The Classical Linear Theory of Verification

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

\[ \square \Diamond P \quad \forall x \exists y (x < y \land P(y)) \]
Linear Temporal Logic (LTL)

\[ \theta ::= P \mid \theta_1 \land \theta_2 \mid \theta_1 \lor \theta_2 \mid \neg \theta \mid \Box \theta \mid \Diamond \theta \mid \square \theta \mid \theta_1 U \theta_2 \]

For example, \( \Box (REQ \rightarrow \Diamond ACK) \).
Specification and Verification

- **Linear Temporal Logic (LTL)**

\[ \theta ::= P \mid \theta_1 \land \theta_2 \mid \theta_1 \lor \theta_2 \mid \neg \theta \mid \Box \theta \mid \Diamond \theta \mid \square \theta \mid \theta_1 \cup \theta_2 \]

For example, \( \square (REQ \rightarrow \Diamond ACK) \).

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

\[ \varphi ::= x < y \mid P(x) \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \forall x \varphi \mid \exists x \varphi \]

For example, \( \forall x \ (REQ(x) \rightarrow \exists y \ (x < y \land ACK(y))) \).
Specification and Verification

▶ Linear Temporal Logic (LTL)

\[ \theta ::= P \mid \theta_1 \land \theta_2 \mid \theta_1 \lor \theta_2 \mid \neg \theta \mid \bigcirc \theta \mid \lozenge \theta \mid \square \theta \mid \theta_1 U \theta_2 \]

For example, \( \square (REQ \rightarrow \lozenge ACK) \).

▶ First-Order Logic (FO(<))

\[ \varphi ::= x < y \mid P(x) \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \forall x \varphi \mid \exists x \varphi \]

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

Verification is model checking: \( \text{IMP} \models \text{SPEC} \) ?
'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)’

\[ P \]

\[ \exists Q (Q \text{ holds precisely at even positions and } (Q \rightarrow P)) \]
‘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(<).
‘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’:

\[ Q \land \square(Q \rightarrow \Diamond \neg Q) \land \square(\neg Q \rightarrow \Diamond Q) \]
‘P holds at every even position (and may or may not hold at odd positions)’

\[
\begin{align*}
\square (Q & \rightarrow \bigcirc \neg Q) \land \square (\neg Q & \rightarrow \bigcirc Q) \\
\end{align*}
\]

- 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’:

\[
Q \land \square (Q \rightarrow \bigcirc \neg Q) \land \square (\neg Q \rightarrow \bigcirc Q)
\]

- 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 $\text{FO(<)}$.

LTL and $\text{FO(<)}$ can however capture the specification: ‘$Q$ holds precisely at even positions’:

\[
Q \land \Box (Q \rightarrow \Diamond \neg Q) \land \Box (\neg Q \rightarrow \Diamond Q)
\]

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

\[
Q \land \square(Q \rightarrow \diamond \neg Q) \land \square(\neg Q \rightarrow \diamond Q)
\]

So one way to capture the original specification would be to write: ‘Q holds precisely at even positions and \(\square(Q \rightarrow P)\)’.

Finally, need to existentially quantify \(Q\) out:
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’:

\[ Q \land \Box (Q \rightarrow \Diamond \neg Q) \land \Box (\neg Q \rightarrow \Diamond Q) \]

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:

\[ \exists Q \ (Q \ holds \ precisely \ at \ even \ positions \ and \ \Box (Q \rightarrow P)) \]
Monadic Second-Order Logic (MSO($<$))

\[ \varphi ::= x < y \mid P(x) \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \forall x \varphi \mid \exists x \varphi \mid \forall P \varphi \mid \exists P \varphi \]
Monadic Second-Order Logic (MSO(<))

\[ \varphi ::= x < y \mid P(x) \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \forall x \, \varphi \mid \exists x \, \varphi \mid \forall P \, \varphi \mid \exists P \, \varphi \]

Theorem (Büchi 1960)

Any MSO(<) formula \( \varphi \) can be effectively translated into an equivalent automaton \( A_\varphi \).
Monadic Second-Order Logic (MSO(<))

\[ \varphi ::= x < y \mid P(x) \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \neg \varphi \mid \forall x \varphi \mid \exists x \varphi \mid \forall P \varphi \mid \exists P \varphi \]

**Theorem (Büchi 1960)**

*Any MSO(<) formula* \( \varphi \) *can be effectively translated into an equivalent automaton* \( A_\varphi \).

