PSPACE-complete.

Metric Interval Temporal Logic (MITL)

In real world there is no infinite precision!

Theorem: [Alur et al., 1996]

MITL can be translated into timed automata.

Theorem: [Alur et al., 1996]

Satisfiability and model checking for MITL are EXPSPACE-complete.

Too expensive?

Theorem: [Raskin and Schobbens, 1999]

Satisfiability and model checking for ECL are PSPACE-complete.

Theorem: [Wilke, 1994, Henzinger et al., 1998]

ECL with projection (i.e. outermost second-order quantification) is equally expressive as timed automata.

UPPAAL

Uppaal

- Started in 1995 (at Uppsala + Aalborg)

UPPAAL

- Started in 1995 (at Uppsala + Aalborg)
- Model checking networks of timed automata against a fragment of TCTL
  - a pretty restricted fragment, but at least reachability is supported

Uppaal

- Started in 1995 (at Uppsala + Aalborg)
- Model checking networks of timed automata against a fragment of TCTL
  - a pretty restricted fragment, but at least reachability is supported
- The *de facto* standard tool for timed automata

UPPAAL in a nutshell
KG Larsen, P Pettersson, W Yi - International journal on software tools for ..., 1997 - Springer
Abstract. This paper presents the overal structure, the design criteria, and the main features of the tool box **Uppaal**. It gives a detailed user guide which describes how to use the various tools of **Uppaal** version 2.02 to construct abstract models of a real-time system, to simulate
Cited by 2153 Related articles All 18 versions Cite Save

A tutorial on **uppaal**
G Behrmann, A David, KG Larsen - Formal methods for the design of real- ..., 2004 - Springer
Abstract This is a tutorial paper on the tool **Uppaal**. Its goal is to be a short introduction on the flavor of timed automata implemented in the tool, to present its interface, and to explain how to use the tool. The contribution of the paper is to provide reference examples and
Cited by 1557 Related articles All 69 versions Cite Save

Tool support for MITL

Tool support for MITL

Practically non-existent. Why?

Tool support for MITL

Practically non-existent. Why?

- Standard construction [Alur, Feder, and Henzinger, 1996]: *monolithic* and notoriously complicated

Tool support for MITL

Practically non-existent. Why?

- Standard construction [Alur, Feder, and Henzinger, 1996]: *monolithic* and notoriously complicated
- Simplified *compositional* constructions (notably [Maler, Nickovic, and Pnueli, 2005]): based on a less common model (timed signal transducers)

Tool support for MITL

Practically non-existent. Why?

- Standard construction [Alur, Feder, and Henzinger, 1996]: *monolithic* and notoriously complicated
- Simplified *compositional* constructions (notably [Maler, Nickovic, and Pnueli, 2005]): based on a less common model (timed signal transducers)
- Usage of continuous semantics, different from existing tools (such as UPPAAL) built upon pointwise semantics

Tool support for MITL

Practically non-existent. Why?

- Standard construction [Alur, Feder, and Henzinger, 1996]: *monolithic* and notoriously complicated
- Simplified *compositional* constructions (notably [Maler, Nickovic, and Pnueli, 2005]): based on a less common model (timed signal transducers)
- Usage of continuous semantics, different from existing tools (such as UPPAAL) built upon pointwise semantics

Construction for ECL ($\equiv$ $\mathsf{MITL}_{0,\infty}$) much simpler and adaptable to the pointwise semantics [Henzinger, 1998].
Still, most LTL-to-BA constructions are monolithic : difficult to modify them to incorporate time.

Tool support for MITL

Practically non-existent. Why?

- Standard construction [Alur, Feder, and Henzinger, 1996]: *monolithic* and notoriously complicated
- Simplified *compositional* constructions (notably [Maler, Nickovic, and Pnueli, 2005]): based on a less common model (timed signal transducers)
- Usage of continuous semantics, different from existing tools (such as UPPAAL) built upon pointwise semantics

Construction for ECL ($\equiv \mathsf{MITL}_{0,\infty}$) much simpler and adaptable to the pointwise semantics [Henzinger, 1998].
Still, most LTL-to-BA constructions are monolithic : difficult to modify them to incorporate time.

Other direction of research: usage of SMT solvers [Bersani, Rossi, and San Pietro, 2015, Kindermann, Junttila, and Niemelä, 2013, Woźna-Szcześniak, Szcześniak, M. Zbrzezny, and Zbrzezny, 2014]

A closer look

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

Theorem: [Brihaye, Estiévenart, and Geeraerts, 2014]

MITL can be translated into *pointwise* timed automata.

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

Theorem: [Brihaye, Estiévenart, and Geeraerts, 2014]

MITL can be translated into *pointwise* timed automata.

This work:

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

Theorem: [Brihaye, Estiévenart, and Geeraerts, 2014]

MITL can be translated into *pointwise* timed automata.

This work:

- Compositional

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

Theorem: [Brihaye, Estiévenart, and Geeraerts, 2014]

MITL can be translated into *pointwise* timed automata.

This work:

- Compositional
- Less states (subsumes [Gastin and Oddoux, 2001])

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

Theorem: [Brihaye, Estiévenart, and Geeraerts, 2014]

MITL can be translated into *pointwise* timed automata.

This work:

- Compositional
- Less states (subsumes [Gastin and Oddoux, 2001])
- Less clocks

A closer look

Theorem: [Alur, Feder, and Henzinger, 1996]

MITL can be translated into *continuous* timed automata.

Theorem: [Brihaye, Estiévenart, and Geeraerts, 2014]

MITL can be translated into *pointwise* timed automata.

This work:

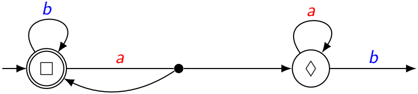
- Compositional
- Less states (subsumes [Gastin and Oddoux, 2001])
- Less clocks
- Works well with UPPAAL!

From LTL to alternating automata [Vardi, 1998]

$$\Box(a \Rightarrow \Diamond b)$$

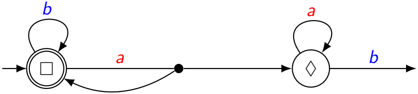
From LTL to alternating automata [Vardi, 1998]

$$\Box(a \Rightarrow \Diamond b)$$

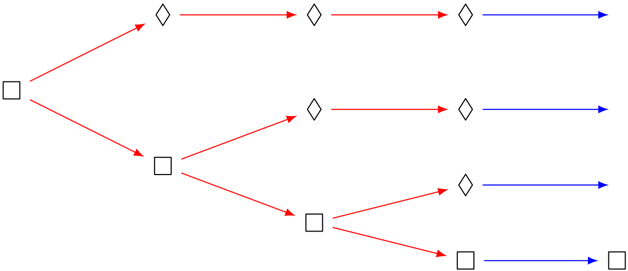


From LTL to alternating automata [Vardi, 1998]

$$\Box(a \Rightarrow \Diamond b)$$



A run on *aaab*:



From alternating automata to non-deterministic automata

Theorem: [Miyano and Hayashi, 1984]

An alternating Büchi automaton with n locations can be translated into a non-deterministic Büchi automaton with 3^n locations.

From alternating automata to non-deterministic automata

Theorem: [Miyano and Hayashi, 1984]

An alternating Büchi automaton with n locations can be translated into a non-deterministic Büchi automaton with 3^n locations.

Theorem: [Gastin and Oddoux, 2001]

An LTL formula of size n can be translated into a non-deterministic Büchi automaton with $n \times 2^n$ locations.

From alternating automata to non-deterministic automata

Theorem: [Miyano and Hayashi, 1984]

An alternating Büchi automaton with n locations can be translated into a non-deterministic Büchi automaton with 3^n locations.

Theorem: [Gastin and Oddoux, 2001]

An LTL formula of size n can be translated into a non-deterministic Büchi automaton with $n \times 2^n$ locations.

The tool $\mathrm{LTL2BA}$ is still in wide use today.

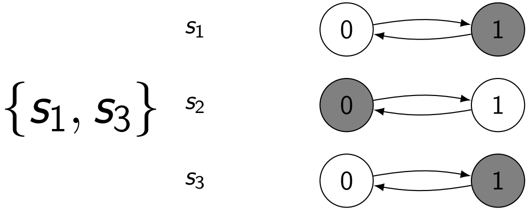
Compositional Gastin-Oddoux

Idea: One component automaton for each location of alternating automaton.

Compositional Gastin-Oddoux

Idea: One component automaton for each location of alternating automaton.

A set of locations is represented by a location of the product of components, e.g.,



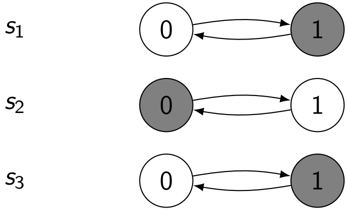
Component in state 1 $\iff$ corresponding location in the configuration of the alternating automaton

Compositional Gastin-Oddoux

Idea: One component automaton for each location of alternating automaton.

A set of locations is represented by a location of the product of components, e.g.,

$$\{s_1, s_3\}$$



Component in state 1 $\iff$ corresponding location in the configuration of the alternating automaton

How to synchronise these components?

Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:

Compositional Gastin-Oddoux

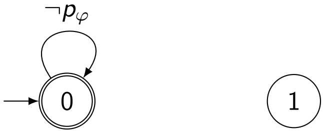
For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:

Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

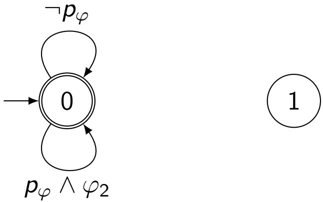
E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

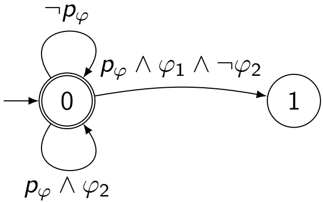
E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

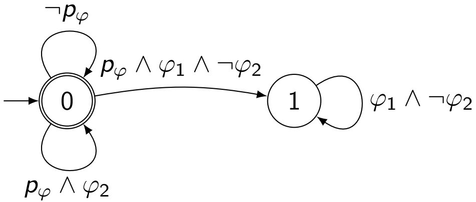
E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

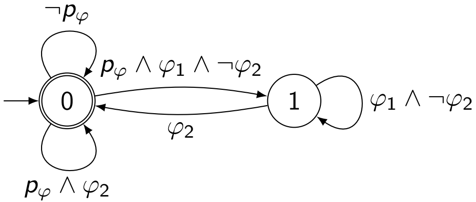
E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

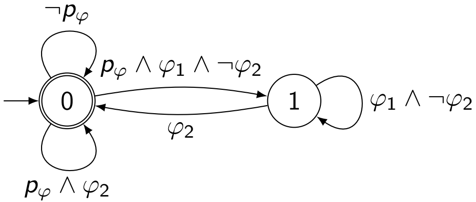
E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



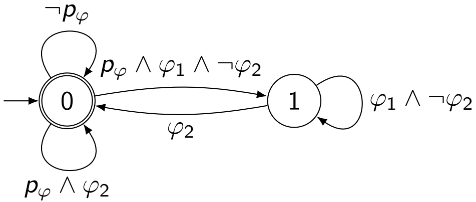
Proposition:

If $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$ accepts a (timed) word ρ then $\rho \models \Box(p_\varphi \Rightarrow \varphi_1 \,\mathcal{U}\, \varphi_2)$.

Compositional Gastin-Oddoux

For each component $\mathcal{C}_\varphi$, we add a fresh proposition p_φ (a **trigger**).

E.g. $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$:



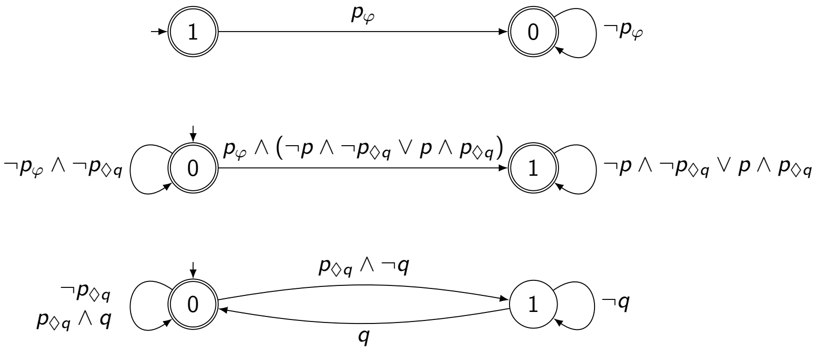
Proposition:

If $\mathcal{C}_{\varphi_1 \mathcal{U} \varphi_2}$ accepts a (timed) word ρ then $\rho \models \Box(p_\varphi \Rightarrow \varphi_1 \,\mathcal{U}\, \varphi_2)$.

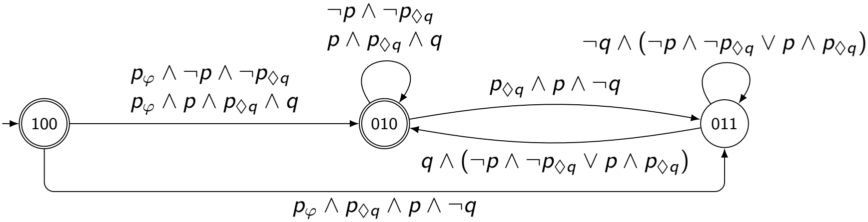
Proposition:

For each LTL formula φ over AP, we can construct a Büchi automaton $\mathcal{A}_\varphi = \mathcal{C}_{\psi_1} \times \cdots \times \mathcal{C}_{\psi_n}$ over AP $\cup$ AP' such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathrm{proj}_{\mathrm{AP}}(\mathcal{A}_\varphi))$.

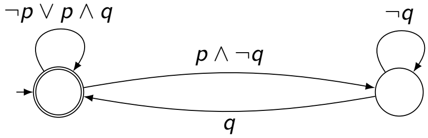
Compositional Gastin-Oddoux: full example
$\varphi = \Box(p \Rightarrow \Diamond q) \equiv \bot\,\mathcal{R}\,(\neg p \vee \top\,\mathcal{U}\,q)$



Compositional Gastin-Oddoux: full example
$\varphi = \Box(p \Rightarrow \Diamond q) \equiv \bot\,\mathcal{R}\,(\neg p \vee \top\,\mathcal{U}\,q)$



Compositional Gastin-Oddoux: full example
$\varphi = \Box(p \Rightarrow \Diamond q) \equiv \bot \, \mathcal{R} \, (\neg p \vee \top \, \mathcal{U} \, q)$

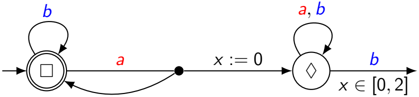


From MITL to one-clock alternating timed automata (OCATA) [Ouaknine and Worrell, 2005]

$$\Box(a \Rightarrow \Diamond_{[0,2]} b)$$

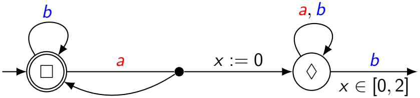
From MITL to one-clock alternating timed automata (OCATA) [Ouaknine and Worrell, 2005]

$$\square(a \Rightarrow \lozenge_{[0,2]} b)$$

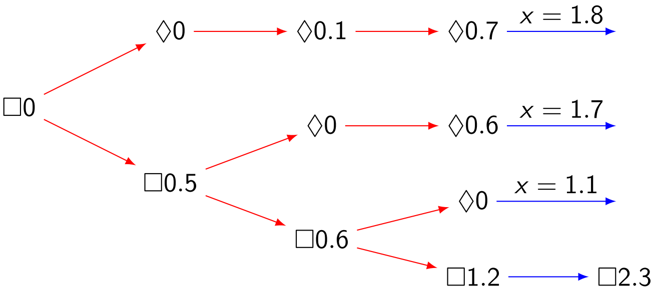


From MITL to one-clock alternating timed automata (OCATA) [Ouaknine and Worrell, 2005]

$$\Box(a \Rightarrow \Diamond_{[0,2]} b)$$

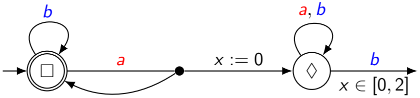


A run on $(a, 0.5)(a, 0.6)(a, 1.2)(b, 2.3)$:

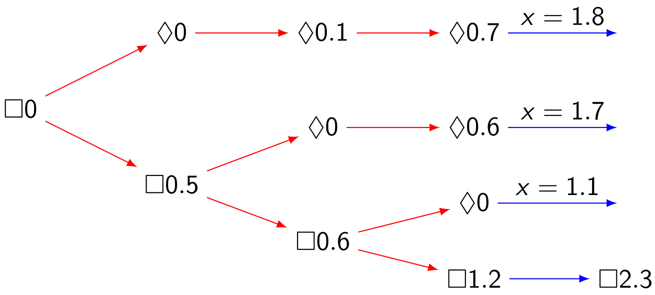


From MITL to one-clock alternating timed automata (OCATA) [Ouaknine and Worrell, 2005]

$$\Box(a \Rightarrow \Diamond_{[0,2]} b)$$



A run on $(a, 0.5)(a, 0.6)(a, 1.2)(b, 2.3)$:



In this case we can simply keep the 'oldest' $\Diamond$.

$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[0,2]} \varphi_2}$:

$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[0,2]} \varphi_2}$:

$\mathsf{MITL}_{0,\infty}$

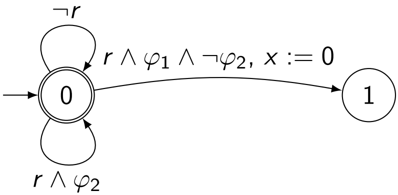
The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[0,2]} \varphi_2}$:

$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

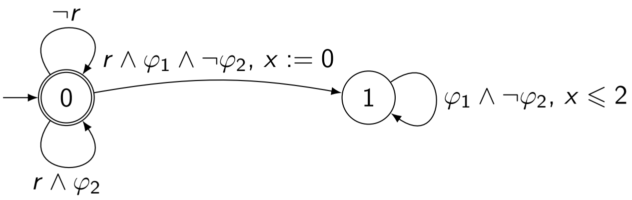
E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[0,2]} \varphi_2}$:



MITL$_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

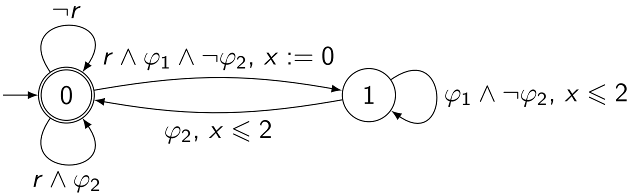
E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[0,2]} \varphi_2}$:



$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

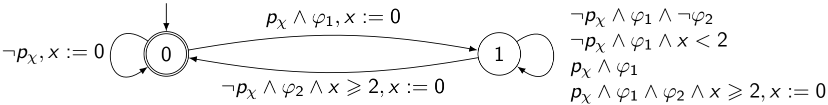
E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[0,2]} \varphi_2}$:



MITL$_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

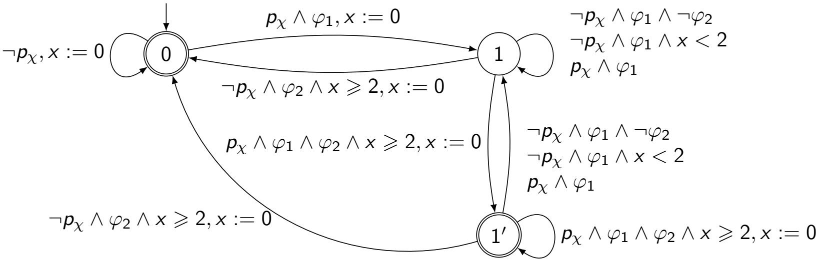
E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[2,\infty]} \varphi_2}$:



$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

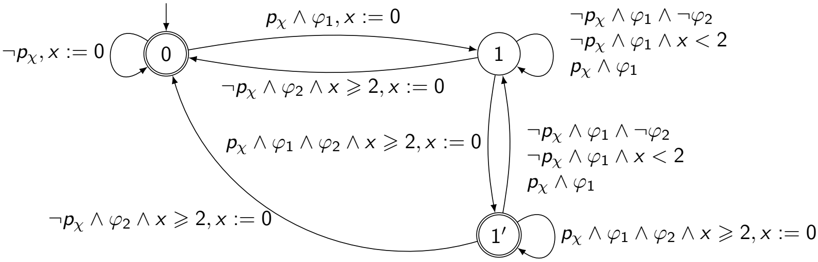
E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[2,\infty]} \varphi_2}$:



$\mathsf{MITL}_{0,\infty}$

The MITL fragment in which all intervals are of the form $< c$, $\leqslant c$, $> c$ or $\geqslant c$.

