A Brief History of Real-Time

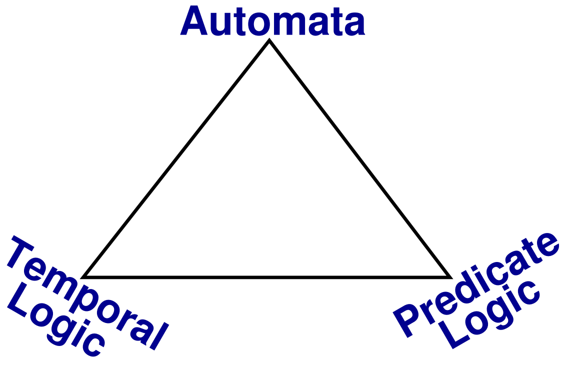
Joël Ouaknine

Department of Computer Science, Oxford University &

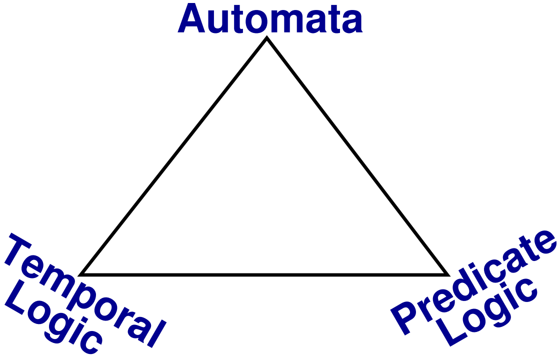
Max Planck Institute for Software Systems

EQINOCS Workshop, Paris, May 2016

The Classical Linear Theory of Verification

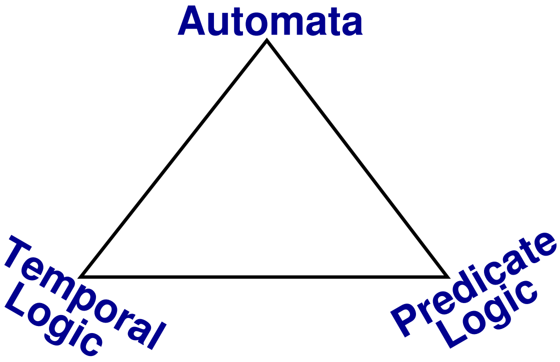


The Classical Linear Theory of Verification



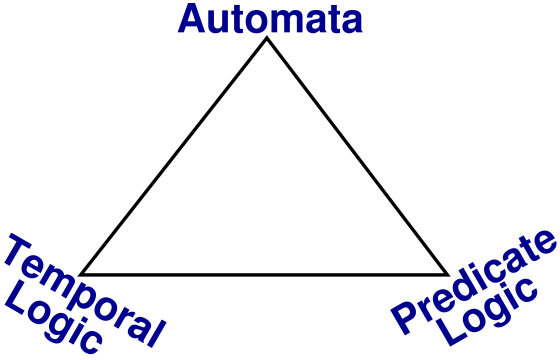
- Qualitative (order-theoretic), rather than quantitative (metric).

The Classical Linear Theory of Verification



- Qualitative (order-theoretic), rather than quantitative (metric).
- Time is modelled as the naturals $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

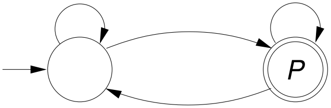
The Classical Linear Theory of Verification



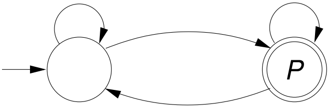
- Qualitative (order-theoretic), rather than quantitative (metric).
- Time is modelled as the naturals $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- Note: focus on linear time (as opposed to branching time).

‘P occurs infinitely often’

‘P occurs infinitely often’

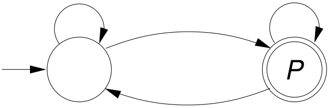


‘P occurs infinitely often’



$\Box\Diamond P$

‘P occurs infinitely often’



$\Box\Diamond P$ $\qquad\qquad$ $\forall x\, \exists y\, (x < y \wedge P(y))$

Specification and Verification

- Linear Temporal Logic (LTL)

$$\theta ::= P \mid \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2 \mid \neg\theta \mid \bigcirc\theta \mid \Diamond\theta \mid \Box\theta \mid \theta_1\, \mathcal{U}\, \theta_2$$

For example, $\Box(REQ \rightarrow \Diamond ACK)$.

Specification and Verification

- Linear Temporal Logic (LTL)

  $$\theta ::= P \mid \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2 \mid \neg\theta \mid \bigcirc\theta \mid \Diamond\theta \mid \Box\theta \mid \theta_1 \,\mathcal{U}\, \theta_2$$

  For example, $\Box(REQ \rightarrow \Diamond ACK)$.

- First-Order Logic (FO($<$))

  $$\varphi ::= x < y \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\, \varphi \mid \exists\, x\varphi$$

  For example, $\forall x\, (REQ(x) \rightarrow \exists y\, (x < y \wedge ACK(y)))$.

Specification and Verification

- Linear Temporal Logic (LTL)

  $$\theta ::= P \mid \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2 \mid \neg\theta \mid \bigcirc\theta \mid \Diamond\theta \mid \Box\theta \mid \theta_1\, \mathcal{U}\, \theta_2$$

  For example, $\Box(REQ \rightarrow \Diamond ACK)$.

- First-Order Logic (FO($<$))

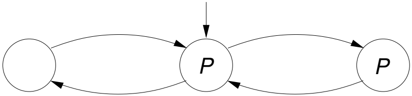
  $$\varphi ::= x < y \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\, \varphi \mid \exists\, x\varphi$$

  For example, $\forall x\, (REQ(x) \rightarrow \exists y\, (x < y \wedge ACK(y)))$.

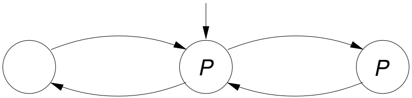
Verification is model checking: IMP $\models$ SPEC ?

‘P holds at every even position (and may or may not hold at odd positions)’

'P holds at every even position
(and may or may not hold at odd positions)'

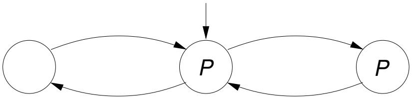


'P holds at every even position
(and may or may not hold at odd positions)'



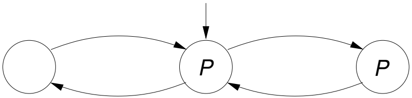
- It turns out it is impossible to capture this requirement using LTL or FO($<$).

‘P holds at every even position
(and may or may not hold at odd positions)’



- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *‘Q holds precisely at even positions’*:

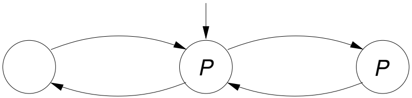
'P holds at every even position
(and may or may not hold at odd positions)'



- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

$$Q \wedge \Box(Q \to \bigcirc\neg Q) \wedge \Box(\neg Q \to \bigcirc Q)$$

'P holds at every even position (and may or may not hold at odd positions)'

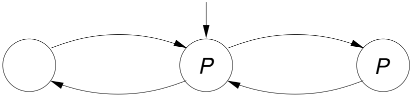


- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

$$Q \wedge \Box(Q \rightarrow \bigcirc\neg Q) \wedge \Box(\neg Q \rightarrow \bigcirc Q)$$

- So one way to capture the original specification would be to write:

'P holds at every even position
(and may or may not hold at odd positions)'

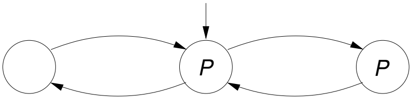


- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

$$Q \wedge \Box(Q \rightarrow \bigcirc\neg Q) \wedge \Box(\neg Q \rightarrow \bigcirc Q)$$

- So one way to capture the original specification would be to write: *'Q holds precisely at even positions* **and** $\Box(Q \rightarrow P)$*'*.

'P holds at every even position
(and may or may not hold at odd positions)'

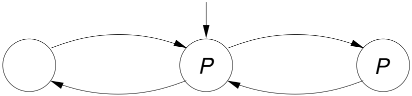


- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

$$Q \wedge \Box(Q \rightarrow \bigcirc\neg Q) \wedge \Box(\neg Q \rightarrow \bigcirc Q)$$

- So one way to capture the original specification would be to write: *'Q holds precisely at even positions* **and** $\Box(Q \rightarrow P)$'.
- Finally, need to existentially quantify Q out:

'P holds at every even position
(and may or may not hold at odd positions)'



- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

$$Q \wedge \Box(Q \rightarrow \bigcirc\neg Q) \wedge \Box(\neg Q \rightarrow \bigcirc Q)$$

- So one way to capture the original specification would be to write: *'Q holds precisely at even positions* **and** $\Box(Q \rightarrow P)$*'*.
- Finally, need to existentially quantify Q out:

$$\exists Q\ (Q \textit{ holds precisely at even positions } \textbf{and } \Box\,(Q \rightarrow P))$$

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO($<$))

$$\varphi ::= x < y \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \forall P\, \varphi \mid \exists P\, \varphi$$

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO($<$))

$\varphi ::= x < y \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \forall P\, \varphi \mid \exists P\, \varphi$

Theorem (Büchi 1960)

Any MSO($<$) *formula* φ *can be effectively translated into an equivalent automaton* A_φ.