**Corollary (Church 1960)**

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

\[ M \models \varphi \iff L(M) \cap L(A_{\neg \varphi}) = \emptyset \]
Complexity

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

... 3EXPSPACE 2EXPSPACE EXPSPACE PSPACE NP P NLOG–SPACE

NON–PRIMITIVE RECURSIVE:

Ackerman: 3, 4, 8, 2048, \(2^{2^{2^{\cdots^2}}}\), . . .
Complexity

- **NON-ELEMENTARY**: $2^{2^n}$
- **NON-PRIMITIVE RECURSIVE**:
  
  Ackerman: 3, 4, 8, 2048, $2^{2^{2^{2^{2^{2048}}}}}$, …

Diagram:

- UNDECIDABLE
- NON-PRIMITIVE RECURSIVE
  - NON-ELEMENTARY (PRIMITIVE RECURSIVE)
    - ELEMENTARY
      - P
      - NP
      - PSPACE
      - EXPSpace
      - 2EXPSPACE
      - 3EXPSPACE
    - NON-ELEMENTARY (PRIMITIVE RECURSIVE)
Complexity

- **UNDECIDABLE**

- **NON-PRIMITIVE RECURSIVE**
  - **NON-ELEMENTARY** (PRIMITIVE RECURSIVE)

- **ELEMENTARY**
  - ...

- **SPACE**
  - \( NLOG^{-P} \)
  - \( PSPACE \)
  - \( 2^{\text{EXPSPACE}} \)
  - \( 3^{\text{EXPSPACE}} \)
  - \( \text{EXPSPACE} \)
  - \( \text{NP} \)
  - \( P \)
  - \( \text{NLOG-SPACE} \)

- **NON-ELEMENTARY**: \( 2^{2^{\ldots^2}} \)

- **NON-PRIMITIVE RECURSIVE:**
  - Ackerman: 3, 4, 8, 2048, \( 2^{2^{\ldots^2}} \), \ldots
In fact:

**Theorem (Stockmeyer 1974)**

$\text{FO(<)}$ *satisfiability has non-elementary complexity.***
In fact:

**Theorem (Stockmeyer 1974)**

$\text{FO}(\prec)$ satisfaction has non-elementary complexity.

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

$LTL$ and $\text{FO}(\prec)$ have precisely the same expressive power.
Complexity and Equivalence

In fact:

Theorem (Stockmeyer 1974)
\( \text{FO}(\prec) \) satisfiability has non-elementary complexity.

Theorem (Kamp 1968; Gabbay, Pnueli, Shelah, Stavi 1980)
LTL and \( \text{FO}(\prec) \) have precisely the same expressive power.

But amazingly:

Theorem (Sistla & Clarke 1982)
LTL satisfiability and model checking are PSPACE-complete.
“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
“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

\[ \text{FO}(\langle) \quad \text{MSO}(\langle) \quad \text{LTL} \quad \text{ETLTL} \quad \text{counter-free automata} \]
The Classical Theory: Expressiveness

counter-free automata \rightarrow FO(\langle \rangle) \rightarrow LTL
The Classical Theory: Expressiveness

automata \text{MSO}(<) \text{FO}(<) \text{ETLTL}\text{counter-free automata} \text{LTL}
The Classical Theory: Expressiveness

- automata
  - counter-free automata
- MSO(<)
- FO(<)
- \( \mu TL \)
- ETL
- LTL
The Classical Theory: Complexity

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

\[ \ldots \]

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

NP

P

NLOG-SPACE
The Classical Theory: Complexity

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

P
PSPACE
EXPSPACE
3EXPSPACE
2EXPSPACE
EXPSPACE
PSPACE
NP
UNDECIDABLE

reachability
NLOGSPACE-complete

NLOGSPACE-complete
reachability
The Classical Theory: Complexity

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

3EXPSPACE
2EXPSPACE
EXPSPACE
PSPACE
NP
P
NLOGSPACE

LTL model checking
PSPACE–complete

language inclusion
PSPACE–complete

reachability
NLOGSPACE–complete
The Classical Theory: Complexity

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

3EXPSpace

2EXPSpace

EXPSPACE

PSPACE

NP

P

NLOGSPACE

reachability
NLOGSPACE–complete

language inclusion
PSPACE–complete

LTL model checking
PSPACE–complete

FO(<) model checking
NON–ELEMENTARY
The Classical Theory: Complexity

**UNDECIDABLE**

**NON-PRIMITIVE RECURSIVE**

- MSO(≤) model checking
  - NON-ELEMENTARY

- FO(≤) model checking
  - NON-ELEMENTARY

**NON-ELEMENTARY**

- (PRIMITIVE RECURSIVE)

**ELEMENTARY**

- LTL model checking
  - PSPACE-complete

- Language inclusion
  - PSPACE-complete

- Reachability
  - NLOGSPACE-complete

**SPACE**

- NLOGSPACE

- PSPACE

- EXPSPACE

- 2EXPSPACE

- 3EXPSPACE

- 2EXPSPACE

- EXPSPACE

- PSPACE

- NP

- P

- NLOGSPACE

- UNDECIDABLE

- 2EXPSPACE
“Lift the classical theory to the real-time world.”