E.g., $\mathcal{C}_{\varphi_1 \mathcal{U}_{[2,\infty]} \varphi_2}$:



Proposition:

For each $\mathsf{MITL}_{0,\infty}$ formula φ with n timed subformulas, we can construct a projection-equivalent timed automaton $\mathcal{A}_\varphi$ that uses n clocks.

Full MITL: inspired by interval semantics for OCATA

New semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]:

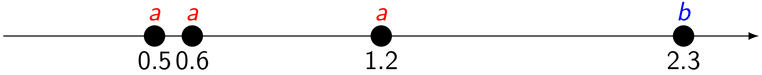
- allows one to bound the number of clock copies
- sufficiently expressive for MITL

Full MITL: inspired by interval semantics for OCATA

New semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]:

- allows one to bound the number of clock copies
- sufficiently expressive for MITL

$$\varphi = \Box(a \Rightarrow \Diamond[[1,2]]b)$$

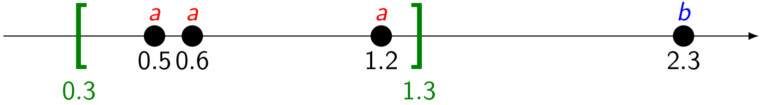


Full MITL: inspired by interval semantics for OCATA

New semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]:

- allows one to bound the number of clock copies
- sufficiently expressive for MITL

$$\varphi = \Box(a \Rightarrow \Diamond[[1,2]]b)$$

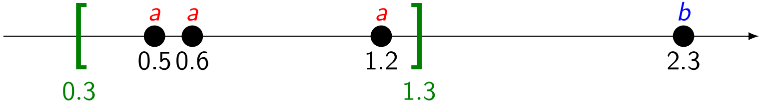


Full MITL: inspired by interval semantics for OCATA

New semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]:

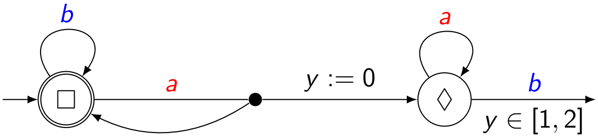
- allows one to bound the number of clock copies
- sufficiently expressive for MITL

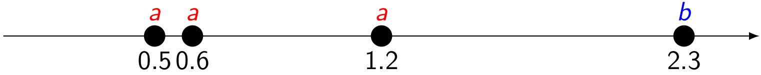
$$\varphi = \Box(a \Rightarrow \Diamond[[1,2]]b)$$



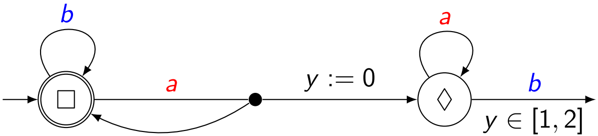
To check that this timed word satisfies φ, we **do not need to remember** the exact timestamp of each a

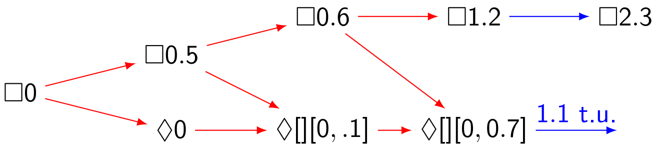
Example run with the interval semantics

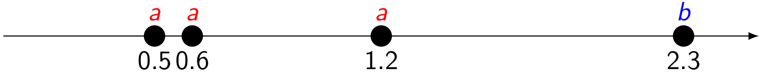




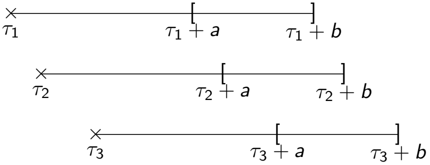
Example run with the interval semantics



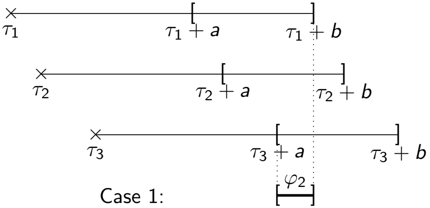




$\varphi = \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics

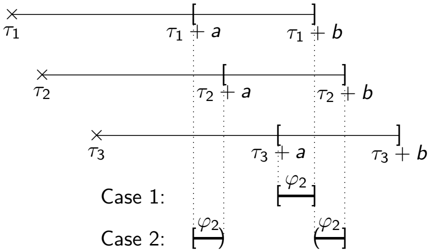


$\varphi = \varphi_1 \,\mathcal{U}_{[a,b]}\, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



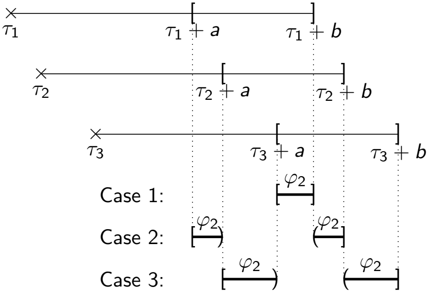
Case 1: $\{1, 2, 3\}$

$\varphi = \varphi_1 \,\mathcal{U}_{[a,b]}\, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



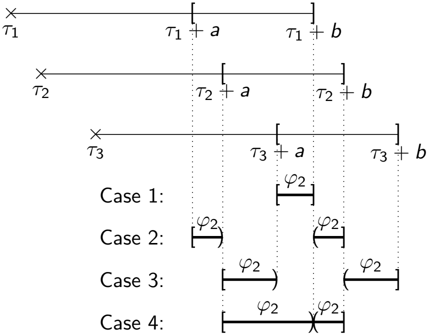
Case 1: $\{1, 2, 3\}$
Case 2: $\{1\}$, $\{2, 3\}$

$\varphi = \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



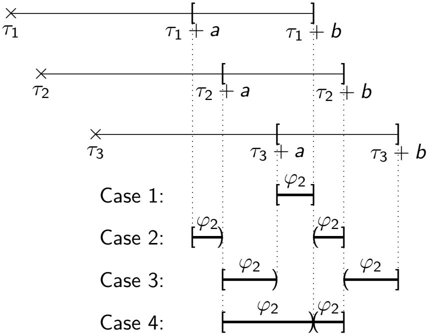
Case 1: $\{1, 2, 3\}$
Case 2: $\{1\}$, $\{2, 3\}$
Case 3: $\{1, 2\}$, $\{3\}$

$\varphi = \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



Case 1: $\{1, 2, 3\}$
Case 2: $\{1\}$, $\{2, 3\}$
Case 3: $\{1, 2\}$, $\{3\}$
Case 4: $\{1, 2\}$, $\{2, 3\}$ or $\{1, 2, 3\}$, $\{2, 3\}$

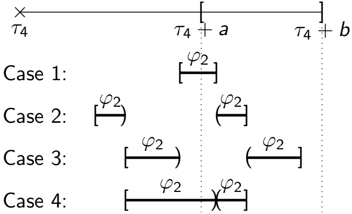
$\varphi = \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



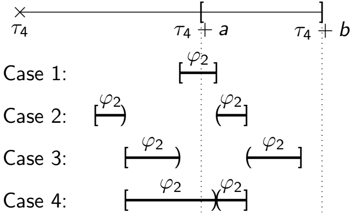
Case 1: $\{1, 2, 3\}$
Case 2: $\{1\}$, $\{2, 3\}$
Case 3: $\{1, 2\}$, $\{3\}$
Case 4: $\{1, 2\}$, $\{2, 3\}$ or $\{1, 2, 3\}$, $\{2, 3\}$

In each case we only need to keep track of two clock values.

$\varphi = \varphi_1 \,\mathcal{U}_{[a,b]}\, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics

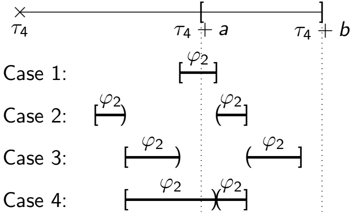


$\varphi = \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



Case 1: Another branching into Case 1, Case 3 and Case 4
Case 2: Done
Case 3: Done
Case 4: Done

$\varphi = \varphi_1 \, \mathcal{U}_{[a,b]} \, \varphi_2$, with $0 < a < b < +\infty$: improvement on the interval semantics



Case 1: Another branching into Case 1, Case 3 and Case 4
Case 2: Done
Case 3: Done
Case 4: Done

Proposition:

For each MITL formula $\varphi = \varphi_1 \, \mathcal{U}_I \, \varphi_2$, $\mathcal{C}_\varphi$ uses $2 \cdot \lceil \frac{\sup I}{|I|} \rceil + 2$ clocks.

Up to half the number of clocks obtained in [Brihaye, Estiévenart, and Geeraerts, 2014]

Experiments

We have implemented the translation in the tool MightyL.

Experiments

We have implemented the translation in the tool MightyL.