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO($<$))

$\varphi ::= x < y \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \forall P\, \varphi \mid \exists P\, \varphi$

Theorem (Büchi 1960)

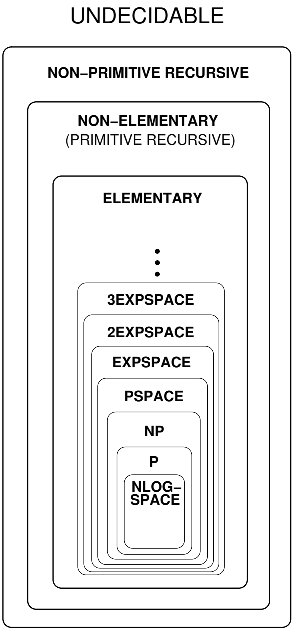
Any MSO($<$) *formula* φ *can be effectively translated into an equivalent automaton* A_φ*.*

Corollary (Church 1960)

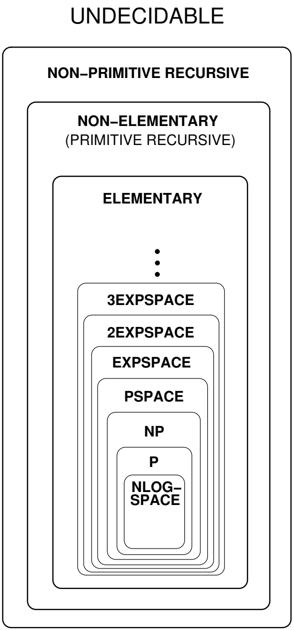
The model-checking problem for automata against MSO($<$) *specifications is decidable:*

$$M \models \varphi \quad \textit{iff} \quad L(M) \cap L(A_{\neg\varphi}) = \emptyset$$

Complexity


UNDECIDABLE
NON–PRIMITIVE RECURSIVE
NON–ELEMENTARY
(PRIMITIVE RECURSIVE)
ELEMENTARY
3EXPSPACE
2EXPSPACE
EXPSPACE
PSPACE
NP
P
NLOG–SPACE

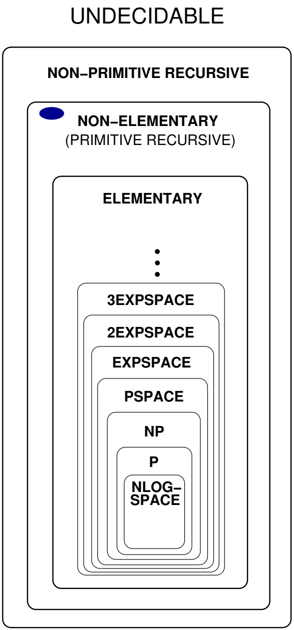
Complexity



- NON-ELEMENTARY: $\underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{n}}}}}}}_{n}$
- NON-PRIMITIVE RECURSIVE:

  Ackerman: $3, 4, 8, 2048, \underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}}_{2048}, \ldots$

Complexity



- NON-ELEMENTARY: $\underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{n}}}}}}}_{n}$
- NON-PRIMITIVE RECURSIVE:

  Ackerman: 3, 4, 8, 2048, $\underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}}_{2048}$, ...

Complexity and Equivalence

In fact:

Theorem (Stockmeyer 1974)

FO($<$) *satisfiability has non-elementary complexity.*

Complexity and Equivalence

In fact:

Theorem (Stockmeyer 1974)

FO($<$) *satisfiability has non-elementary complexity.*

**Theorem (Kamp 1968;
Gabbay, Pnueli, Shelah, Stavi 1980)**

LTL *and* FO($<$) *have precisely the same expressive power.*

Complexity and Equivalence

In fact:

Theorem (Stockmeyer 1974)

FO($<$) *satisfiability has non-elementary complexity.*

**Theorem (Kamp 1968;
Gabbay, Pnueli, Shelah, Stavi 1980)**

LTL *and* FO($<$) *have precisely the same expressive power.*

But amazingly:

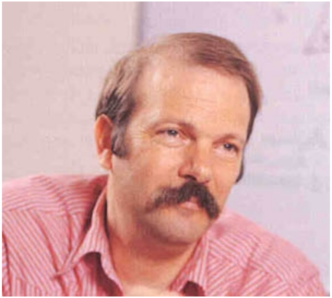
Theorem (Sistla & Clarke 1982)

LTL *satisfiability and model checking are PSPACE-complete.*

Logics and Automata

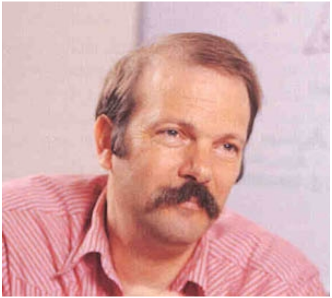
"The paradigmatic idea of the automata-theoretic approach to verification is that we can compile high-level logical specifications into an equivalent low-level finite-state formalism."

Moshe Vardi



Logics and Automata

"The paradigmatic idea of the automata-theoretic approach to verification is that we can compile high-level logical specifications into an equivalent low-level finite-state formalism."



Moshe Vardi

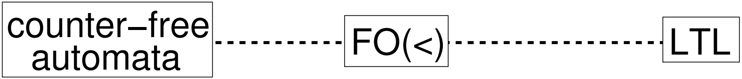
Theorem

Automata are closed under all Boolean operations. Moreover, the language inclusion problem ($L(A) \subseteq L(B)$?) is PSPACE-complete.

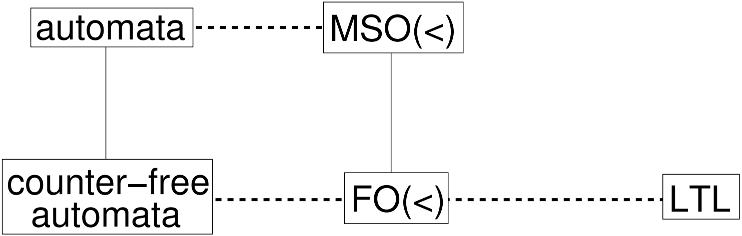
The Classical Theory: Expressiveness

$\boxed{\mathrm{FO}(<)}$ - - - - - - - - - - - - - - - - - - $\boxed{\mathrm{LTL}}$