Boris Trakhtenbrot, LICS 1995
Airbus A350 XWB
A350 XWB Fuel Management Sub-System

5.2 (1)

5.2.1

5.2.2

5.2.3

5.2.4

5.2.5

5.2.6

5.2.7

GROUND_OPS
entry:
GROUND_OPS_ACTIVE = TRUE;
evaluate_conditions();
during:
evaluate_conditions();
BMW Hydrogen 7
Timed Systems

Timed systems are everywhere... 

- Hardware circuits
- Communication protocols
- Cell phones
- Plant controllers
- Aircraft navigation systems
- Sensor networks
- ...
Timed Automata

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

Timed Automata

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


⇒ Led to inaugural CAV Award (2008) and inaugural Church Award (2016)!
Timed Automata

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

**Theorem (Alur, Courcoubetis, 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, Courcoubetis, 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 \( \text{PSPACE-complete} \).

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

Unfortunately:

Theorem (Alur & Dill 1990)

Language inclusion is undecidable for timed automata.
An Uncomplementable Timed Automaton

A cannot be complemented: There is no timed automaton $B$ with $L(B) = L(A)$. 

\[ A : \begin{array}{c}
    \node[state] (q0) at (0,0) {$a$}; \\
    \node[state] (q1) at (2,0) {$a$}; \\
    \node[state] (q2) at (4,0) {$a$}; \\
    \node[state] (q3) at (6,0) {$a$}; \\
    \node[state] (q4) at (8,0) {$a$}; \\
    \draw (q0) edge node {$x:=0$} (q1) \\
    \draw (q1) edge node {$x=1?$} (q2) \\
    \draw (q2) edge (q3) \\
    \draw (q3) edge (q4) \\
    \draw (q4) edge (q0) \\
\end{array} \]
An Uncomplementable Timed Automaton

\[ A : \]

\[
\begin{array}{c}
\text{a} \\
\text{a} \\
\text{a} \\
\end{array}
\]

\[
\begin{array}{c}
x:=0 \\
x:=1? \\
\end{array}
\]

\[ L(A) : \text{A cannot be complemented: There is no timed automaton } B \text{ with } L(B) = L(A). \]
An Uncomplementable Timed Automaton

\[ A : \]

\[ L(A) : \]

\( A \) cannot be complemented: There is no timed automaton \( B \) with \( L(B) = L(A) \).
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 (MTL) [Koymans; de Roever; Pnueli ∼1990] is a central quantitative specification formalism for timed systems.
Metric Temporal Logic

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

- MTL = LTL + timing constraints on operators:

\[ \square (\text{PEDAL} \rightarrow \diamond [5,10] \text{ BRAKE}) \]
Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli ~1990] is a central quantitative specification formalism for timed systems.

MTL = LTL + timing constraints on operators:

\(\Box(PEDAL \rightarrow \Diamond [5,10] BRAKE)\)

Widely cited and used (over 1600 papers according to Google Scholar!).
Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli ∼1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:

\[ \square(PEDAL \rightarrow \Diamond[5,10] BRAKE) \]

- 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 (MTL) [Koymans; de Roever; Pnueli ~1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:

\[ \Box (PEDAL \rightarrow \Diamond [5,10] BRAKE) \]

- 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].)*
The first-order metric logic of order (FO(<, +1)) extends FO(<) by the unary function ‘+1’.
The first-order metric logic of order (FO(<, +1)) extends FO(<) by the unary function ‘+1’.

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

$$\forall x \ (PEDAL(x) \rightarrow \exists y \ (x + 5 \leq y \leq x + 10 \land BRAKE(y)))$$
Theorem (Hirshfeld & Rabinovich 2007)

$\text{FO}(<, +1)$ is strictly more expressive than $\text{MTL}$ 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)

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

Theorem (Hunter, O., Worrell 2013)

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

Corollary: $\text{FO}(\prec, +1)$, $\text{FO}(\prec, +\mathbb{Q})$, $\text{MSO}(\prec, +1)$, $\text{MSO}(\prec, +\mathbb{Q})$ satisfiability and model checking are all undecidable over $\mathbb{R}_{\geq 0}$. 
The Real-Time Theory: Expressiveness

\[
\text{MTL} \quad \text{FO}(\lt, +1) \quad \text{MSO}(\lt, +1)
\]
The Real-Time Theory: Expressiveness