$F(k,I) = \bigwedge_{i=1}^{k} \Diamond_I p_i$

$G(k,I) = \bigwedge_{i=1}^{k} \Box_I p_i$

$U(k,I) = (\cdots (p_1 \, \mathcal{U}_I \, p_2) \, \mathcal{U}_I \cdots) \, \mathcal{U}_I \, p_k$

$R(k,I) = (\cdots (p_1 \, \mathcal{R}_I \, p_2) \, \mathcal{R}_I \cdots) \, \mathcal{R}_I \, p_k$

$\theta(k,I) = \neg((\bigwedge_{i=1}^{k} \Box\Diamond p_i) \Rightarrow \Box(q \Rightarrow \Diamond_I r))$

$\mu(k) = \bigwedge_{i=1}^{k} \Diamond_{[3(i-1),3i]} t_i \wedge \Box\neg p$

Formula	MightyL	LTSmin	Uppaal
$F(5,[0,\infty))$	9ms	3.48s/2.18s/0.12s	0.75s
$F(5,[0,2])$	7ms	3.76s/2.23s/0.15s	0.84s
$F(5,[2,\infty))$	6ms	3.76s/2.26s/0.91s	1.64s
$F(3,[1,2])$	70ms	6m5.15s/38.01s/0.22s	9.00s
$F(5,[1,2])$	70ms	$>$15m	2m6s
$G(5,[0,\infty))$	10ms	3.83s/2.43s/0.05s	0.75s
$G(5,[0,2])$	10ms	4.01s/2.51s/0.10s	0.82s
$G(5,[2,\infty))$	9ms	4.06s/2.47s/0.04s	0.85s
$G(5,[1,2])$	15ms	7.81s/2.99s/0.09s	1.12s
$\mu(1)$	13ms	-	0.39s
$\mu(2)$	21ms	-	2.33s
$\mu(3)$	76ms	-	15.77s
$\mu(4)$	87ms	-	2m23s

Formula	MightyL	LTSmin	Uppaal
$U(5,[0,\infty))$	16ms	1.90s/1.44s/0.05s	0.41s
$U(5,[0,2])$	8ms	2.08s/1.54s/0.06s	0.42s
$U(5,[2,\infty))$	8ms	2.08s/1.53s/0.09s	0.52s
$U(3,[1,2])$	49ms	4m0.14s/23.54s/0.09s	4.92s
$U(5,[1,2])$	97ms	$>$15m	21.80s
$R(5,[0,\infty))$	7ms	1.86s/1.42s/0.03s	0.40s
$R(5,[0,2])$	7ms	1.97s/1.44s/0.03s	0.40s
$R(5,[2,\infty))$	7ms	1.92s/1.42s/0.03s	0.42s
$R(5,[1,2])$	10ms	5.37s/2.16s/0.04s	0.62s
$\theta(1,[100,1000])$	9ms	1.88s/1.74s/0.04s	0.25s
$\theta(2,[100,1000])$	13ms	5.04s/3.17s/0.19s	0.86s
$\theta(3,[100,1000])$	14ms	36.57s/16.27s/3.20s	21.84s
$\theta(4,[100,1000])$	15ms	5m30s/4m18s/2m16s	18m39s

Experiments

We have implemented the translation in the tool MightyL.

$$F(k,I) = \bigwedge_{i=1}^{k} \Diamond_I p_i \qquad G(k,I) = \bigwedge_{i=1}^{k} \Box_I p_i$$

$$U(k,I) = (\cdots(p_1 \,\mathcal{U}_I\, p_2)\,\mathcal{U}_I \cdots)\,\mathcal{U}_I\, p_k \qquad R(k,I) = (\cdots(p_1 \,\mathcal{R}_I\, p_2)\,\mathcal{R}_I \cdots)\,\mathcal{R}_I\, p_k$$

$$\theta(k,I) = \neg((\bigwedge_{i=1}^{k} \Box\Diamond p_i) \Rightarrow \Box(q \Rightarrow \Diamond_I r)) \qquad \mu(k) = \bigwedge_{i=1}^{k} \Diamond_{[3(i-1),3i]} t_i \wedge \Box\neg p$$

Formula	MightyL	LTSmin	Uppaal
$F(5,[0,\infty))$	9ms	3.48s/2.18s/0.12s	0.75s
$F(5,[0,2])$	7ms	3.76s/2.23s/0.15s	0.84s
$F(5,[2,\infty))$	6ms	3.76s/2.26s/0.91s	1.64s
$F(3,[1,2])$	70ms	6m5.15s/38.01s/0.22s	9.00s
$F(5,[1,2])$	70ms	>15m	2m6s
$G(5,[0,\infty))$	10ms	3.83s/2.43s/0.05s	0.75s
$G(5,[0,2])$	10ms	4.01s/2.51s/0.10s	0.82s
$G(5,[2,\infty))$	9ms	4.06s/2.47s/0.04s	0.85s
$G(5,[1,2])$	15ms	7.81s/2.99s/0.09s	1.12s
$\mu(1)$	13ms	-	0.39s
$\mu(2)$	21ms	-	2.33s
$\mu(3)$	76ms	-	15.77s
$\mu(4)$	87ms	-	2m23s

Formula	MightyL	LTSmin	Uppaal
$U(5,[0,\infty))$	16ms	1.90s/1.44s/0.05s	0.41s
$U(5,[0,2])$	8ms	2.08s/1.54s/0.06s	0.42s
$U(5,[2,\infty))$	8ms	2.08s/1.53s/0.09s	0.52s
$U(3,[1,2])$	49ms	4m0.14s/23.54s/0.09s	4.92s
$U(5,[1,2])$	97ms	>15m	21.80s
$R(5,[0,\infty))$	7ms	1.86s/1.42s/0.03s	0.40s
$R(5,[0,2])$	7ms	1.97s/1.44s/0.03s	0.40s
$R(5,[2,\infty))$	7ms	1.92s/1.42s/0.03s	0.42s
$R(5,[1,2])$	10ms	5.37s/2.16s/0.04s	0.62s
$\theta(1,[100,1000])$	9ms	1.88s/1.74s/0.04s	0.25s
$\theta(2,[100,1000])$	13ms	5.04s/3.17s/0.19s	0.86s
$\theta(3,[100,1000])$	14ms	36.57s/16.27s/3.20s	21.84s
$\theta(4,[100,1000])$	15ms	5m30s/4m18s/2m16s	18m39s

Formula	MightyL	LTSmin	Uppaal	SMT-based approach
$\Diamond_{[0,30]}(p \Rightarrow \Box_{[0,20]} p)$ valid	7ms	0.98s	0.32s	7s
$\Box_{[0,30]}\neg p \vee \Diamond_{[0,20]} p$ valid	7ms	0.95s	0.14s	not considered
$\Diamond_{[0,30]} p \wedge \Diamond_{[0,20]} p$ redundant	13ms	1.99s	0.44s	14s
$\Box_{[0,20]}\Diamond_{[0,20]} p \wedge \Box_{[0,40]} p \wedge \Diamond_{[20,40]} \top$ redundant	22ms	1m26s	2.63s	not considered

Summary for the satisfiability/model-checking of MITL

Contributions:

- A compositional translation from MITL to timed automata
- An implementation that works with UPPAAL and the like

Summary for the satisfiability/model-checking of MITL

Contributions:

- A compositional translation from MITL to timed automata
- An implementation that works with UPPAAL and the like

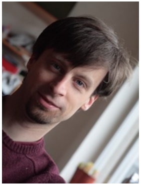
Possible future directions:

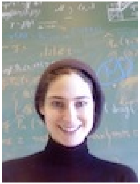
- Native support for ECL
- Past modalities, counting modalities
- Antichain-based optimisations
- ...

Part 2: reactive synthesis

Based on a joint work with Thomas Brihaye (UMONS), Morgane Estiévenart (UMONS), Gilles Geeraerts (ULB), Hsi-Ming Ho (UMONS) and Nathalie Sznajder (LIP6, UPMC)







Published at FORMATS 2016 @ Quebec City

Reactive synthesis

$\Sigma = \Sigma_C \uplus \Sigma_E$

- controllable actions owned by controller C:
  $\{$*MoveUp*, *MoveDown*, *OpenDoor*, *Opened*, $\ldots\}$
- uncontrollable actions owned by environment E:
  $\{$*0F-Up*, *0F-Down*, $\ldots$, *-1F*, *0F*, $\ldots$ *Open*, *Close*, $\ldots\}$

Reactive synthesis

$\Sigma = {\color{blue}\Sigma_C} \uplus {\color{red}\Sigma_E}$

- controllable actions owned by controller C:
  $\{$*MoveUp*, *MoveDown*, *OpenDoor*, *Opened*, $\ldots\}$
- uncontrollable actions owned by environment E:
  $\{$*0F-Up*, *0F-Down*, $\ldots$, *-1F*, *0F*, $\ldots$ *Open*, *Close*, $\ldots\}$

$+$ state (at which floor, opening, $\ldots$)

Reactive synthesis

$\Sigma = \Sigma_C \uplus \Sigma_E$

- controllable actions owned by controller C:
  $\{$*MoveUp*, *MoveDown*, *OpenDoor*, *Opened*, ...$\}$
- uncontrollable actions owned by environment E:
  $\{$*0F-Up*, *0F-Down*, ..., *-1F*, *0F*, ... *Open*, *Close*, ...$\}$

+ state (at which floor, opening, ...)

+ timing restrictions (latency, ...)

Reactive synthesis

$\Sigma = \Sigma_C \uplus \Sigma_E$

- controllable actions owned by controller C: $\{$*MoveUp*, *MoveDown*, *OpenDoor*, *Opened*, $\ldots\}$
- uncontrollable actions owned by environment E: $\{$*0F-Up*, *0F-Down*, $\ldots$, *-1F*, *0F*, $\ldots$ *Open*, *Close*, $\ldots\}$

$+$ state (at which floor, opening, $\ldots$)

$+$ timing restrictions (latency, $\ldots$)

$=$

$\boxed{\text{Plant } \mathcal{P}\text{: a DTA over } \Sigma}$

Reactive synthesis

$\Sigma = \Sigma_C \uplus \Sigma_E$

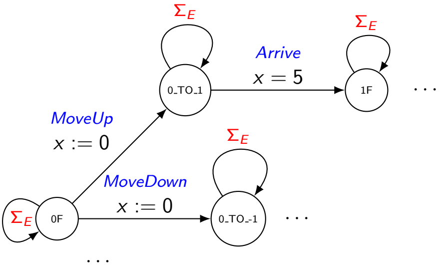
- controllable actions owned by controller C:
  $\{$*MoveUp*, *MoveDown*, *OpenDoor*, *Opened*, $\ldots\}$
- uncontrollable actions owned by environment E:
  $\{$*0F-Up*, *0F-Down*, $\ldots$, *-1F*, *0F*, $\ldots$ *Open*, *Close*, $\ldots\}$

$+$ state (at which floor, opening, $\ldots$)

$+$ timing restrictions (latency, $\ldots$)

$=$

$\boxed{\text{Plant } \mathcal{P}\text{: a DTA over } \Sigma}$



Reactive synthesis

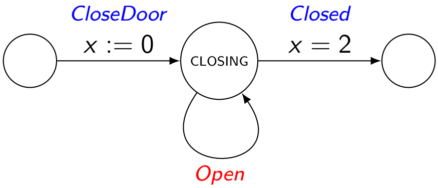
A run of $\mathcal{P}$ can be seen as a play of the *timed game* between C and E.