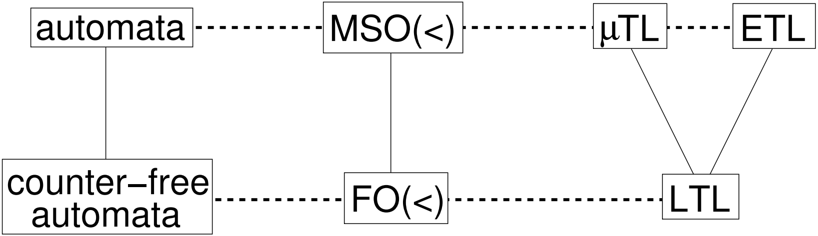
The Classical Theory: Expressiveness



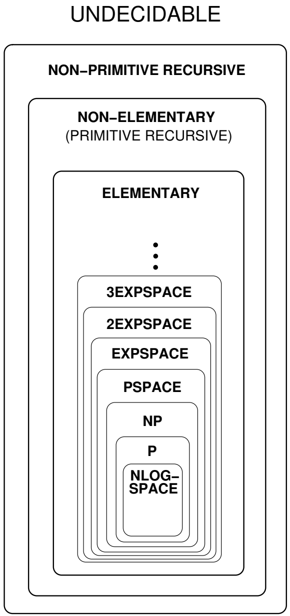
The Classical Theory: Expressiveness



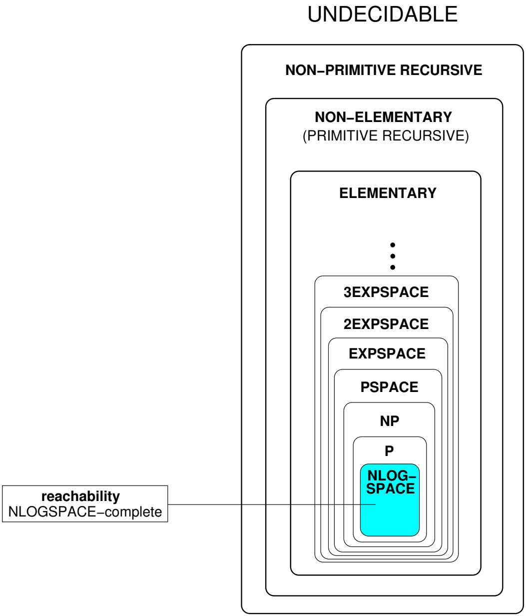
The Classical Theory: Expressiveness



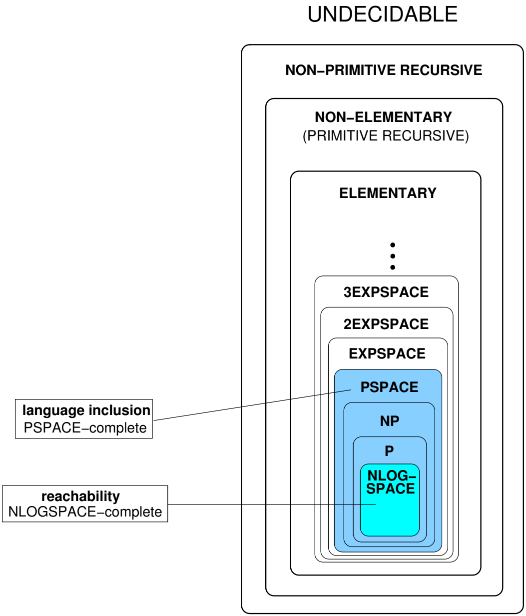
The Classical Theory: Complexity



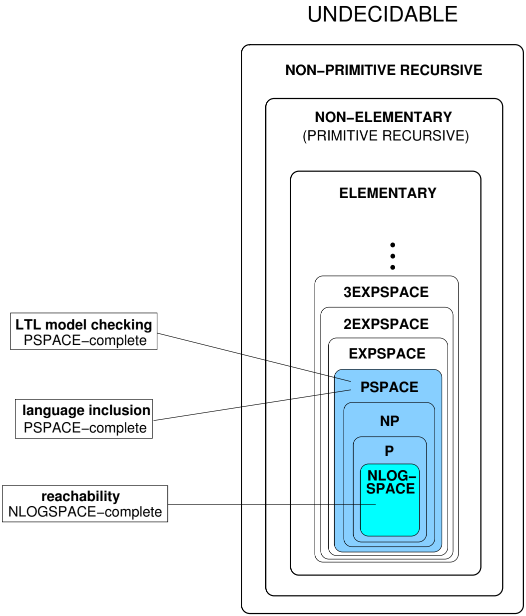
The Classical Theory: Complexity



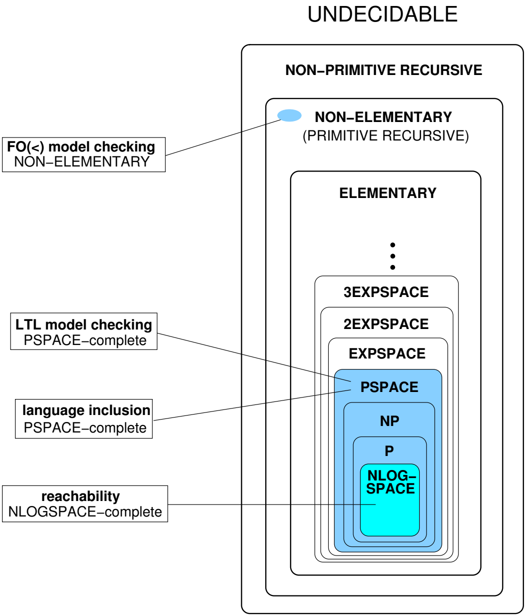
The Classical Theory: Complexity



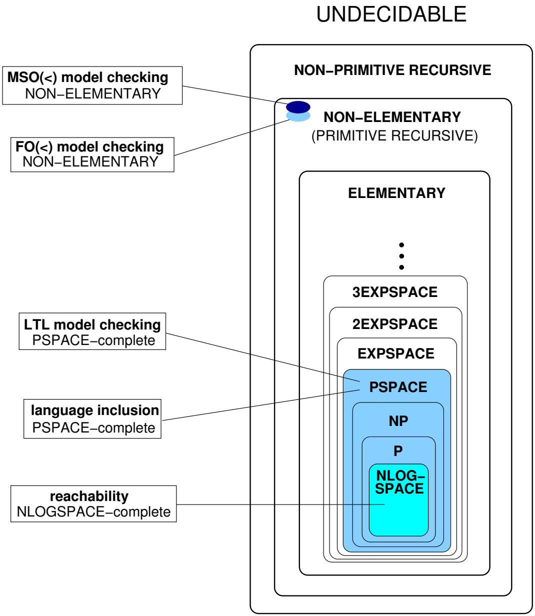
The Classical Theory: Complexity



The Classical Theory: Complexity



The Classical Theory: Complexity



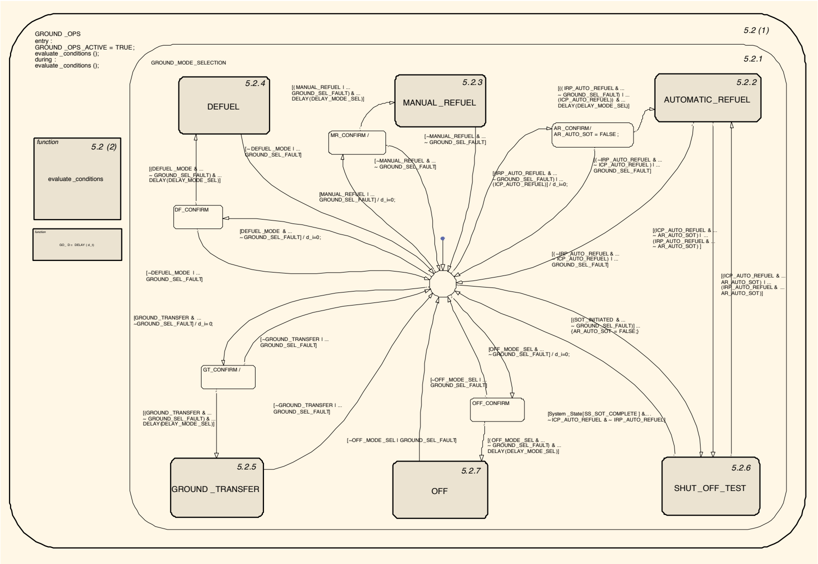
From Qualitative to Quantitative

"Lift the classical theory to the real-time world."

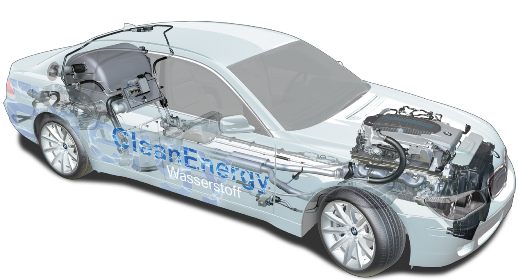
Boris Trakhtenbrot, LICS 1995

Airbus A350 XWB

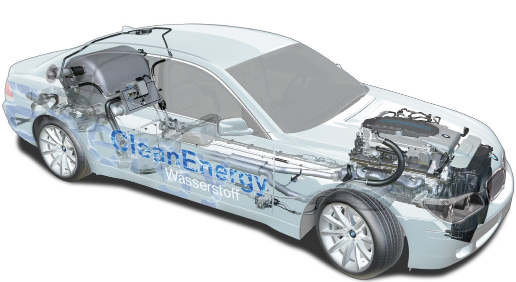
A350 XWB Fuel Management Sub-System

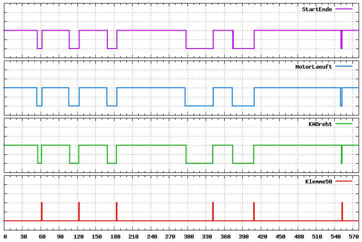


BMW Hydrogen 7



BMW Hydrogen 7

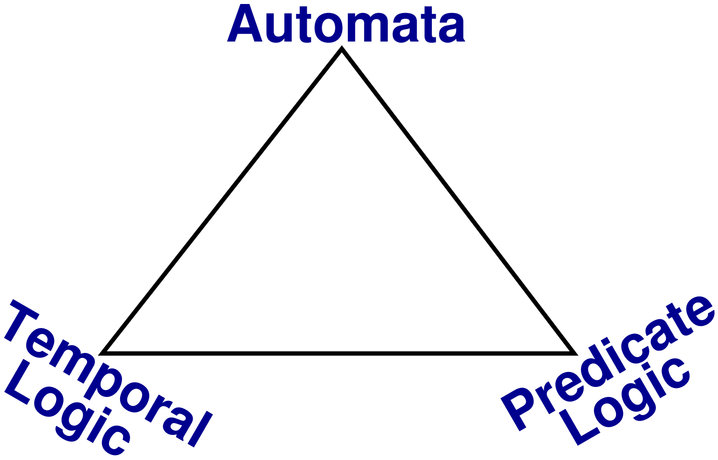




Timed Systems

Timed systems are everywhere…

- Hardware circuits
- Communication protocols
- Cell phones
- Plant controllers
- Aircraft navigation systems
- Sensor networks
- …


Automata
Temporal Logic
Predicate Logic

Timed Automata

Timed automata were introduced by Rajeev Alur at Stanford during his PhD thesis under David Dill:

- Rajeev Alur, David L. Dill: *Automata For Modeling Real-Time Systems*. ICALP 1990: 322-335
- Rajeev Alur, David L. Dill: *A Theory of Timed Automata*. TCS 126(2): 183-235, 1994

Timed Automata

Timed automata were introduced by Rajeev Alur at Stanford during his PhD thesis under David Dill:

- Rajeev Alur, David L. Dill: *Automata For Modeling Real-Time Systems*. ICALP 1990: 322-335
- Rajeev Alur, David L. Dill: *A Theory of Timed Automata*. TCS 126(2): 183-235, 1994

$\Rightarrow$ Led to inaugural CAV Award (2008) *and* inaugural Church Award (2016)!

Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Theorem (Alur, Courcourbetis, Dill 1990)

Reachability is decidable, in fact PSPACE-complete.

Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Theorem (Alur, Courcourbetis, Dill 1990)

Reachability is decidable, in fact PSPACE-complete.

$\Rightarrow$ LICS Test-of-Time Award (2010)

Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Theorem (Alur, Courcourbetis, Dill 1990)

Reachability is decidable, in fact PSPACE-complete.

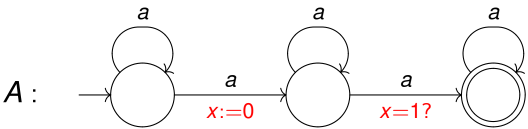
$\Rightarrow$ LICS Test-of-Time Award (2010)

Unfortunately:

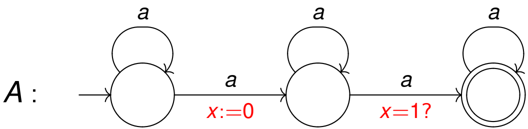
Theorem (Alur & Dill 1990)

Language inclusion is undecidable for timed automata.

An Uncomplementable Timed Automaton

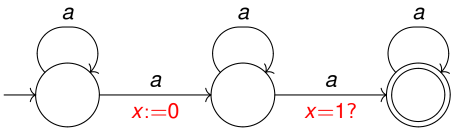


An Uncomplementable Timed Automaton



An Uncomplementable Timed Automaton

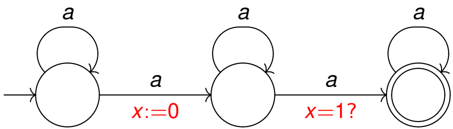
A :



$L(A)$:

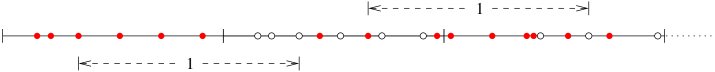
An Uncomplementable Timed Automaton

A :

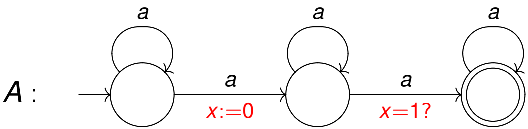


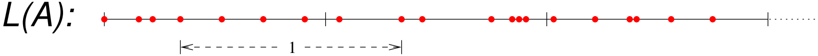
$L(A)$:

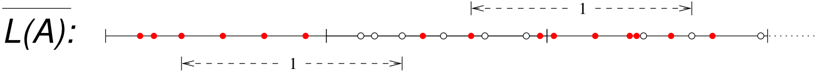
$\overline{L(A)}$:



An Uncomplementable Timed Automaton







A cannot be complemented:
There is no timed automaton *B* with $L(B) = \overline{L(A)}$.

Metric Temporal Logic

Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli $\sim$1990] is a central quantitative specification formalism for timed systems.

Metric Temporal Logic

Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli $\sim$1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:

$$\Box(\mathit{PEDAL} \rightarrow \Diamond_{[5,10]}\, \mathit{BRAKE})$$

Metric Temporal Logic

Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli $\sim$1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:

$$\Box(\mathit{PEDAL} \rightarrow \Diamond_{[5,10]}\, \mathit{BRAKE})$$

- Widely cited and used (over 1600 papers according to Google Scholar!).

Metric Temporal Logic

Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli $\sim$1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:

$$\Box(\mathit{PEDAL} \rightarrow \Diamond_{[5,10]}\, \mathit{BRAKE})$$

- Widely cited and used (over 1600 papers according to Google Scholar!).