MSO($<$, +1)

Timed automata

FO($<$, +1)

MTL
The Real-Time Theory: Complexity

Classical Theory

Real-Time Theory

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY

(PRIMITIVE RECURSIVE)

ELEMENTARY

LTL model checking
PSPACE-complete

FO(<) model checking
NON-ELEMENTARY

MSO(<) model checking
NON-ELEMENTARY

reachability
NLOGSPACE-complete

NP

language inclusion
PSPACE-complete

P

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

NLOGSPACE
The Real-Time Theory: Complexity

Classical Theory

Real-Time Theory

UNDECIDABLE

NON−PRIMITIVE RECURSIVE

NON−ELEMENTARY

(PRIMITIVE RECURSIVE)

ELEMENTARY

EXPSPACE

PSPACE

2EXPSPACE

3EXPSPACE

NP

NON−PRIMITIVE RECURSIVE

MSO(<) model checking
NON−ELEMENTARY

FO(<) model checking
NON−ELEMENTARY

LTL model checking
PSPACE−complete

language inclusion
PSPACE−complete

reachability
NLOGSPACE−complete

1−clock reachability
NLOGSPACE−complete

language inclusion
PSPACE−complete

reachability
NLOGSPACE−complete

1−clock reachability
NLOGSPACE−complete
The Real-Time Theory: Complexity

Classical Theory

Real-Time Theory

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY

(PRIMITIVE RECURSIVE)

ELEMENTARY

NLOGSPACE-complete

language inclusion

PSPACE-complete

reachability

NLOGSPACE-complete

LTL model checking

PSPACE-complete

2-clock+ reachability

PSPACE-complete

1-clock reachability

NLOGSPACE-complete

MSO(<) model checking

NON-ELEMENTARY

FO(<) model checking

NON-ELEMENTARY

PSPACE-complete

reachability

NLOGSPACE-complete

2EXPSPACE

PSPACE

NP

NON-ELEMENTARY

PRIMITIVE RECURSIVE

 ELEMENTARY

...
The Real-Time Theory: Complexity

Classical Theory

- MSO(<) model checking
  - NON–ELEMENTARY

- FO(<) model checking
  - NON–ELEMENTARY

- LTL model checking
  - PSPACE–complete

- language inclusion
  - PSPACE–complete

- reachability
  - NLOGSPACE–complete

Real–Time Theory

- 1–clock language inclusion
  - NON–PRIMITIVE RECURSIVE

- 2–clock+ reachability
  - PSPACE–complete

- 1–clock reachability
  - NLOGSPACE–complete

- ELEMENTARY

- NON–ELEMENTARY (PRIMITIVE RECURSIVE)

- P

- EXPSPACE

- 3EXPSPACE

- NP

- NON–PRIMITIVE RECURSIVE

- UNDECIDABLE
The Real-Time Theory: Complexity

Classical Theory

- MSO(⟨) model checking
  - NON–ELEMENTARY

- FO(⟨) model checking
  - NON–ELEMENTARY

- LTL model checking
  - PSPACE–complete

- Language inclusion
  - PSPACE–complete

- Reachability
  - NLOGSPACE–complete

Real–Time Theory

- 1–clock language inclusion
  - NON–PRIMITIVE RECURSIVE

- 2–clock+ language inclusion
  - UNDECIDABLE

- 1–clock reachability
  - NLOGSPACE–complete

- 2–clock+ reachability
  - PSPACE–complete

- EXPSPACE

- PSPACE

- NP

- ELEMENTARY

- NON–ELEMENTARY (PRIMITIVE RECURSIVE)

- NON–ELEMENTARY

- SPACE

- NLOG–P

- PSPACE

- 3EXPSPACE

- EXPSPACE

- P

- NLOGSPACE

- MSO(<) model checking
  - NON–ELEMENTARY

- NON–PRIMITIVE RECURSIVE

- UNDECIDABLE
The Real-Time Theory: Complexity

Classical Theory

Real-Time Theory

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY

PRIMITIVE RECURSIVE

ELEMENTARY

EXPSPACE

3EXPSPACE

2EXPSPACE

PSPACE

NP

MSO(\prec) model checking
NON-ELEMENTARY

FO(\prec) model checking
NON-ELEMENTARY

LTL model checking
PSPACE-complete

language inclusion
PSPACE-complete

reachability
NLOGSPACE-complete

1-clock reachability
NLOGSPACE-complete

2-clock+ reachability
PSPACE-complete

MTL model checking
NON-PRIMITIVE RECURSIVE/
UNDECIDABLE

1-clock language inclusion
NON-PRIMITIVE RECURSIVE

2-clock+ language inclusion
NON-PRIMITIVE RECURSIVE

UNDECIDABLE
The Real-Time Theory: Complexity

Classical Theory

- **LTL model checking**: PSPACE-complete
- **MSO(<) model checking**: NON-ELEMENTARY
- **FO(<) model checking**: NON-ELEMENTARY
- **Language inclusion**: PSPACE-complete
- **Reachability**: NLOGSPACE-complete