Reactive synthesis

A run of $\mathcal{P}$ can be seen as a play of the *timed game* between C and E.
In each round, each player proposes a pair (delay, action) **enabled in** $\mathcal{P}$:

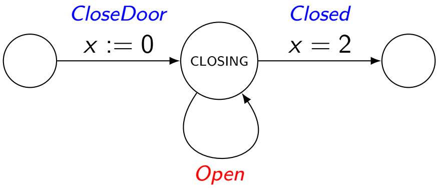
Reactive synthesis

A run of $\mathcal{P}$ can be seen as a play of the *timed game* between C and E.
In each round, each player proposes a pair (delay, action) **enabled in** $\mathcal{P}$:



Reactive synthesis

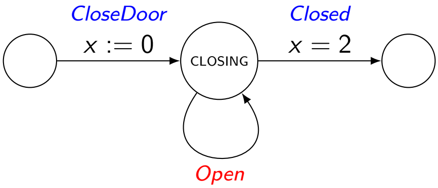
A run of $\mathcal{P}$ can be seen as a play of the *timed game* between C and E.
In each round, each player proposes a pair (delay, action) **enabled in** $\mathcal{P}$:



$(\Delta_C, \mathit{Closed})$ $(\Delta_E, \mathit{Open})$

Reactive synthesis

A run of $\mathcal{P}$ can be seen as a play of the *timed game* between C and E.
In each round, each player proposes a pair (delay, action) **enabled in** $\mathcal{P}$:



$(\Delta_C, \textit{Closed})$ $(\Delta_E, \textit{Open})$

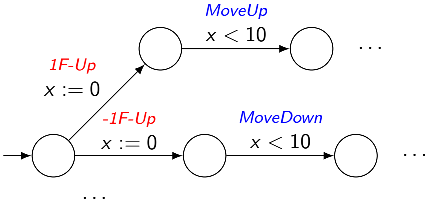
Only action(s) with the shortest delay $\min(\Delta_C, \Delta_E)$ may be played.

Reactive synthesis

‘The lift responds to any call in less than 10 t.u.’

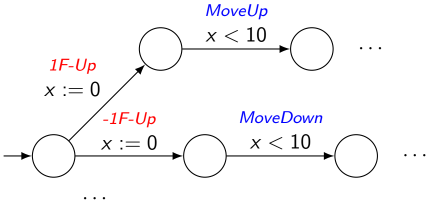
Reactive synthesis

'The lift responds to any call in less than 10 t.u.'



Reactive synthesis

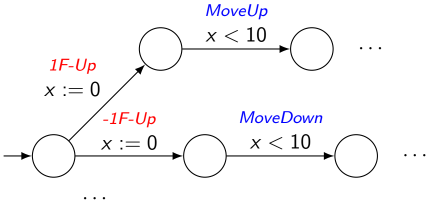
'The lift responds to any call in less than 10 t.u.'



Specification $\mathcal{L}$: a set of timed words over Σ

Reactive synthesis

'The lift responds to any call in less than 10 t.u.'



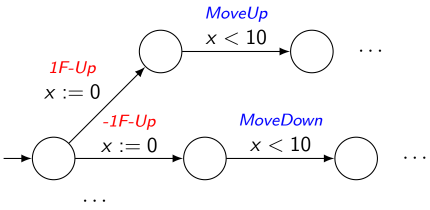
Specification $\mathcal{L}$: a set of timed words over Σ

Reactive synthesis problem (RS)

Given plant $\mathcal{P}$ and specification $\mathcal{L}$, find a strategy of Controller such that no matter what Environment does, every play satisfies the specification.

Reactive synthesis

'The lift responds to any call in less than 10 t.u.'



Specification $\mathcal{L}$: a set of timed words over Σ

Reactive synthesis problem (RS)

Given plant $\mathcal{P}$ and specification $\mathcal{L}$, find a strategy of Controller such that no matter what Environment does, every play satisfies the specification.

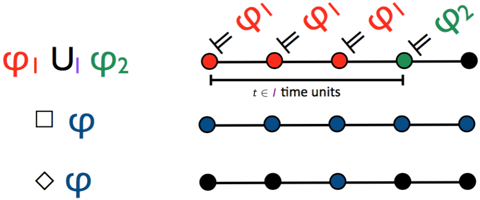
Realizability problem: the special case where all actions are always enabled, i.e., $\mathcal{P}$ is a universal DTA over Σ.

Metric Temporal Logic (MTL)

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathsf{U}_I \, \varphi$$

with $a \in \Sigma$, $I \subseteq [0, \infty)$ with bounds in $\mathbb{Q} \cup \{+\infty\}$

Models: finite (or infinite) timed words $\sigma = (a_1, t_1)(a_2, t_2) \cdots$



Theorem: [Doyen, Geeraerts, Raskin, and Reichert, 2009]

Reactive synthesis problem is undecidable for ECL (hence, MTL) specifications, even without plant.

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

As an MTL formula:

$$\Box(a \implies \neg\Diamond_{>1}\top \vee \Diamond_{=1} b)$$

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

As an MTL formula:

$$\Box(a \implies \neg\Diamond_{>1}\top \vee \Diamond_{=1} b)$$

CONTROLLABLE for RS: C acknowledges each a (in chronological order)
by playing a b 1 t.u. after

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

As an MTL formula:

$$\Box(a \implies \neg\Diamond_{>1}\top \vee \Diamond_{=1} b)$$

CONTROLLABLE for RS: C acknowledges each a (in chronological order)
by playing a b 1 t.u. after

- C requires unbounded memory: unboundedly many a's in 1 t.u.

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

As an MTL formula:

$$\Box(a \implies \neg\Diamond_{>1}\top \vee \Diamond_{=1} b)$$

CONTROLLABLE for RS: C acknowledges each a (in chronological order)
by playing a b 1 t.u. after

- C requires unbounded memory: unboundedly many a's in 1 t.u.

Theorem: [Doyen, Geeraerts, Raskin, and Reichert, 2009]

The infinite-word realizability problem is undecidable for ECL specifications.

Implementable reactive synthesis (IRS)

C = deterministic symbolic transition system $\mathcal{T}$

- set of locations; if finite $\rightarrow \mathcal{T}$ is a DTA
- finite set of clocks X
- finite set of possible clock constraints *precision* (m, K):

  $$g ::= \top \mid g \wedge g \mid x < \alpha/m \mid x \leqslant \alpha/m \mid x = \alpha/m \mid x \geqslant \alpha/m \mid x > \alpha/m$$

  with $x \in X$, $m \in \mathbb{N}$ and $0 \leqslant \alpha \leqslant K$.

Implementable reactive synthesis (IRS)

C $=$ deterministic symbolic transition system $\mathcal{T}$

- set of locations; if finite $\rightarrow \mathcal{T}$ is a DTA
- finite set of clocks X
- finite set of possible clock constraints *precision* (m, K):

$$g ::= \top \mid g \wedge g \mid x < \alpha/m \mid x \leqslant \alpha/m \mid x = \alpha/m \mid x \geqslant \alpha/m \mid x > \alpha/m$$

with $x \in X$, $m \in \mathbb{N}$ and $0 \leqslant \alpha \leqslant K$.

Definition

Implementable reactive synthesis problem (IRS): *Given $\mathcal{P}$ and $\mathcal{L}$, find such a $\mathcal{T}$ that no matter what E does, every play satisfies the specification.*

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

As an MTL formula:

$$\Box(a \implies \neg\Diamond_{>1}\top \vee \Diamond_{=1} b)$$

CONTROLLABLE for RS: C acknowledges each a (in chronological order)
by playing a b 1 t.u. after

- C requires unbounded memory: unboundedly many a's in 1 t.u.

A toy example

Let $\mathcal{P}$ be a universal plant, and the spec be

'Each a is followed exactly 1 t.u. later by a b.'

As an MTL formula:

$$\Box(a \implies \neg\Diamond_{>1}\top \vee \Diamond_{=1} b)$$

CONTROLLABLE for RS: C acknowledges each a (in chronological order)
by playing a b 1 t.u. after

- C requires unbounded memory: unboundedly many a's in 1 t.u.

NOT CONTROLLABLE for IRS

- each $\mathcal{T}$ has a bounded set of clocks

Reactive synthesis for MTL

Reactive synthesis	Undec.	[Doyen, Geeraerts, Raskin, and Reichert, 2009]

Reactive synthesis for MTL

Reactive synthesis Undec. [Doyen, Geeraerts, Raskin, and Reichert, 2009]

Controller = timed automaton

Implementable reactive synthesis Undec. [Bouyer, Bozzelli, and Chevalier, 2006]

Recovering decidability...

Clock constraints in $\mathcal{T}$:

$$g ::= \top \mid g \wedge g \mid x < \alpha/m \mid x \leqslant \alpha/m \mid x = \alpha/m \mid x \geqslant \alpha/m \mid x > \alpha/m$$

with $x \in X$, $m \in \mathbb{N}$ and $0 \leqslant \alpha \leqslant K$.

Recovering decidability...

Clock constraints in $\mathcal{T}$:

$$g ::= \top \mid g \wedge g \mid x < \alpha/m \mid x \leqslant \alpha/m \mid x = \alpha/m \mid x \geqslant \alpha/m \mid x > \alpha/m$$

with $x \in X$, $m \in \mathbb{N}$ and $0 \leqslant \alpha \leqslant K$.

Fix X and $(m, K) \Longrightarrow$ the alphabet of $\mathcal{T}$ is given!

Recovering decidability...

Clock constraints in $\mathcal{T}$:

$$g ::= \top \mid g \wedge g \mid x < \alpha/m \mid x \leqslant \alpha/m \mid x = \alpha/m \mid x \geqslant \alpha/m \mid x > \alpha/m$$

with $x \in X$, $m \in \mathbb{N}$ and $0 \leqslant \alpha \leqslant K$.