Unfortunately:

Theorem (Alur & Henzinger 1992)

MTL *satisfiability and model checking are undecidable over* $\mathbb{R}_{\geq 0}$.

Metric Temporal Logic

Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli $\sim$1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:

$$\Box(\mathit{PEDAL} \to \Diamond_{[5,10]}\ \mathit{BRAKE})$$

- Widely cited and used (over 1600 papers according to Google Scholar!).

Unfortunately:

Theorem (Alur & Henzinger 1992)

MTL *satisfiability and model checking are undecidable over* $\mathbb{R}_{\geq 0}$. *(Decidable but non-primitive recursive under certain semantic restrictions [O. & Worrell 2005].)*

Metric Predicate Logic

The first-order metric logic of order (FO($<$,$+1$)) extends FO($<$) by the unary function '$+1$'.

Metric Predicate Logic

The first-order metric logic of order (FO($<$,$+1$)) extends FO($<$) by the unary function '$+1$'.

For example, $\Box(\mathit{PEDAL} \rightarrow \Diamond_{[5,10]} \mathit{BRAKE})$ becomes

$$\forall x\, (\mathit{PEDAL}(x) \rightarrow \exists y\, (x + 5 \leq y \leq x + 10 \wedge \mathit{BRAKE}(y)))$$

Theorem (Hirshfeld & Rabinovich 2007)

FO($<, +1$) *is strictly more expressive than* MTL *over* $\mathbb{R}_{\geq 0}$.

Theorem (Hirshfeld & Rabinovich 2007)

FO$(<, +1)$ *is strictly more expressive than* MTL *over* $\mathbb{R}_{\geq 0}$.

Theorem (Hunter, O., Worrell 2013)

FO$(<, +\mathbb{Q})$ *and* MTL$_{\mathbb{Q}}$ *have precisely the same expressive power.*

Theorem (Hirshfeld & Rabinovich 2007)

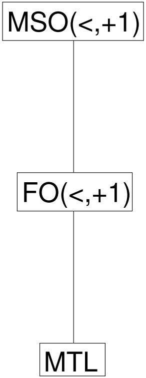
FO$(<, +1)$ *is strictly more expressive than* MTL *over* $\mathbb{R}_{\geq 0}$*.*

Theorem (Hunter, O., Worrell 2013)

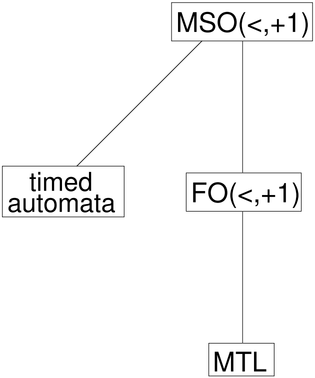
FO$(<, +\mathbb{Q})$ *and* MTL$_{\mathbb{Q}}$ *have precisely the same expressive power.*

Corollary: FO$(<, +1)$, FO$(<, +\mathbb{Q})$, MSO$(<, +1)$, MSO$(<, +\mathbb{Q})$ satisfiability and model checking are all undecidable over $\mathbb{R}_{\geq 0}$.