Real-Time Theory

- **2-clock+ language inclusion**: UNDECIDABLE
- **MTL model checking**: NON-PRIMITIVE RECURSIVE/UNDECIDABLE
- **1-clock language inclusion**: NON-PRIMITIVE RECURSIVE
- **2-clock+ reachability**: PSPACE-complete
- **1-clock reachability**: NLOGSPACE-complete

**Complexity Classes**:
- ELEMENTARY
- NON-ELEMENTARY (PRIMITIVE RECURSIVE)
- NON-PRIMITIVE RECURSIVE
- SPACE:
  - NLOG−P
  - PSPACE
  - 3EXPSPACE
  - EXPSPACE
  - NP
- 1-clock language inclusion: UNDECIDABLE
- 2-clock+ language inclusion: UNDECIDABLE
- MSO(<) model checking: NON-ELEMENTARY
- FO(<,+1) model checking: UNDECIDABLE
The Real-Time Theory: Complexity

Classical Theory

- LTL model checking
  - PSPACE-complete
- MSO(<) model checking
  - NON–ELEMENTARY
- FO(<) model checking
  - NON–ELEMENTARY

Real-Time Theory

- MTL model checking
  - NON–PRIMITIVE RECURSIVE/
    - UNDECIDABLE
- MSO(<,+1) model checking
  - UNDECIDABLE
- FO(<,+1) model checking
  - UNDECIDABLE
- 2–clock+ language inclusion
  - UNDECIDABLE
- 1–clock language inclusion
  - NON–PRIMITIVE RECURSIVE
- 2–clock+ reachability
  - PSPACE–complete
- 1–clock reachability
  - NLOGSPACE–complete

Complexity Classes:
- ELEMENTARY
- NON–ELEMENTARY
- SPACE
  - NLOG–P
  - PSPACE
  - 3EXPSPACE
  - EXPSPACE
  - NP
  - NON–PRIMITIVE RECURSIVE
  - 2EXPSPACE
  - 3EXPSPACE

- MSO(<) model checking
  - NON–ELEMENTARY
- FO(<) model checking
  - NON–ELEMENTARY
- LTL model checking
  - PSPACE–complete
- language inclusion
  - PSPACE–complete
- reachability
  - NLOGSPACE–complete
- 1–clock reachability
  - NLOGSPACE–complete
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($\langle, +1\rangle$)
- 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):

\[ f : [0, T) \rightarrow 2^{MP} \]
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

\( Q: \)

\( R: \)
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

\[ P: \]
\[ 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

\[ Q: \]
\[ 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

\[ R: \]
\[ 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

Predicates must have finite variability: Disallow e.g.

Then:
Theorem (Rabinovich 2002)
\[ \text{MSO}(<) \text{satisfiability over finitely-variable flows is decidable.} \]
Finite Variability

Timed behaviours are modelled as flows (or signals):

\[ f : [0, T) \to 2^{\mathbb{M}} \]

Predicates must have finite variability: Disallow e.g.

Then:

Theorem (Rabinovich 2002)

\( \text{MSO}(<) \) satisfiability over finitely-variable flows is decidable.
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

Predicates must have finite variability:
Finite Variability

Timed behaviours are modelled as flows (or signals):

\[ f : [0, T) \rightarrow 2^{	ext{MP}} \]

Predicates must have finite variability:

Disallow e.g. \( Q \):

Disallow e.g. \( Q \):
Finite Variability

Timed behaviours are modelled as flows (or signals):

\[ f : [0, T) \rightarrow 2^{\mathbb{MP}} \]

Predicates must have finite variability:

Disallow e.g. \( Q \):

Then:

**Theorem (Rabinovich 2002)**

\( \text{MSO}(\lt) \) 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:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSO((&lt;, +1))</td>
<td>NON-ELEMENTARY</td>
</tr>
<tr>
<td>FO((&lt;, +1))</td>
<td>NON-ELEMENTARY</td>
</tr>
<tr>
<td>MTL</td>
<td>EXPSPACE-complete</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>MSO((&lt;, +1))</th>
<th>NON-ELEMENTARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO((&lt;, +1))</td>
<td>NON-ELEMENTARY</td>
</tr>
<tr>
<td>MTL</td>
<td>EXPSPACE-complete</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>MSO((&lt;, +1))</th>
<th>NON-ELEMENTARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO((&lt;, +1))</td>
<td>NON-ELEMENTARY</td>
</tr>
<tr>
<td>MTL</td>
<td>EXPSPACE-complete</td>
</tr>
</tbody>
</table>

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

Let $\varphi$ be an MSO($\prec, +1$) formula and let $T \in \mathbb{N}$.
MSO($\langle, +1\rangle$) Time-Bounded Satisfiability

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

- Let $\varphi$ be an MSO($\langle, +1\rangle$) formula and let $T \in \mathbb{N}$.
- Construct an MSO($\langle\rangle$) 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 \( \varphi \) 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($\langle, +1 \rangle$) to MSO($\langle \rangle$)
From MSO($\langle, +1 \rangle$) to MSO($\langle \rangle$)

$P$: 0 1 2 3
From MSO$(\prec, +1)$ to MSO$(\prec)$
From MSO($\prec, +1$) to MSO($\prec$)

\[ P: \]

\[
P_0: 0 \quad 1
\]

\[
P_1: 0 \quad 1
\]

\[
P_2: 0 \quad 1
\]
From MSO(\(<, +1\)) to MSO(\(<\))

\[
\begin{align*}
P: & \quad & P_0: & \quad & P_1: & \quad & P_2: \\
0 & \quad 1 & \quad 2 & \quad 3 & \quad 0 & \quad 1 & \quad 0 & \quad 1
\end{align*}
\]

Replace every:

\[
\begin{align*}
\forall x \psi(x) & \text{ by } \forall x (\psi(x) \land \psi(x+1) \land \psi(x+2)) \\
x + k_1 < y + k_2 & \text{ by } \begin{cases} x < y & \text{if } k_1 = k_2 \\
true & \text{if } k_1 < k_2 \\
false & \text{if } k_1 > k_2
\end{cases}
\end{align*}
\]

\[
\begin{align*}
P(x+k) & \text{ by } P_k(x) \\
\forall P \psi & \text{ by } \forall P \psi_0 \forall P \psi_1 \forall P \psi_2
\end{align*}
\]

Then \(\phi\) is satisfiable over \([0, T)\) \iff \(\phi\) is satisfiable over \([0, 1)\).
From MSO($\langle, +1 \rangle$) to MSO($\langle \rangle$)

$P$:

\[ P_0: \]

\[ P_1: \]

\[ P_2: \]
From MSO($\prec, +1$) to MSO($\prec$)

$P$: 

$P_0$: 

$P_1$: 

$P_2$: 

Replace every:

$\forall x \psi(x)$

by

$\forall x (\psi(x) \land \psi(x + 1) \land \psi(x + 2))$

$\forall x + k < y + k$ by

$$
\begin{cases}
    x < y & \text{if } k_1 = k_2 \\
    \text{true} & \text{if } k_1 < k_2 \\
    \text{false} & \text{if } k_1 > k_2
\end{cases}
$$

$P(x + k)$ by $P_k(x)$

$\forall P \psi$ by $\forall P_0 \forall P_1 \forall P_2 \psi$

Then $\phi$ is satisfiable over $[0, T)$ $\iff$ $\phi$ is satisfiable over $[0, 1)$. 

From MSO($\langle, +1\rangle$) to MSO($\langle\rangle$)

$P$: 

$P_0$: 

$P_1$: 

$P_2$: 

Replace every:

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

$x+k_1 < y+k_2$ by

$$
\begin{cases}
  x < y & \text{if } k_1 = k_2 \\
  \text{true} & \text{if } k_1 < k_2 \\
  \text{false} & \text{if } k_1 > k_2
\end{cases}
$$

$P(x+k)$ by $P_{k}(x)$

$\forall P \psi$ by $\forall P_0 \forall P_1 \forall P_2 \psi$

Then $\phi$ is satisfiable over $[0, T)$ $\iff$ $\phi$ is satisfiable over $[0, 1)$.
From MSO(\(<, +1\)) to MSO(\(<\))

\[
\forall x \psi(x) \rightarrow \forall x (\psi(x) \land \psi(x + 1) \land \psi(x + 2))
\]

\[
x + k_1 < y + k_2 \rightarrow \begin{cases} 
  x < y & \text{if } k_1 = k_2 \\
  \text{true} & \text{if } k_1 < k_2 \\
  \text{false} & \text{if } k_1 > k_2
\end{cases}
\]

\[
\forall P \psi \rightarrow \forall P_0 \forall P_1 \forall P_2 \psi
\]