Fix X and $(m, K) \implies$ the alphabet of $\mathcal{T}$ is given!

Definition

Bounded-resources reactive synthesis problem (BResRS): *Given $\mathcal{P}$, $\mathcal{L}$, and* **a set of clocks** X **and precision** (m, K), *find such a resource-bounded $\mathcal{T}$ that no matter what E does, every play satisfies the specification.*

Reactive synthesis for MTL

Reactive synthesis	Undec. [Doyen, Geeraerts, Raskin, and Reichert, 2009]

$\downarrow$ Controller = timed automaton

Implementable reactive synthesis	Undec. [Bouyer, Bozzelli, and Chevalier, 2006]

Reactive synthesis for MTL

Reactive synthesis Undec. [Doyen, Geeraerts, Raskin, and Reichert, 2009]

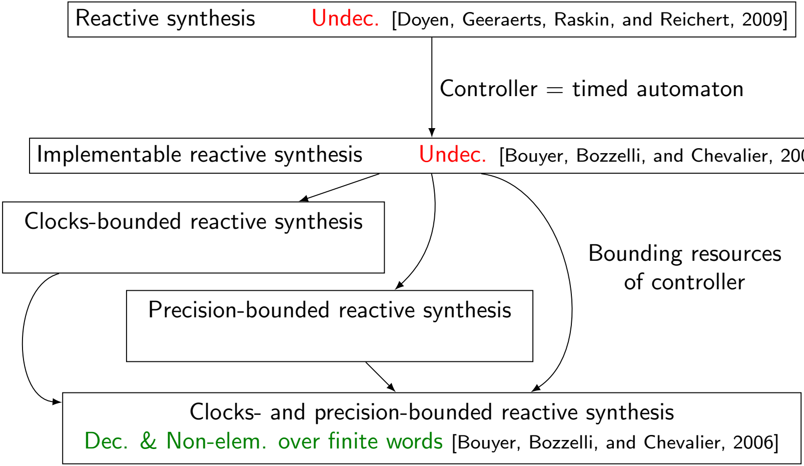
Controller = timed automaton

Implementable reactive synthesis Undec. [Bouyer, Bozzelli, and Chevalier, 2006]

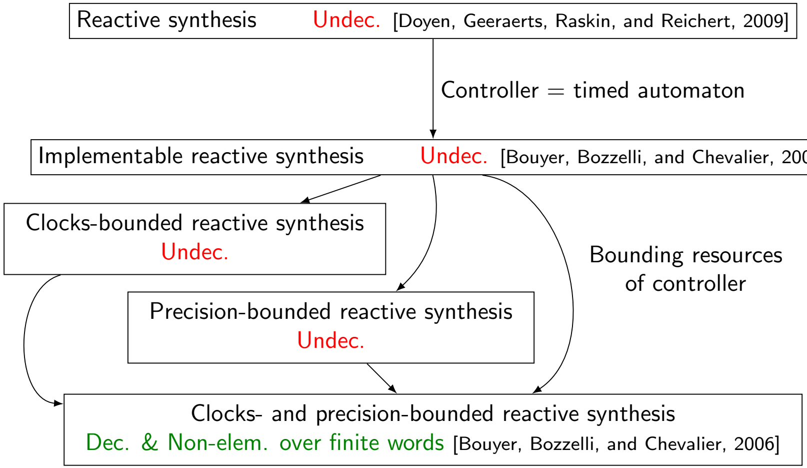
Bounding resources of controller

Clocks- and precision-bounded reactive synthesis
Dec. & Non-elem. over finite words [Bouyer, Bozzelli, and Chevalier, 2006]

Reactive synthesis for MTL



Reactive synthesis for MTL



Regaining hope? Less expressive specifications

Undecidability proofs heavily use 'punctuality' of MTL:
$\mathit{request} \to \Diamond_{=1} \mathit{grant}$

$$\mathit{request} \to \Diamond_{[1,2]} \mathit{grant}$$

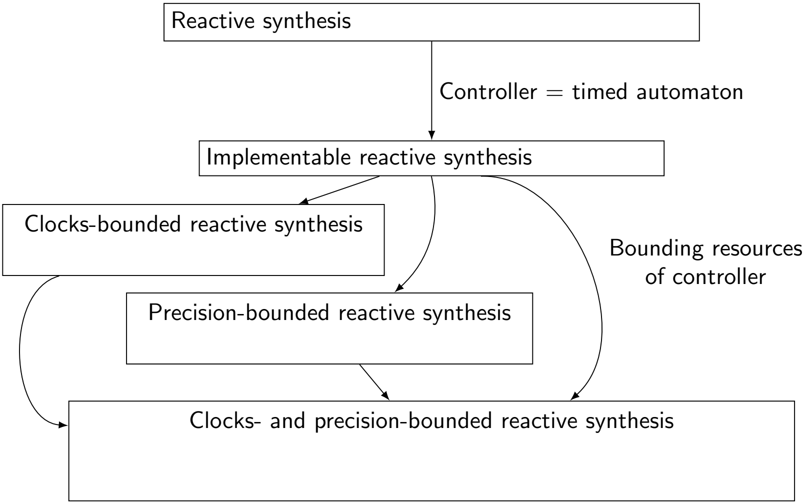
$$\mathit{request} \to \Diamond_{\leqslant 3} \mathit{grant}$$

MITL = non-punctual fragment of MTL:

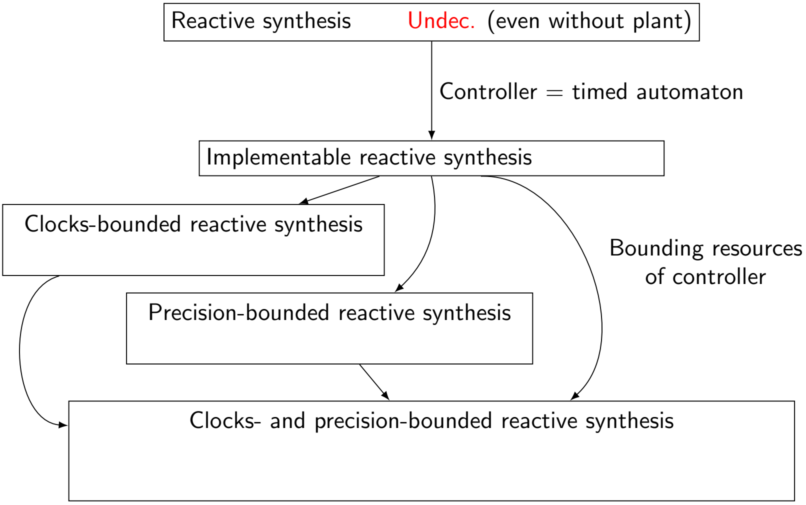
$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathsf{U}_I \, \varphi$$

with $a \in \Sigma$, $I \subseteq [0, \infty)$ is a **non-singular** with bounds in $\mathbb{Q} \cup \{+\infty\}$

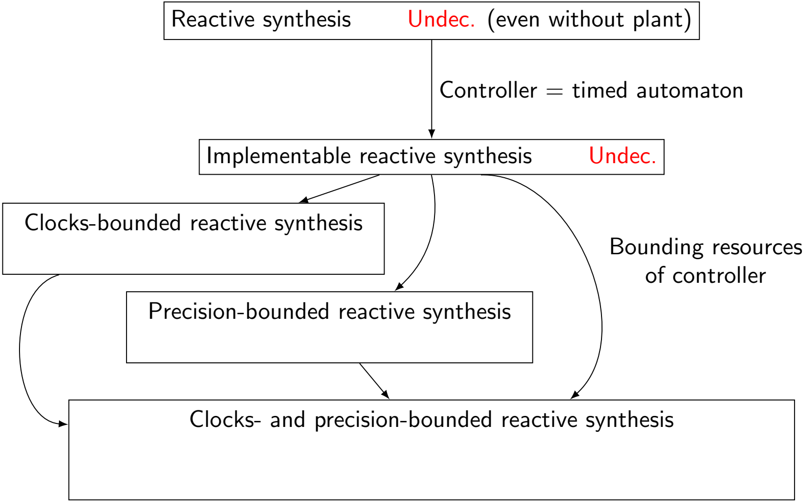
Contribution: reactive synthesis for MITL



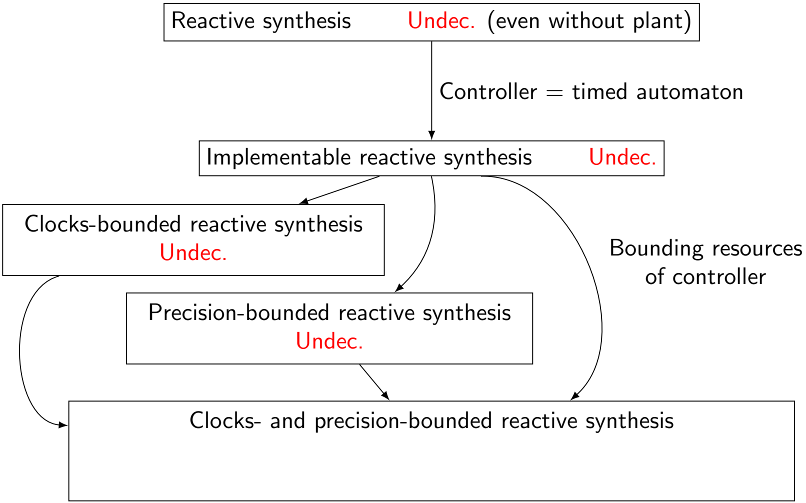
Contribution: reactive synthesis for MITL



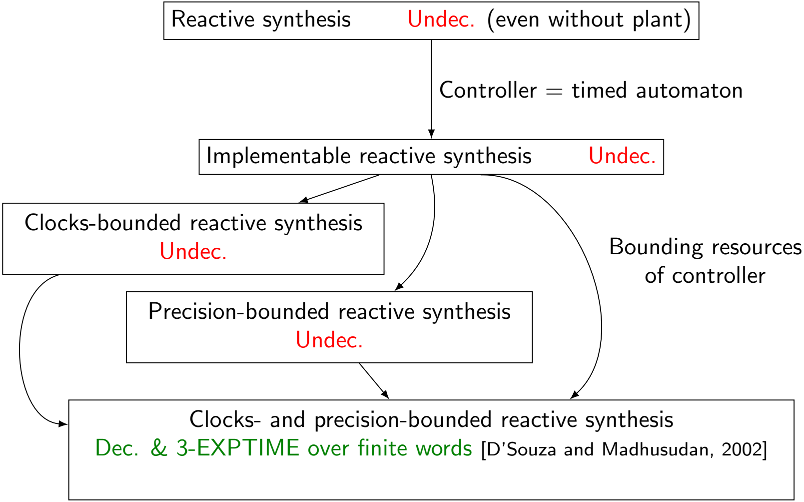
Contribution: reactive synthesis for MITL