The Real-Time Theory: Expressiveness



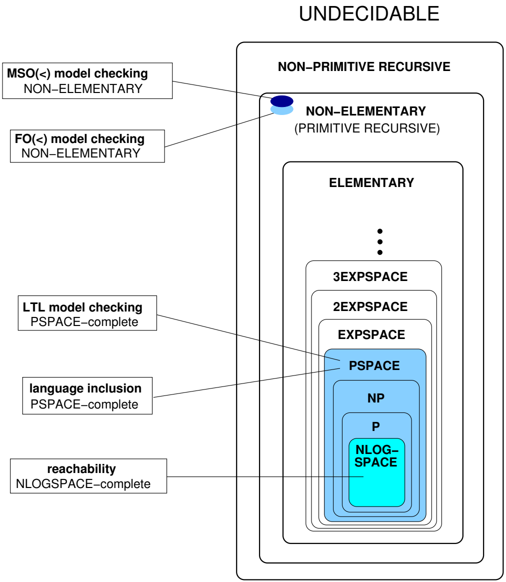
The Real-Time Theory: Expressiveness



The Real-Time Theory: Complexity

Classical Theory

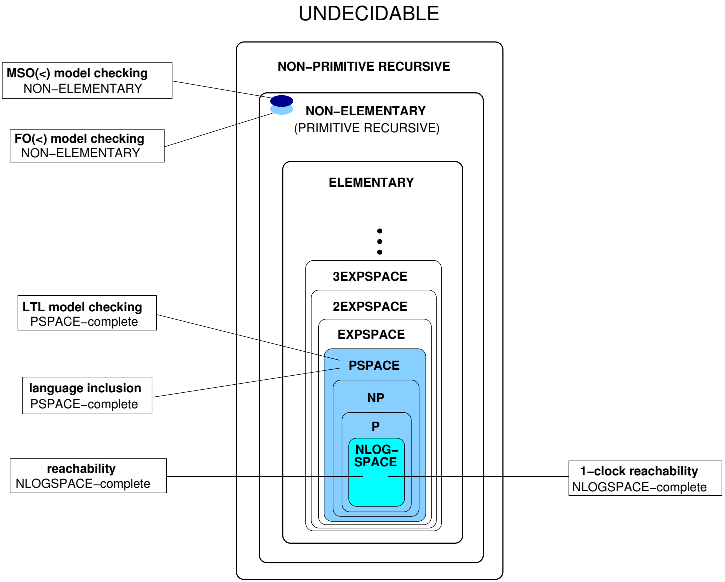
Real–Time Theory



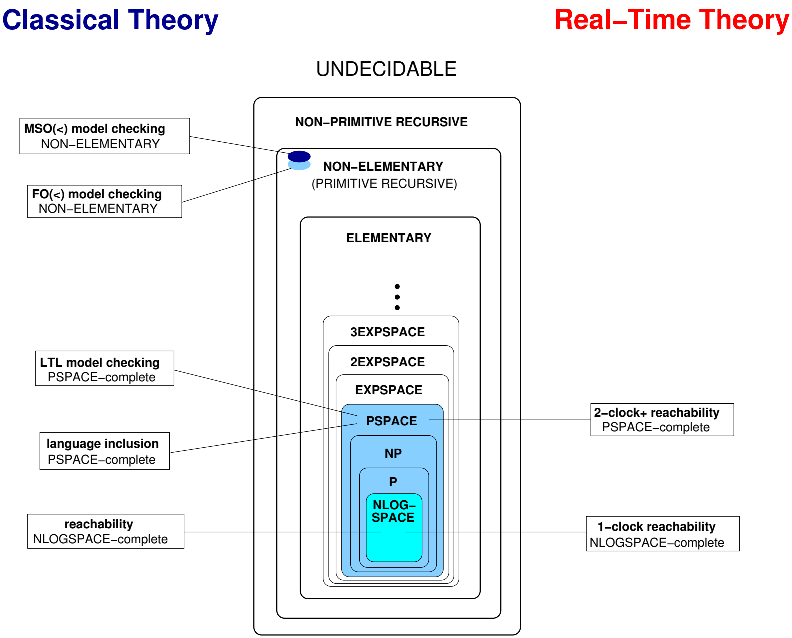
The Real-Time Theory: Complexity

Classical Theory

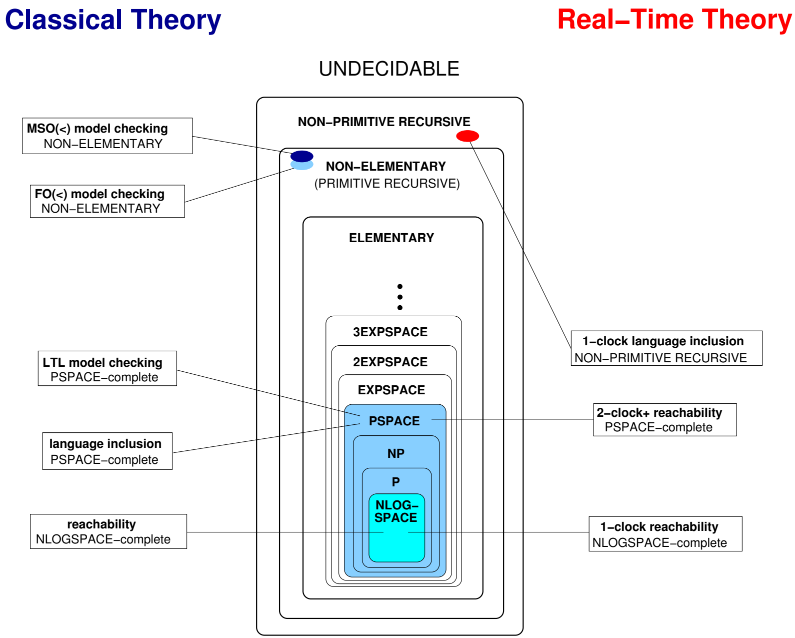
Real–Time Theory



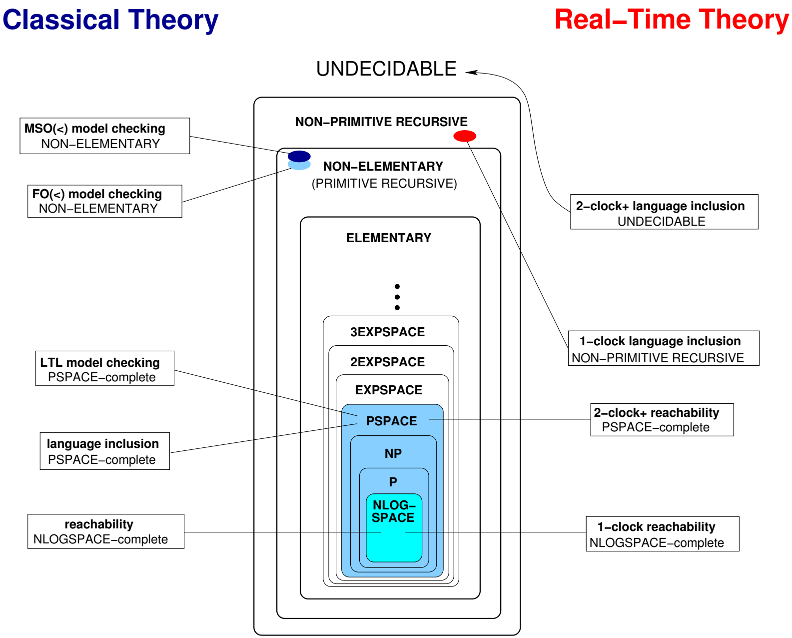
The Real-Time Theory: Complexity



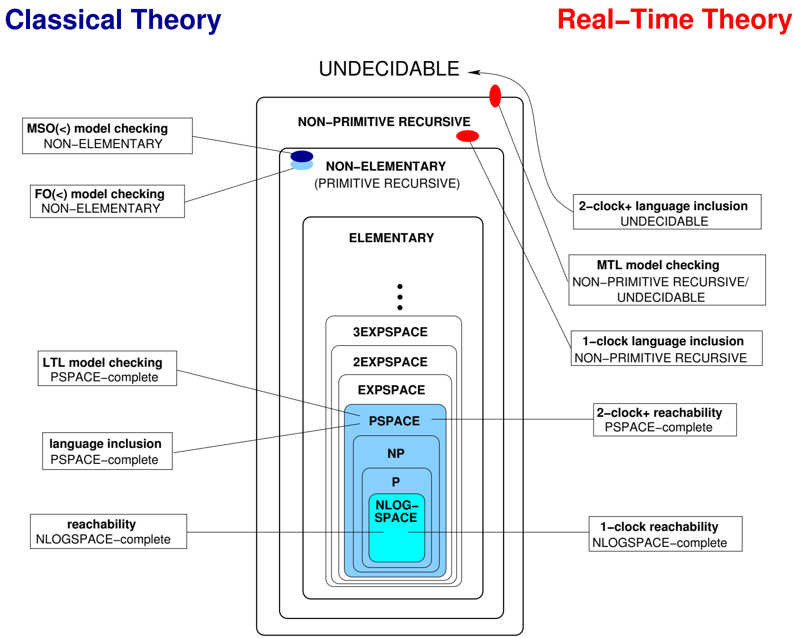
The Real-Time Theory: Complexity



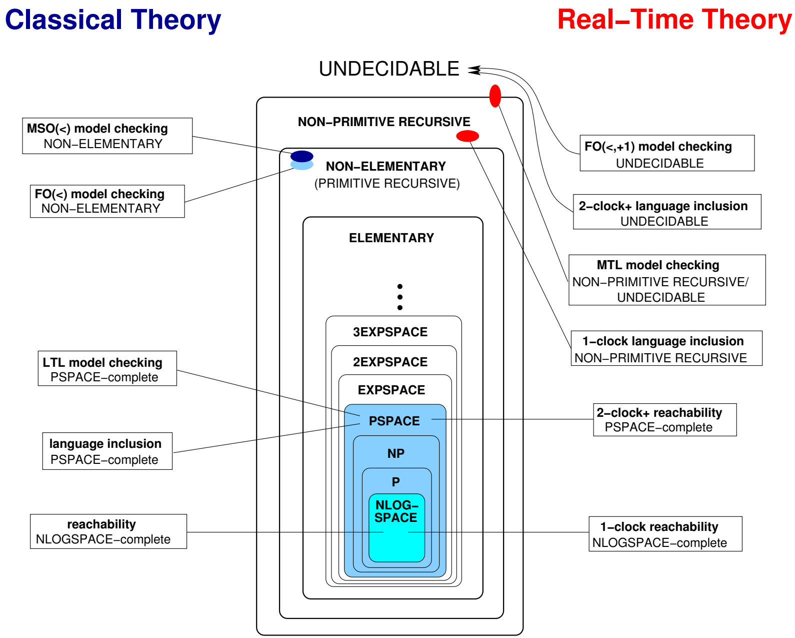
The Real-Time Theory: Complexity



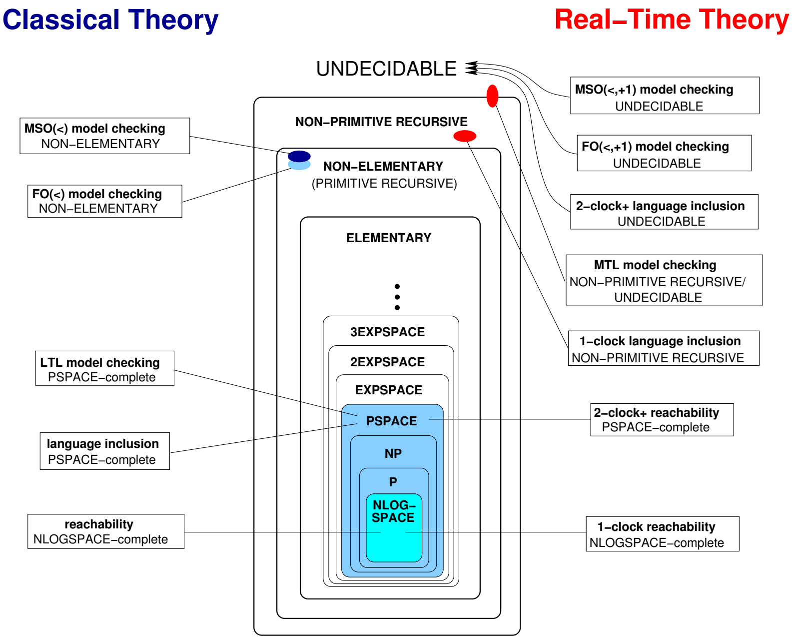
The Real-Time Theory: Complexity



The Real-Time Theory: Complexity



The Real-Time Theory: Complexity



Key Stumbling Block

Theorem (Alur & Dill 1990)

Language inclusion is undecidable for timed automata.

Timed Language Inclusion: Some Related Work

- Topological restrictions and digitization techniques: [Henzinger, Manna, Pnueli 1992], [Bošnački 1999], [O. & Worrell 2003]
- Fuzzy semantics / noise-based techniques: [Maass & Orponen 1996], [Gupta, Henzinger, Jagadeesan 1997], [Fränzle 1999], [Henzinger & Raskin 2000], [Puri 2000], [Asarin & Bouajjani 2001], [O. & Worrell 2003], [Alur, La Torre, Madhusudan 2005]
- Determinisable subclasses of timed automata: [Alur & Henzinger 1992], [Alur, Fix, Henzinger 1994], [Wilke 1996], [Raskin 1999]
- Timed simulation relations and homomorphisms: [Lynch *et al.* 1992], [Taşiran *et al.* 1996], [Kaynar, Lynch, Segala, Vaandrager 2003]
- Restrictions on the number of clocks: [O. & Worrell 2004], [Emmi & Majumdar 2006]

Time-Bounded Language Inclusion

TIME-BOUNDED LANGUAGE INCLUSION PROBLEM

Instance: Timed automata A, B, and time bound $T \in \mathbb{N}$

Question: Is $L_T(A) \subseteq L_T(B)$?

Time-Bounded Language Inclusion

TIME-BOUNDED LANGUAGE INCLUSION PROBLEM

Instance: Timed automata A, B, and time bound $T \in \mathbb{N}$

Question: Is $L_T(A) \subseteq L_T(B)$?

- Inspired by Bounded Model Checking.

Time-Bounded Language Inclusion

TIME-BOUNDED LANGUAGE INCLUSION PROBLEM

Instance: Timed automata A, B, and time bound $T \in \mathbb{N}$

Question: Is $L_T(A) \subseteq L_T(B)$?

- Inspired by Bounded Model Checking.
- Timed systems often have time bounds (e.g. timeouts), even if total number of actions is potentially unbounded.

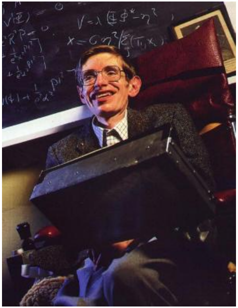
Time-Bounded Language Inclusion

TIME-BOUNDED LANGUAGE INCLUSION PROBLEM

Instance: Timed automata A, B, and time bound $T \in \mathbb{N}$

Question: Is $L_T(A) \subseteq L_T(B)$?

- Inspired by Bounded Model Checking.
- Timed systems often have time bounds (e.g. timeouts), even if total number of actions is potentially unbounded.
- Universe's lifetime is believed to be bounded anyway…



Timed Automata and Metric Logics

- Unfortunately, timed automata cannot be complemented even over bounded time…

Timed Automata and Metric Logics

- Unfortunately, timed automata cannot be complemented even over bounded time…
- Key to solution is to translate problem into logic: Behaviours of timed automata can be captured in MSO($<$,$+1$)

Timed Automata and Metric Logics

- Unfortunately, timed automata cannot be complemented even over bounded time...
- Key to solution is to translate problem into logic: Behaviours of timed automata can be captured in MSO($<$,$+1$)
- This reverses Vardi's 'automata-theoretic approach to verification' paradigm!

Monadic Second-Order Logic

Theorem (Shelah 1975)

MSO($<$) *is undecidable over* $[0,1)$.

Monadic Second-Order Logic

Theorem (Shelah 1975)

MSO($<$) *is undecidable over* $[0, 1)$.