Then \(\phi\) is satisfiable over \([0, T)\) \(\iff\) \(\phi\) is satisfiable over \([0, 1)\).
From MSO($\langle, +1 \rangle$) to MSO($\langle \rangle$)

\[
\forall x \psi(x) \leftrightarrow \forall x \left( \psi(x) \land \psi(x + 1) \land \psi(x + 2) \right)
\]

\[
x + k_1 < y + k_2 \leftrightarrow \begin{cases} 
  x < y & \text{if } k_1 = k_2 \\
  \text{true} & \text{if } k_1 < k_2 \\
  \text{false} & \text{if } k_1 > k_2 
\end{cases}
\]

\[
\forall P \psi(x + k) \leftrightarrow P_k(x)
\]

\[
\forall P \psi \leftrightarrow \forall P_0 \forall P_1 \forall P_2 \psi
\]

Then $\phi$ is satisfiable over $[0, T)$ $\iff$ $\phi$ is satisfiable over $[0, 1)$.
From MSO($\prec, +1$) to MSO($\prec$)

\[ P: \]

\[ P_0: \]

\[ P_1: \]

\[ P_2: \]
From MSO($\langle\text{,<}+1\rangle$) to MSO($\langle\text{<}\rangle$)

Replace every:

- $\forall x \, \psi(x)$
From MSO($\prec, +1$) to MSO($\prec$)

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 $\forall x \, (\psi(x) \land \psi(x + 1) \land \psi(x + 2))$
- $x + k_1 < y + k_2$
From MSO(\(\langle, +1 \rangle\)) to MSO(\(\langle \rangle\))

Replace every:

\[ \forall x \, \psi(x) \quad \text{by} \quad \forall x \, (\psi(x) \land \psi(x + 1) \land \psi(x + 2)) \]

\[ x + k_1 < y + k_2 \quad \text{by} \quad \begin{cases} 
  x < y & \text{if } k_1 = k_2 \\
  \text{true} & \text{if } k_1 < k_2 \\
  \text{false} & \text{if } k_1 > k_2 
\end{cases} \]
From MSO($<$, $+$1) to MSO($<$)

Replace every:

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

Replace every:

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

Replace every:

- \(\forall x \, \psi(x)\) by \(\forall x \, (\psi(x) \land \psi(x + 1) \land \psi(x + 2))\)
- \(x + k_1 < y + k_2\) by \(x < y\) if \(k_1 = k_2\), \(\text{true}\) if \(k_1 < k_2\), \(\text{false}\) if \(k_1 > k_2\)
- \(P(x + k)\) by \(P_k(x)\)
- \(\forall P \, \psi\)
From MSO($\prec, +1$) to MSO($\prec$)

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

Then \(\varphi\) is satisfiable over \([0, T)\) \(\iff\) \(\overline{\varphi}\) is satisfiable over \([0, 1)\).
The Time-Bounded Theory: Expressiveness

FO(\textless) \quad \text{alternating} \quad \text{timed automata} \quad \text{LTL}
The Time-Bounded Theory: Expressiveness

- FO(<)
- FO(<,+1)
- MTL
- FO(<=)
- LTL
The Time-Bounded Theory: Expressiveness

- Timed automata
- MSO($\langle, +1\rangle$)
- FO($\langle\rangle$)
- FO($\langle, +1\rangle$)
- MTL
- LTL
The Time-Bounded Theory: Expressiveness

- alternating timed automata
- timed automata
- automata
- MSO(≤)
- FO(≤)
- LTL
- MSO(≤,+1)
- FO(≤,+1)
- MTL
- LTL
The Time-Bounded Theory: Complexity

Classical Theory

- MSO(<) model checking
  - NON–ELEMENTARY
- FO(<) model checking
  - NON–ELEMENTARY
- LTL model checking
  - PSPACE–complete
- Language inclusion
  - PSPACE–complete
- Reachability
  - NLOGSPACE–complete

Time–Bounded Theory

- NON–ELEMENTARY
  - (PRIMITIVE RECURSIVE)
- ELEMENTARY
  -...