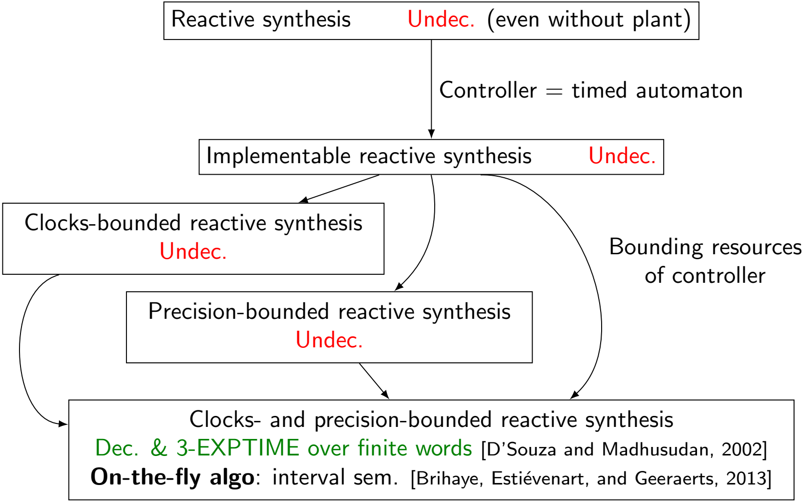
Contribution: reactive synthesis for MITL



Contribution: reactive synthesis for MITL

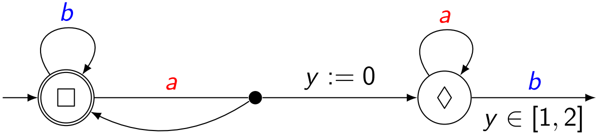


Contribution: reactive synthesis for MITL



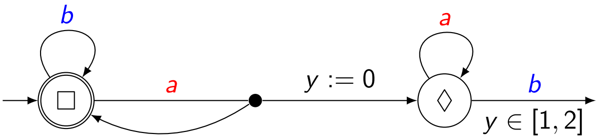
From MTL to OCATA

$$\varphi = \Box(a \Rightarrow \Diamond[[1,2]]b)$$

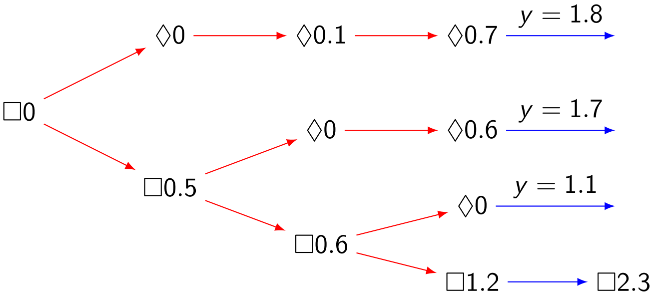


From MTL to OCATA

$$\varphi = \Box(a \Rightarrow \Diamond_{[1,2]} b)$$



Execution on the timed word $(a, 0.5)(a, 0.6)(a, 1.2)(b, 2.3)$:



Bounded-resources reactive synthesis for MTL

- Action (a, g, R)
  - a: an action in $\Sigma_C \cup \Sigma_C$
  - g: guard over clocks of X and $X_{\mathcal{P}}$
  - R: resets of clocks of X

$$\boxed{(q, \{H_1, H_2, \ldots, H_n\})} \xrightarrow{a, g, R} \boxed{(q', \{H'_1, H'_2, \ldots, H'_{n'}\})}$$

Bounded-resources reactive synthesis for MTL

- Action (a, g, R)
  - a: an action in $\Sigma_C \cup \Sigma_C$
  - g: guard over clocks of X and $X_{\mathcal{P}}$
  - R: resets of clocks of X

$$\begin{tikzcd} & \boxed{(q, \{H_1, H_2, \dots, H_n\})} \arrow[dl, "{a, g, R}"] \\ \boxed{(q', \{H'_1, H'_2, \dots, H'_{n'}\})} & \end{tikzcd}$$

- Finite abstraction is a (time-abstract) bisimulation

Bounded-resources reactive synthesis for MTL

- Action (a, g, R)
  - a: an action in $\Sigma_C \cup \Sigma_C$
  - g: guard over clocks of X and $X_{\mathcal{P}}$
  - R: resets of clocks of X

$$\boxed{(q, \{H_1, H_2, \ldots, H_n\})} \xrightarrow{a, g, R} \boxed{(q', \{H'_1, H'_2, \ldots, H'_{n'}\})}$$

- Finite abstraction is a (time-abstract) bisimulation
- Sufficient to detect when a bad configuration has been reached: one H_i contains only accepting locations of the OCATA $\mathcal{A}$ ($\equiv \neg\varphi$)

Bounded-resources reactive synthesis for MTL

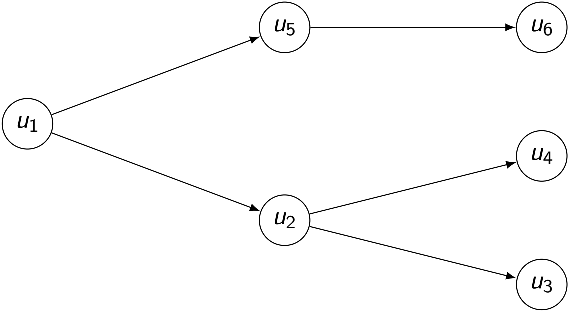
- Action (a, g, R)
  - a: an action in $\Sigma_C \cup \Sigma_C$
  - g: guard over clocks of X and $X_{\mathcal{P}}$
  - R: resets of clocks of X

$$\boxed{(q, \{H_1, H_2, \ldots, H_n\})} \xrightarrow{a, g, R} \boxed{(q', \{H'_1, H'_2, \ldots, H'_{n'}\})}$$

- Finite abstraction is a (time-abstract) bisimulation
- Sufficient to detect when a bad configuration has been reached: one H_i contains only accepting locations of the OCATA $\mathcal{A}$ ($\equiv \neg\varphi$)
- If tree finite and winning strategy: we have a (finite) controller $\mathcal{T}$

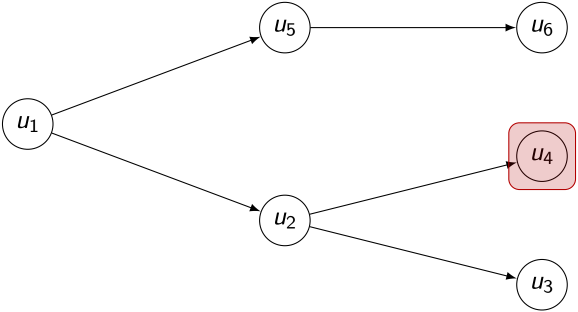
Make the tree finite

For MTL specifications [Bouyer, Bozzelli, and Chevalier, 2006]: stop the computation with a well-quasi order $\sqsubseteq$ on the labels of the nodes



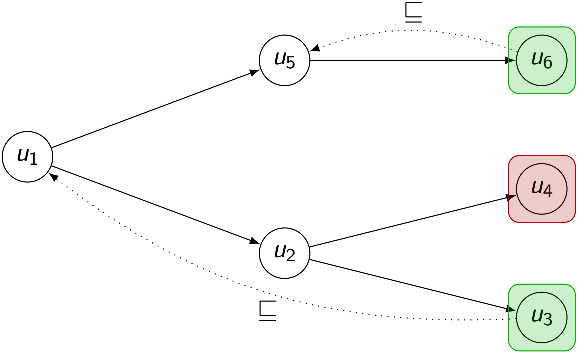
Make the tree finite

For MTL specifications [Bouyer, Bozzelli, and Chevalier, 2006]: stop the computation with a well-quasi order $\sqsubseteq$ on the labels of the nodes



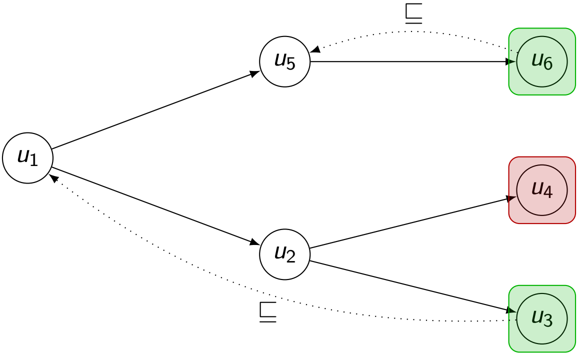
Make the tree finite

For MTL specifications [Bouyer, Bozzelli, and Chevalier, 2006]: stop the computation with a well-quasi order $\sqsubseteq$ on the labels of the nodes



Make the tree finite

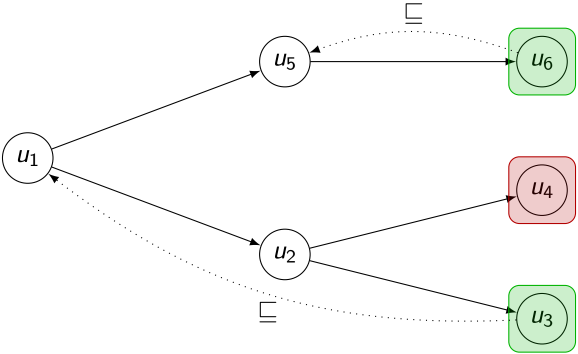
For MTL specifications [Bouyer, Bozzelli, and Chevalier, 2006]: stop the computation with a well-quasi order $\sqsubseteq$ on the labels of the nodes



- Correctness: this finite tree is **sufficient** to answer the problem

Make the tree finite

For MTL specifications [Bouyer, Bozzelli, and Chevalier, 2006]: stop the computation with a well-quasi order $\sqsubseteq$ on the labels of the nodes