By contrast,

Theorem

- MSO($<$) *is decidable over* $\mathbb{N}$ *[Büchi 1960]*
- MSO($<$) *is decidable over* $\mathbb{Q}$, *via [Rabin 1969]*

Finite Variability

Timed behaviours are modelled as flows (or signals):

Finite Variability

Timed behaviours are modelled as flows (or signals):

$f : [0, T) \rightarrow 2^{\mathbf{MP}}$

Finite Variability

Timed behaviours are modelled as flows (or signals):

$f : [0, T) \to 2^{\mathbf{MP}}$

P:

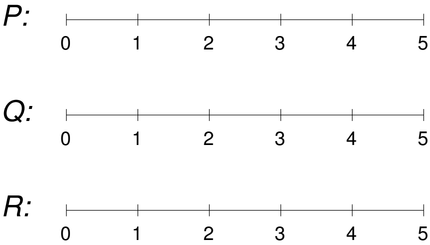
Q:

R:

Finite Variability

Timed behaviours are modelled as flows (or signals):

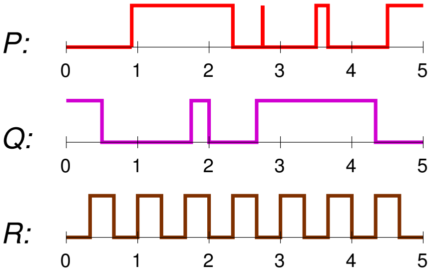
$f : [0, T) \to 2^{\mathbf{MP}}$



Finite Variability

Timed behaviours are modelled as flows (or signals):

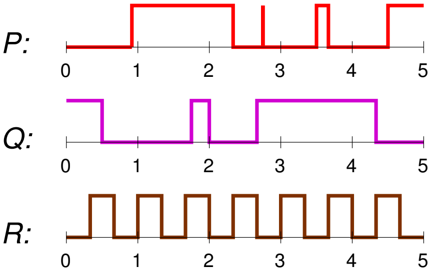
$f : [0, T) \rightarrow 2^{\mathbf{MP}}$



Finite Variability

Timed behaviours are modelled as flows (or signals):

$f : [0, T) \to 2^{\mathbf{MP}}$

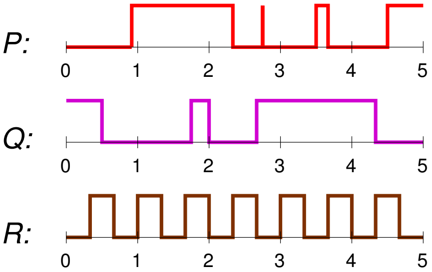


Predicates must have finite variability:

Finite Variability

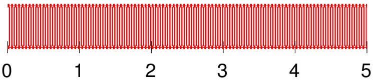
Timed behaviours are modelled as flows (or signals):

$f : [0, T) \to 2^{\mathbf{MP}}$



Predicates must have finite variability:

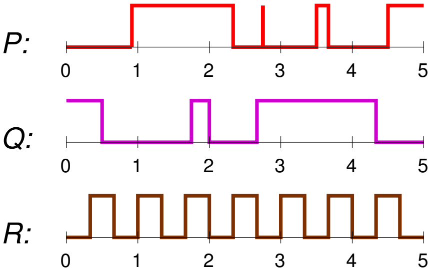
Disallow e.g. $\mathbb{Q}$:



Finite Variability

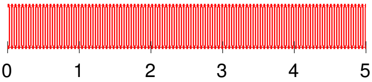
Timed behaviours are modelled as flows (or signals):

$f : [0, T) \to 2^{\mathbf{MP}}$



Predicates must have finite variability:

Disallow e.g. $\mathbb{Q}$:



Then:

Theorem (Rabinovich 2002)

MSO($<$) *satisfiability over finitely-variable flows is decidable.*

The Time-Bounded Theory of Verification

Theorem

For any bounded time domain $[0, T)$, satisfiability and model checking are decidable as follows:

MSO$(<, +1)$	*NON-ELEMENTARY*
FO$(<, +1)$	*NON-ELEMENTARY*
MTL	*EXPSPACE-complete*

The Time-Bounded Theory of Verification

Theorem

For any bounded time domain $[0, T)$, satisfiability and model checking are decidable as follows:

MSO$(<, +1)$	*NON-ELEMENTARY*
FO$(<, +1)$	*NON-ELEMENTARY*
MTL	*EXPSPACE-complete*

Theorem

MTL *and* FO$(<, +1)$ *are equally expressive over any fixed bounded time domain $[0, T)$.*

The Time-Bounded Theory of Verification

Theorem

For any bounded time domain $[0, T)$, satisfiability and model checking are decidable as follows:

MSO$(<, +1)$	*NON-ELEMENTARY*
FO$(<, +1)$	*NON-ELEMENTARY*
MTL	*EXPSPACE-complete*

Theorem

MTL *and* FO$(<, +1)$ *are equally expressive over any fixed bounded time domain* $[0, T)$*.*

Theorem

Given timed automata A, B, and time bound $T \in \mathbb{N}$, the time-bounded language inclusion problem $L_T(A) \subseteq L_T(B)$ is decidable and 2EXPSPACE-complete.

MSO($<$,+1) Time-Bounded Satisfiability

Key idea: eliminate the metric by ‘vertical stacking’.

MSO($<$,+1) Time-Bounded Satisfiability

Key idea: eliminate the metric by 'vertical stacking'.

- Let φ be an MSO($<$,$+1$) formula and let $T \in \mathbb{N}$.

MSO(<,+1) Time-Bounded Satisfiability

Key idea: eliminate the metric by 'vertical stacking'.

- Let φ be an MSO($<$,$+1$) formula and let $T \in \mathbb{N}$.
- Construct an MSO($<$) formula $\overline{\varphi}$ such that:

$$\varphi \text{ is satisfiable over } [0, T) \iff \overline{\varphi} \text{ is satisfiable over } [0, 1)$$

MSO($<$,+1) Time-Bounded Satisfiability

Key idea: eliminate the metric by 'vertical stacking'.

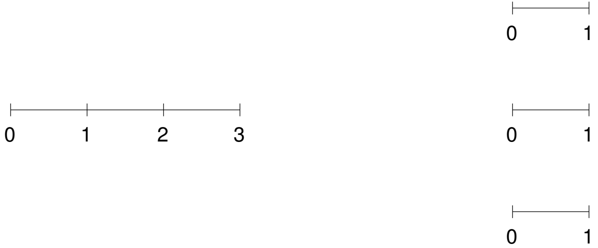
- Let φ be an MSO($<$,$+$1) formula and let $T \in \mathbb{N}$.
- Construct an MSO($<$) formula $\overline{\varphi}$ such that:

$$\varphi \text{ is satisfiable over } [0, T) \iff \overline{\varphi} \text{ is satisfiable over } [0, 1)$$

- Conclude by invoking decidability of MSO($<$).

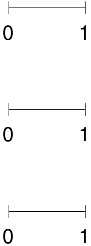
From MSO($<$,$+1$) to MSO($<$)

From MSO($<$,$+1$) to MSO($<$)

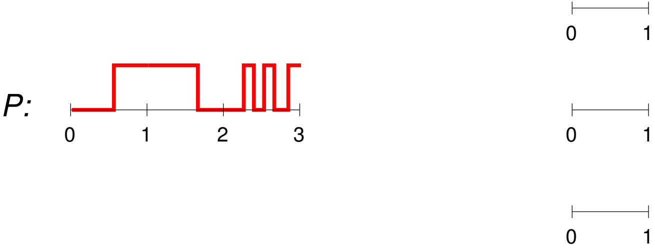


From MSO($<$,$+1$) to MSO($<$)

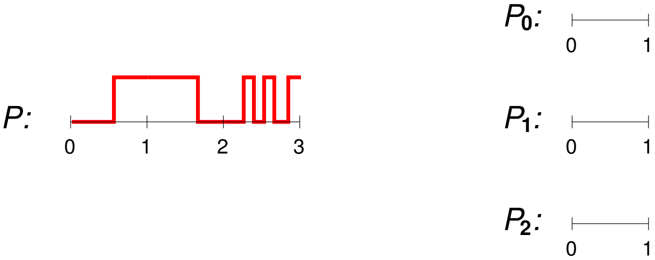
P:



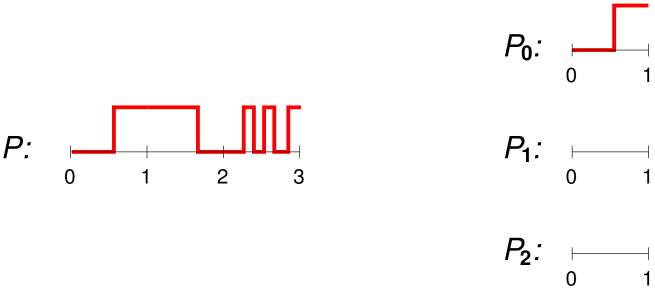
From MSO($<$,$+1$) to MSO($<$)



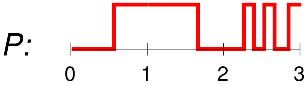
From MSO(<,+1) to MSO(<)

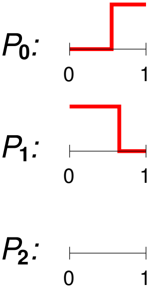


From MSO(<,+1) to MSO(<)

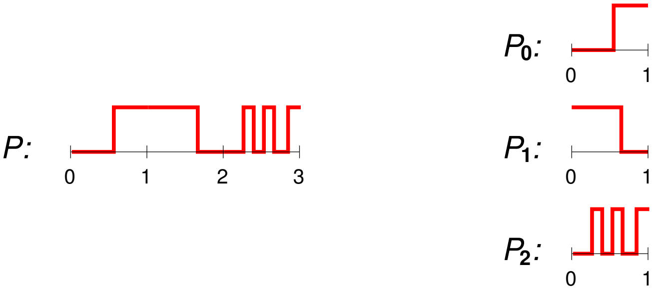


From MSO($<$,$+1$) to MSO($<$)

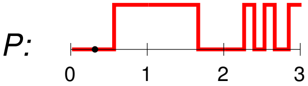


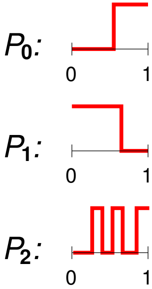


From MSO(<,+1) to MSO(<)

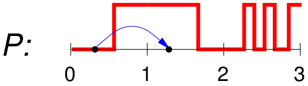


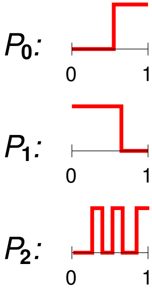
From MSO($<$,$+1$) to MSO($<$)



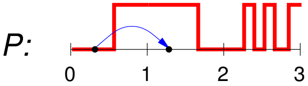


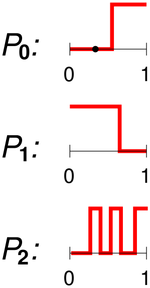
From MSO($<$,$+1$) to MSO($<$)



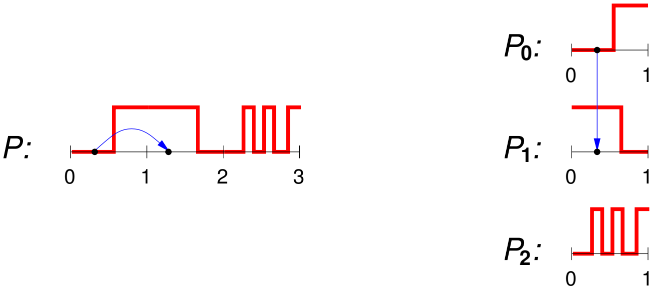


From MSO(<,+1) to MSO(<)