- EXPSPACE
  - PSPACE
  - NP
  - P
  - NLOGSPACE

- 2EXPSPACE
  - 3EXPSPACE
  - UNDECIDABLE
The Time-Bounded Theory: Complexity

Classical Theory

Time-Bounded Theory

- MSO(<) model checking: NON-ELEMENTARY
- FO(<) model checking: NON-ELEMENTARY
- LTL model checking: PSPACE-complete
- Language inclusion: PSPACE-complete
- Reachability: NLOGSPACE-complete
- Reachability: PSPACE-complete

Complexity Classes:
- ELEMENTARY
- NON-ELEMENTARY
- NON-PRIMITIVE RECURSIVE
- UNDECIDABLE
- EXPSPACE
- 2EXPSPACE
- 3EXPSPACE
- PSPACE
- NP
- P
- NLOGSPACE

Classical Theory

Time-Bounded Theory
The Time-Bounded Theory: Complexity

Classical Theory

- MSO(<) model checking
  - NON–ELEMENTARY

- FO(<) model checking
  - NON–ELEMENTARY

- LTL model checking
  - PSPACE–complete

- Language inclusion
  - PSPACE–complete

- Reachability
  - NLOGSPACE–complete

Time-Bounded Theory

- MTL model checking
  - EXPSPACE–complete

- Reachability
  - PSPACE–complete
The Time-Bounded Theory: Complexity

Classical Theory

Time-Bounded Theory

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY

(PRIMITIVE RECURSIVE)

ELEMENTARY

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

NP

P

NLOGSPACE

language inclusion

2EXPSPACE–complete

MTL model checking

EXPSPACE–complete

reachability

PSPACE–complete

reachability

NLOGSPACE–complete

language inclusion

PSPACE–complete

LTL model checking

PSPACE–complete

MSO(<) model checking

NON–ELEMNETARY

FO(<) model checking

NON–ELEMNETARY

Non–Primitve Recursive

Non–Elementary

Elementary

Classical Theory

Time–Bounded Theory
The Time-Bounded Theory: Complexity

Classical Theory

Non-Primitive Recursive

- MSO(\textless) model checking
  - Non-ELEMENTARY

- FO(\textless) model checking
  - Non-ELEMENTARY

- LTL model checking
  - PSPACE-complete

- Language inclusion
  - PSPACE-complete

- Reachability
  - NLOGSPACE-complete

Primitive Recursive

- NON-ELEMENTARY
  - (PRIMITIVE RECURSIVE)

- ELEMENTARY

- EXPSPACE

- NLOGSPACE

- PSPACE

- P

- NP

Time-Bounded Theory

UNDECIDABLE

- 2EXPSPACE

- 3EXPSPACE

- EXPSPACE

- 2EXPSPACE-complete

- EXPSPACE-complete

- PSPACE-complete

- Language inclusion
  - 2EXPSPACE-complete

- MTL model checking
  - EXPSPACE-complete

- Reachability
  - PSPACE-complete
The Time-Bounded Theory: Complexity

Classical Theory

Time-Bounded Theory

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY

(PRIMITIVE RECURSIVE)

ELEMENTARY

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

NP

P

NLOGSPACE

reachability

NLOGSPACE-complete

language inclusion

PSPACE-complete

MTL model checking

EXPSPACE-complete

reachability

PSPACE-complete

FO(<) model checking

NON-ELEMENTARY

FO(<,+1) model checking

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

MSO(<,+1) model checking

NON-ELEMENTARY

language inclusion

2EXPSPACE-complete

reachability

PSPACE-complete

LTL model checking

PSPACE-complete

NP

P

NLOGSPACE

reachability

NLOGSPACE-complete

reachability

PSPACE-complete

language inclusion

PSPACE-complete

MTL model checking

EXPSPACE-complete

reachability

PSPACE-complete

LTL model checking

PSPACE-complete

NP

P

NLOGSPACE

reachability

NLOGSPACE-complete

reachability

PSPACE-complete
The Time-Bounded Theory: Complexity

Classical Theory

Time-Bounded Theory

- NLOGSPACE-complete
- reachability
- PSPACE-complete
- language inclusion
- MTL model checking
- 2EXPSPACE-complete
- EXPSPACE
- NP
- P
- NLOGSPACE

- MSO(<) model checking
- NON-ELEMENTARY
- non-primitive recursive
- non-primitive recursive
- UNDECIDABLE

- alternating timed automata
- NON-ELEMENTARY

- MSO(<,+1) model checking
- NON-ELEMENTARY

- FO(<) model checking
- NON-ELEMENTARY

- FO(<,+1) model checking
- NON-ELEMENTARY

- LTL model checking
- PSPACE-complete

- language inclusion
- 2EXPSPACE-complete

- MTL model checking
- EXPSPACE-complete

- reachability
- PSPACE-complete

- reachability
- NLOGSPACE-complete

- ELEMENTARY

- 3EXPSPACE
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