- Correctness: this finite tree is **sufficient** to answer the problem
- Complexity: **non-primitive recursive** due to well-quasi orderings

Make the tree finite for MITL

- The tree is finite by using interval semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]: triply-exponential size

Make the tree finite for MITL

- The tree is finite by using interval semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]: triply-exponential size
- We obtain the same complexity as [D'Souza and Madhusudan, 2002], but with an on-the-fly exploration: may terminate more quickly

Make the tree finite for MITL

- The tree is finite by using interval semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]: triply-exponential size
- We obtain the same complexity as [D'Souza and Madhusudan, 2002], but with an on-the-fly exploration: may terminate more quickly
- Experimental results on a scheduling problem

Realisable instances

T	n	# clocks	exec. time (sec) / #nodes
1	1	0	46 / 52
1	1	1	199 / 147
1	1	2	4,599 / 1,343
2	2	1	2,632 / 645
2	2	2	18,453 / 2,358
3	3	1	182,524 / 2,297
3	3	2	$>$5min
4	4	0	54,893 / 667
4	4	1	$>$5min

Unealisable instances

T	n	# clocks	exec. time (sec) / #nodes
2	1	0	77 / 84
2	1	1	824 / 311
2	1	2	3,079 / 1,116
3	2	1	17,134 / 1698
3	2	2	$>$5min
4	3	0	10,621 / 540
4	3	1	$>$5min

Make the tree finite for MITL

- The tree is finite by using interval semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]: triply-exponential size
- We obtain the same complexity as [D'Souza and Madhusudan, 2002], but with an on-the-fly exploration: may terminate more quickly
- Experimental results on a scheduling problem

Realisable instances

T	n	# clocks	exec. time (sec) / #nodes
1	1	0	46 / 52
1	1	1	199 / 147
1	1	2	4,599 / 1,343
2	2	1	2,632 / 645
2	2	2	18,453 / 2,358
3	3	1	182,524 / 2,297
3	3	2	>5min
4	4	0	54,893 / 667
4	4	1	>5min

Unealisable instances

T	n	# clocks	exec. time (sec) / #nodes
2	1	0	77 / 84
2	1	1	824 / 311
2	1	2	3,079 / 1,116
3	2	1	17,134 / 1698
3	2	2	>5min
4	3	0	10,621 / 540
4	3	1	>5min

- Can handle small but non-trivial examples: but do not scale well

Make the tree finite for MITL

- The tree is finite by using interval semantics for OCATA [Brihaye, Estiévenart, and Geeraerts, 2013]: triply-exponential size
- We obtain the same complexity as [D'Souza and Madhusudan, 2002], but with an on-the-fly exploration: may terminate more quickly
- Experimental results on a scheduling problem

Realisable instances

T	n	# clocks	exec. time (sec) / #nodes
1	1	0	46 / 52
1	1	1	199 / 147
1	1	2	4,599 / 1,343
2	2	1	2,632 / 645
2	2	2	18,453 / 2,358
3	3	1	182,524 / 2,297
3	3	2	$>$5min
4	4	0	54,893 / 667
4	4	1	$>$5min

Unealisable instances

T	n	# clocks	exec. time (sec) / #nodes
2	1	0	77 / 84
2	1	1	824 / 311
2	1	2	3,079 / 1,116
3	2	1	17,134 / 1698
3	2	2	$>$5min
4	3	0	10,621 / 540
4	3	1	$>$5min

- Can handle small but non-trivial examples: but do not scale well
- This was before MIGHTYL, which could make things easier...

Summary for the reactive synthesis for MITL

For almost all reactive synthesis problems, MITL is as hard as MTL...

Summary for the reactive synthesis for MITL

For almost all reactive synthesis problems, MITL is as hard as MTL...

... except for **resources-bounded problem** over finite words:

- Non-elementary for MTL;
- 3-EXPTIME for MITL;
- on-the-fly algorithm

Summary for the reactive synthesis for MITL

For almost all reactive synthesis problems, MITL is as hard as MTL...

... except for **resources-bounded problem** over finite words:

- Non-elementary for MTL;
- 3-EXPTIME for MITL;
- on-the-fly algorithm

Other fragments?? Hopeless!

	Safety-MTL	coFlat-MTL	Open-MITL	Closed-MITL
implementable RS				
clock-bounded RS				
precision-bounded RS				

Summary for the reactive synthesis for MITL

For almost all reactive synthesis problems, MITL is as hard as MTL...

... except for **resources-bounded problem** over finite words:

- Non-elementary for MTL;
- 3-EXPTIME for MITL;
- on-the-fly algorithm

Other fragments?? Hopeless!

	Safety-MTL	coFlat-MTL	Open-MITL	Closed-MITL
implementable RS	undec.	undec.	undec.	undec.
clock-bounded RS	undec.	undec.	undec.	undec.
precision-bounded RS	undec.	undec.	undec.	undec.

Summary for the reactive synthesis for MITL

For almost all reactive synthesis problems, MITL is as hard as MTL...

... except for **resources-bounded problem** over finite words:

- Non-elementary for MTL;
- 3-EXPTIME for MITL;
- on-the-fly algorithm

Other fragments?? Hopeless!

	Safety-MTL	coFlat-MTL	Open-MITL	Closed-MITL
implementable RS	undec.	undec.	undec.	undec.
clock-bounded RS	undec.	undec.	undec.	undec.
precision-bounded RS	undec.	undec.	undec.	undec.

Possible future directions:

- Decidable fragments for BPrecRS/BClockRS
- Heuristics for speed-up for the on-the-fly algorithm: well-quasi orderings as in [Bouyer, Bozzelli, and Chevalier, 2006], zone-based versions?
- Experiments of the on-the-fly algorithm over the fragments
- Robustness of controllers

Summary for the reactive synthesis for MITL

For almost all reactive synthesis problems, MITL is as hard as MTL...

... except for **resources-bounded problem** over finite words:

- Non-elementary for MTL;
- 3-EXPTIME for MITL;
- on-the-fly algorithm

Other fragments?? Hopeless!

	Safety-MTL	coFlat-MTL	Open-MITL	Closed-MITL
implementable RS	undec.	undec.	undec.	undec.
clock-bounded RS	undec.	undec.	undec.	undec.
precision-bounded RS	undec.	undec.	undec.	undec.

Possible future directions:

- Decidable fragments for BPrecRS/BClockRS
- Heuristics for speed-up for the on-the-fly algorithm: well-quasi orderings as in [Bouyer, Bozzelli, and Chevalier, 2006], zone-based versions?
- Experiments of the on-the-fly algorithm over the fragments
- Robustness of controllers

Thank you for your attention! Questions?

References I

Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 (2):183–235, 1994.

Rajeev Alur and Thomas A. Henzinger. A really temporal logic. In *30th Annual Symposium on Foundations of Computer Science (FOCS'89)*, pages 164–169. IEEE Computer Society Press, 1989. doi: 10.1109/SFCS.1989.63473.

Rajeev Alur, Costas A. Courcoubetis, and David L. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90)*, pages 414–425. IEEE Computer Society Press, 1990. doi: 10.1109/LICS.1990.113766.

Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211(1-2):253–273, 1999.

Marcello M. Bersani, Matteo Rossi, and Pierluigi San Pietro. An SMT-based approach to satisfiability checking of MITL. *Information and Computation*, 245:72–97, 2015.

Patricia Bouyer, Laura Bozzelli, and Fabrice Chevalier. Controller synthesis for MTL specifications. In *Proceedings of the 17th International Conference on Concurrency Theory (CONCUR'06)*, volume 4137 of *Lecture Notes in Computer Science*, pages 450–464. Springer, 2006.

Thomas Brihaye, Morgane Estiévenart, and Gilles Geeraerts. On MITL and alternating timed automata. In *Proceedings of the 11th international conference on Formal Modeling and Analysis of Timed Systems (FORMATS'13)*, volume 8053 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2013.

Thomas Brihaye, Morgane Estiévenart, and Gilles Geeraerts. On MITL and alternating timed automata of infinite words. In *Proceedings of the 12th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'14)*, volume 8711 of *Lecture Notes in Computer Science*. Springer, 2014.

References II

Laurent Doyen, Gilles Geeraerts, Jean-François Raskin, and Julien Reichert. Realizability of real-time logics. In *Proceedings of the 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'09)*, volume 5813 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2009.

Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proceedings of the 19th Annual conference on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *Lecture Notes in Computer Science*, pages 571–582. Springer, 2002.

Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

Thomas A. Henzinger. It's about time: Real-time logics reviewed. In *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR '98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 439–454. Springer, 1998. doi: 10.1007/BFb0055640.

Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The regular real-time languages. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 580–591. Springer, 1998. doi: 10.1007/BFb0055086.

Roland Kindermann, Tommi A. Junttila, and Ilkka Niemelä. Bounded model checking of an MITL fragment for timed automata. In *Proceedings of the 13th International Conference on Application of Concurrency to System Design (ACSD'13)*, pages 216–225. IEEE Computer Society Press, 2013. doi: 10.1109/ACSD.2013.25.

Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2 (4):255–299, 1990.

References III

Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In *Proceedings of the Third International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2005.

Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theoretical Computer Science*, 32:321–330, 1984. doi: 10.1016/0304-3975(84)90049-5.

Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Proceedings of the 20th Annual Symposium on Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Computer Society Press, 2005.

Joël Ouaknine and James Worrell. Safety metric temporal logic is fully decidable. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2006.

Jean-François Raskin and Pierre-Yves Schobbens. The logic of event clocks: Decidability, complexity and expressiveness. *Journal of Automata, Languages and Combinatorics*, 4(3): 247–282, 1999.

Moshe Y. Vardi. Reasoning about the past with two-way automata. In Kim G. Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641, 1998.

Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 694–715. Springer, 1994.

Bożena Woźna-Szcześniak, Ireneusz Szcześniak, Agnieszka M. Zbrzezny, and Andrzej Zbrzezny. Bounded model checking for weighted interpreted systems and for flat weighted epistemic computation tree logic. In *Proceedings of the 17th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'14)*, volume 8861 of *Lecture Notes in Artificial Intelligence*, pages 107–115. Springer, 2014.