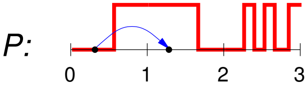


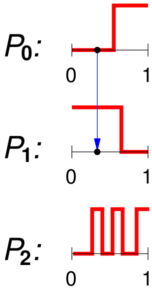


From MSO($<$,+1) to MSO($<$)



From MSO($<$,$+1$) to MSO($<$)

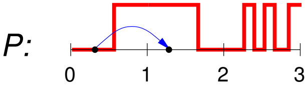


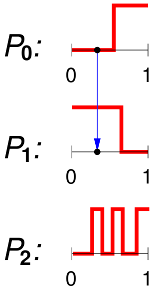


Replace every:

- $\forall x\, \psi(x)$

From MSO($<$,$+1$) to MSO($<$)

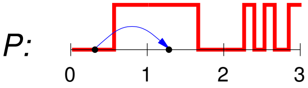


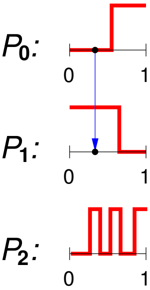


Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \land \psi(x+1) \land \psi(x+2))$

From MSO($<$,$+1$) to MSO($<$)

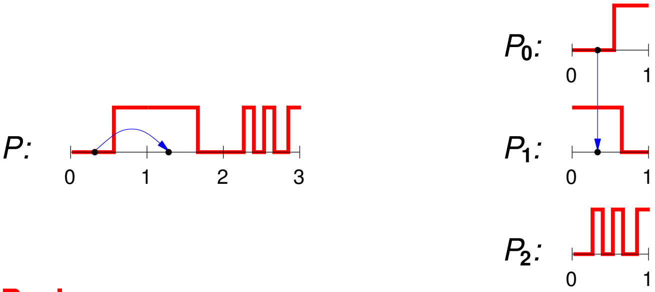




Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$

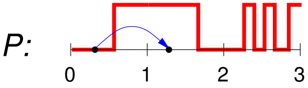
From MSO($<$,$+1$) to MSO($<$)

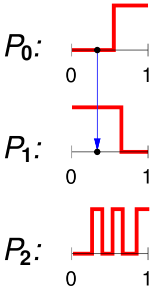


Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$ **by** $\begin{cases} x < y & \text{if } k_1 = k_2 \\ \textbf{true} & \text{if } k_1 < k_2 \\ \textbf{false} & \text{if } k_1 > k_2 \end{cases}$

From MSO(<,+1) to MSO(<)

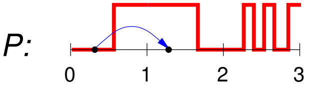


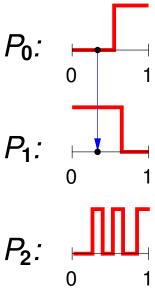


Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$ **by** $\begin{cases} x < y & \text{if } k_1 = k_2 \\ \textbf{true} & \text{if } k_1 < k_2 \\ \textbf{false} & \text{if } k_1 > k_2 \end{cases}$
- $P(x+k)$

From MSO($<$,$+1$) to MSO($<$)

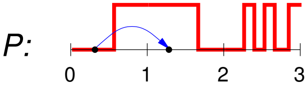


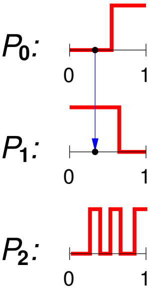


Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$ **by** $\begin{cases} x < y & \text{if } k_1 = k_2 \\ \textbf{true} & \text{if } k_1 < k_2 \\ \textbf{false} & \text{if } k_1 > k_2 \end{cases}$
- $P(x+k)$ **by** $P_k(x)$

From MSO(<,+1) to MSO(<)

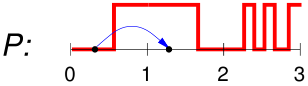


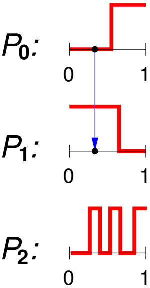


Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$ **by** $\begin{cases} x < y & \text{if } k_1 = k_2 \\ \textbf{true} & \text{if } k_1 < k_2 \\ \textbf{false} & \text{if } k_1 > k_2 \end{cases}$
- $P(x+k)$ **by** $P_k(x)$
- $\forall P\, \psi$

From MSO($<$,$+1$) to MSO($<$)

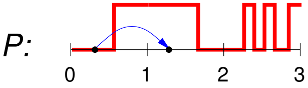


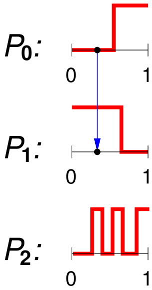


Replace every:

- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$ **by** $\begin{cases} x < y & \text{if } k_1 = k_2 \\ \textbf{true} & \text{if } k_1 < k_2 \\ \textbf{false} & \text{if } k_1 > k_2 \end{cases}$
- $P(x+k)$ **by** $P_k(x)$
- $\forall P\, \psi$ **by** $\forall P_0\, \forall P_1\, \forall P_2\, \psi$

From MSO(<,+1) to MSO(<)





Replace every:

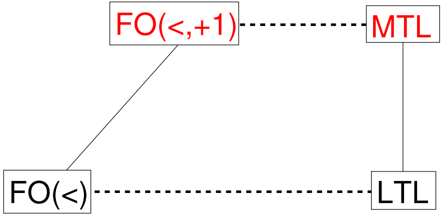
- $\forall x\, \psi(x)$ **by** $\forall x\ (\psi(x) \wedge \psi(x+1) \wedge \psi(x+2))$
- $x + k_1 < y + k_2$ **by** $\begin{cases} x < y & \text{if } k_1 = k_2 \\ \textbf{true} & \text{if } k_1 < k_2 \\ \textbf{false} & \text{if } k_1 > k_2 \end{cases}$
- $P(x+k)$ **by** $P_k(x)$
- $\forall P\, \psi$ **by** $\forall P_0\, \forall P_1\, \forall P_2\, \psi$

Then φ is satisfiable over $[0, T)$ $\iff$ $\overline{\varphi}$ is satisfiable over $[0, 1)$.

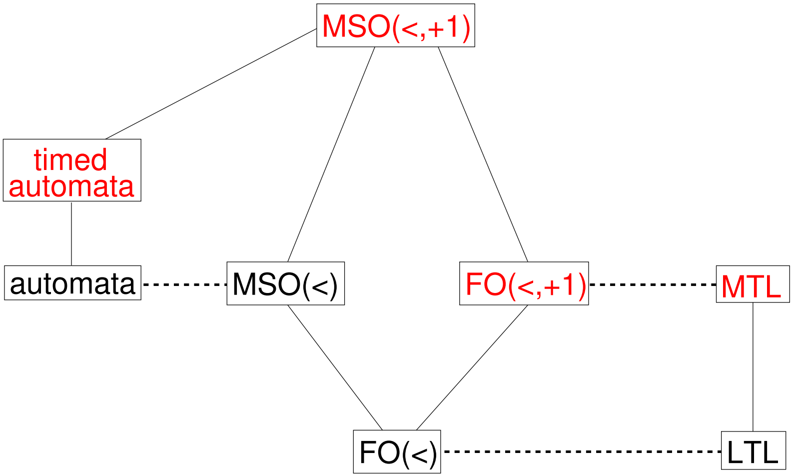
The Time-Bounded Theory: Expressiveness

$\boxed{\text{FO}(<)}$ - - - - - - - - - - $\boxed{\text{LTL}}$

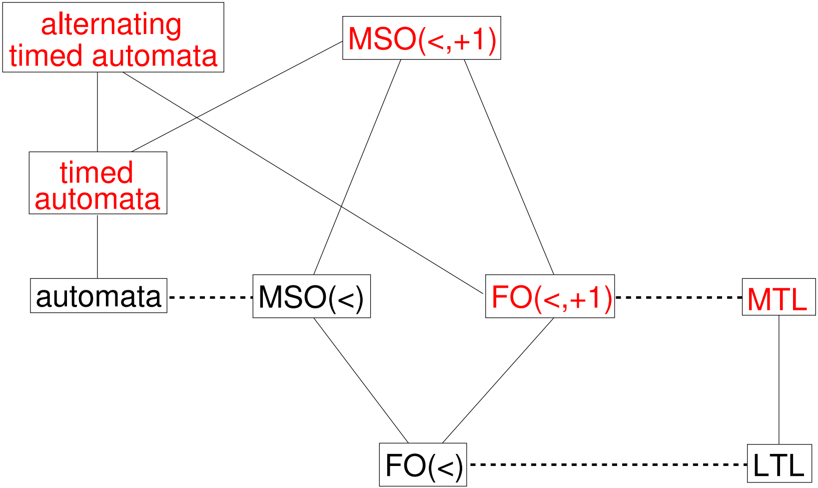
The Time-Bounded Theory: Expressiveness



The Time-Bounded Theory: Expressiveness



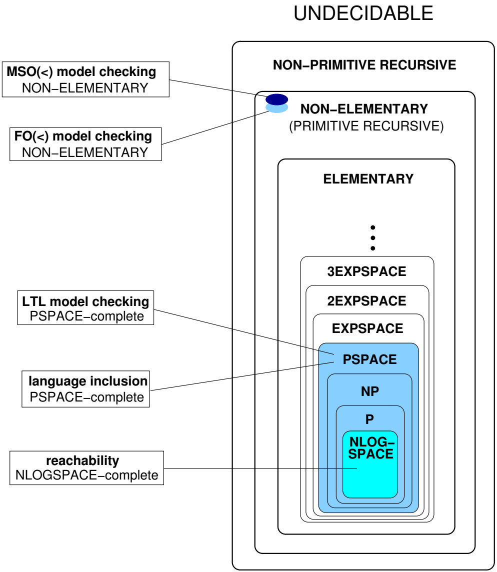
The Time-Bounded Theory: Expressiveness



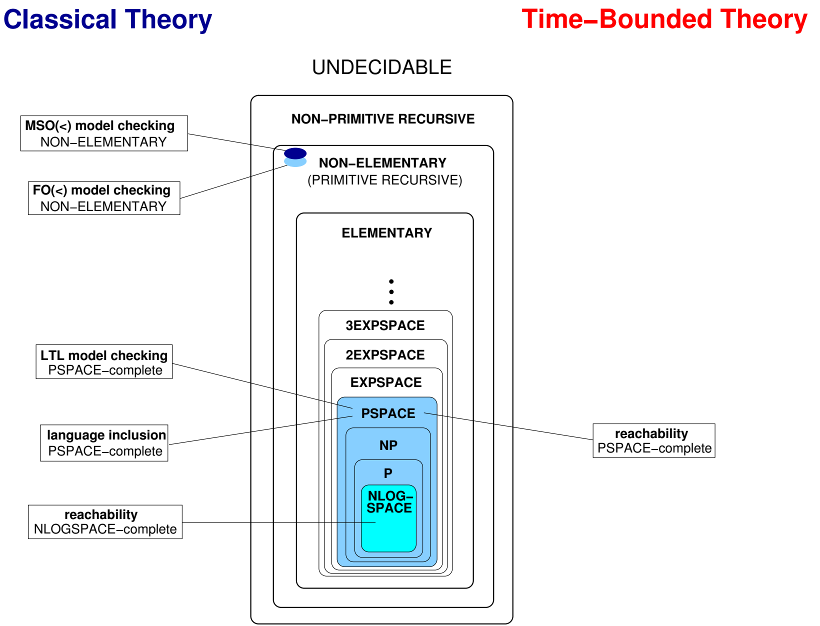
The Time-Bounded Theory: Complexity

Classical Theory

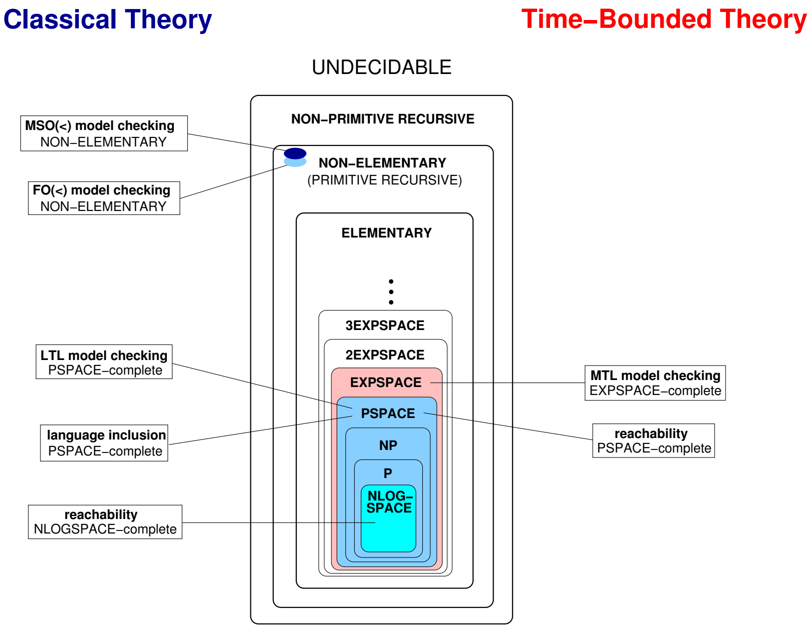
Time–Bounded Theory



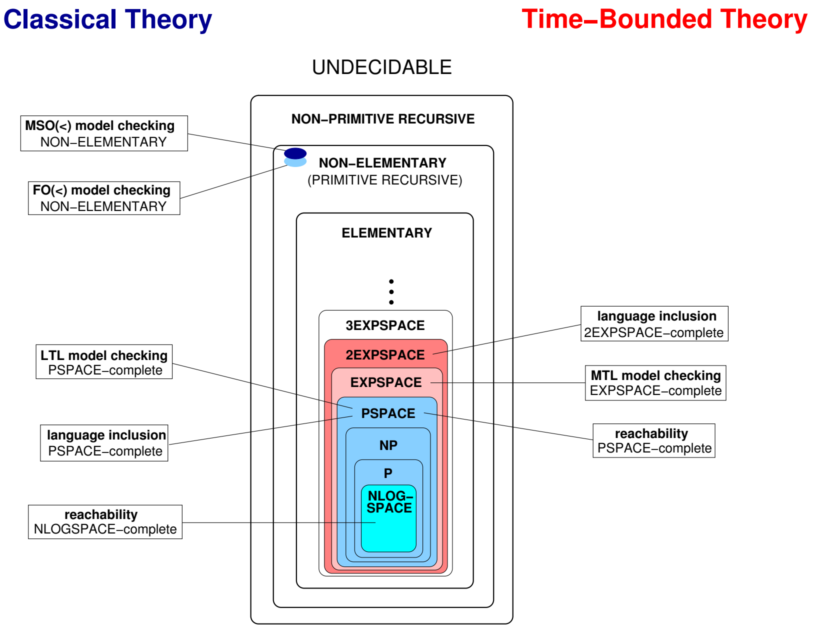
The Time-Bounded Theory: Complexity



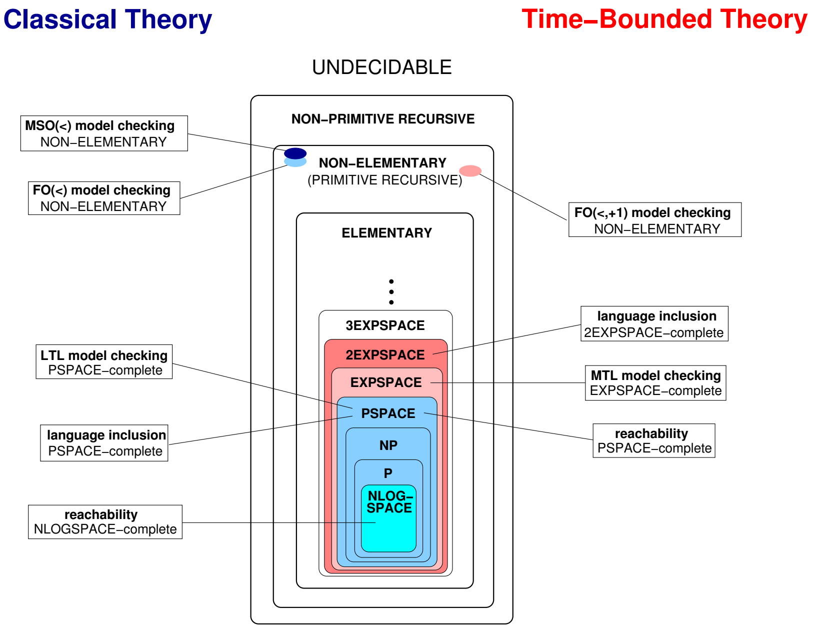
The Time-Bounded Theory: Complexity



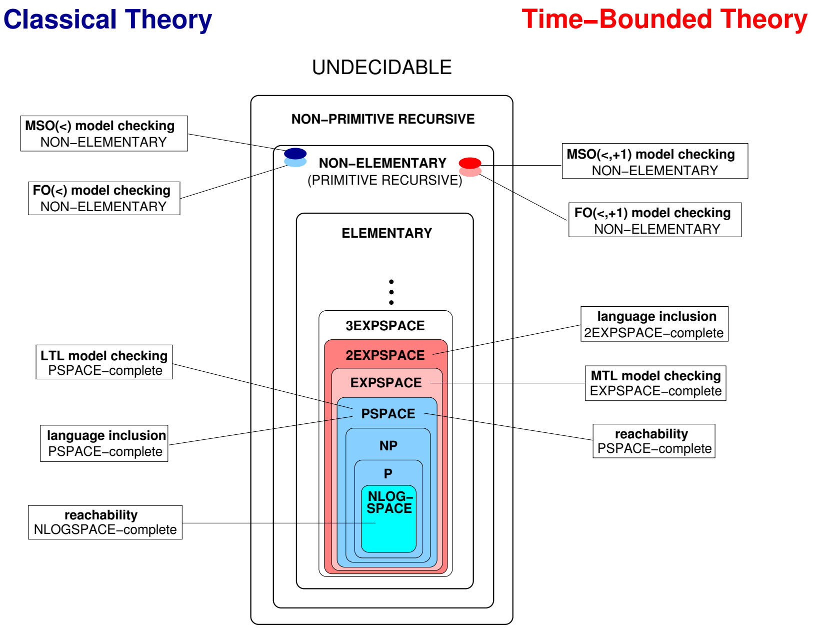
The Time-Bounded Theory: Complexity



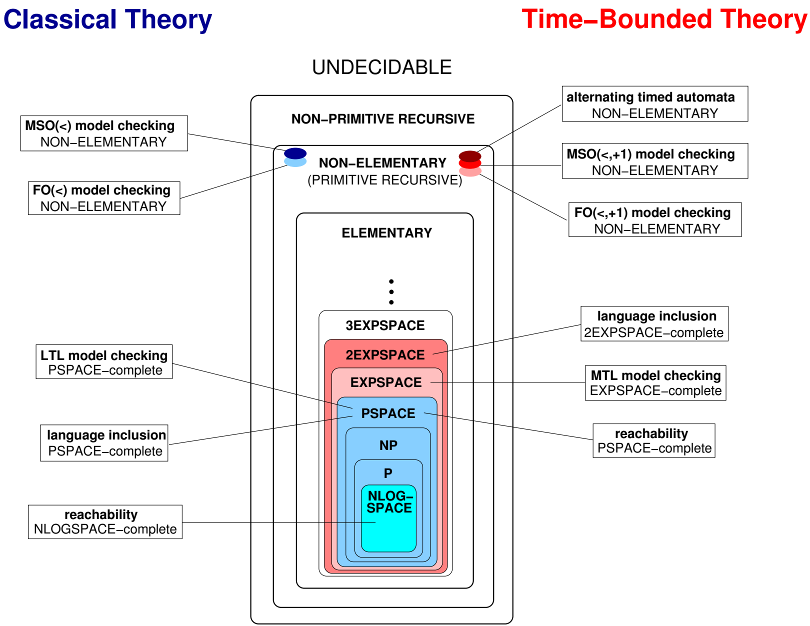
The Time-Bounded Theory: Complexity



The Time-Bounded Theory: Complexity



The Time-Bounded Theory: Complexity



Conclusion and Perspective

- For real-time systems, the time-bounded theory is much better behaved than the real-time theory.

Conclusion and Perspective

- For real-time systems, the time-bounded theory is much better behaved than the real-time theory.

Going forward:

- Extend the theory further!
  - Branching-time
  - Timed games and synthesis
  - Weighted and hybrid automata
  - ...
- Algorithmic and complexity issues
- Expressiveness issues
- Implementation and case studies