A Survey of Classical, Real-Time, and Time-Bounded Verification

Joël Ouaknine

Department of Computer Science
Oxford University

Quantitative Model Checking Winter School, February 2012
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).
The Classical Theory of Verification

- Qualitative (order-theoretic), rather than quantitative (metric).
The Classical Theory of Verification

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

‘P occurs infinitely often’
A Simple Example

‘P occurs infinitely often’
A Simple Example

‘$P$ occurs infinitely often’

$\square \lozenge P$
A Simple Example

‘P occurs infinitely often’

∀x ∃y (x < y ∧ P(y))
Specification and Verification

- Linear Temporal Logic (LTL)

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

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

- **Linear Temporal Logic (LTL)**

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

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

Verification is model checking: \( \text{IMP} \models \text{SPEC} \)?
Another Example

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

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

\[
P \land (P \rightarrow \Box \neg P) \land (\neg P \rightarrow \Box P)
\]

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

‘Q holds precisely at even positions and (Q \rightarrow P)’

Finally, need to existentially quantify Q out:

\[\exists Q (Q \text{ holds precisely at even positions and } (Q \rightarrow P))\]
Another Example

‘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(<).

\[ \forall \exists \forall x \exists y (x < y) \]
Another Example

‘\(P\) 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’:

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

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

\[
\exists Q \left( Q \text{ holds precisely at even positions and } \left( Q \rightarrow P \right) \right)
\]
Another Example

‘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’:

\[ Q \land \Box (Q \rightarrow \bigcirc \neg Q) \land \Box (\neg Q \rightarrow \bigcirc Q) \]
Another Example

‘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’:

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

▶ So one way to capture the original specification would be to write:
Another Example

‘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’:

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

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

‘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’:

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

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:
Another Example

‘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’:

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

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:

\[ \exists Q (Q \text{ holds precisely at even positions and } \Box (Q \rightarrow P)) \]
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(<))

ϕ ::= x < y | P(x) | ϕ₁ ∧ ϕ₂ | ϕ₁ ∨ ϕ₂ | ¬ϕ | ∀x ϕ | ∃x ϕ | ∀P ϕ | ∃P ϕ

Theorem (Büchi 1960)

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

Corollary (Church 1960)

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

\[ M \models ϕ \text{ iff } L(M) \cap L(A_{¬ϕ}) = \emptyset \]
Complexity

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY
(PRIMITVE RECURSIVE)

ELEMENTARY

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

NP

P

NLOG–SPACE
Complexity

- ELEMENTARY
  - (PRIMITIVE RECURSIVE)

- NON-ELEMENTARY
  - SPACE
    - NLOG-P
    - PSPACE
    - 2EXPSPACE
    - 3EXPSPACE
    - EXPSPACE
    - NP

- NON-PRIMITIVE RECURSIVE
  - UNDECIDABLE

- NON-ELEMENTARY: $2^{2^n}$

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

Complexity

- **UNDECIDABLE**
- **NON-PRIMITIVE RECURSIVE**
  - **NON-ELEMENTARY** (PRIMITIVE RECURSIVE)
  - **ELEMENTARY**
    - 3EXPSPACE
    - 2EXPSPACE
    - EXPSPACE
    - PSPACE
    - NP
    - P
    - NLOGSPACE

- NON-ELEMENTARY: \(2^{2^n}\)
- NON-PRIMITIVE RECURSIVE:
  - Ackerman: 3, 4, 8, 2048, \(2^{2^{2^{2^{2^{2^{2048}}}}}}\), ...
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.
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

\[
\text{FO}(<) \quad \text{MSO}(<) \quad \text{LTL} \quad \text{ETLTLm} \quad \text{automata}
\]
The Classical Theory: Expressiveness

counter-free automata

$\text{FO}(\lt)$

$LTL$

The Classical Theory: Expressiveness

automata \rightarrow MSO(\langle \rangle)

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

automata  -- MSO(<) -- μTL  -- ETL

counter-free automata  -- FO(<) -- LTL
The Classical Theory: Complexity

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY (PRIMITIVE RECURSIVE)

ELEMENTARY

P

NP

NLOG–SPACE

PSPACE

EXPSPACE

2EXPSPACE

3EXPSPACE

\ldots

\ldots

\ldots
The Classical Theory: Complexity

UNDECIDABLE

NON-PRIMITIVE RECURSIVE

NON-ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

...
The Classical Theory: Complexity

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

3EXPSPACE
2EXPSPACE
EXPSPACE
PSPACE
NP
P
NLOGSPACE

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

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**
          - **NLOGSPACE**-complete
          - **PSPACE**-complete
          - **LTL model checking**
          - **PSPACE**-complete
          - **language inclusion**
          - **PSPACE**-complete
          - **reachability**
          - **NLOGSPACE**-complete
    - **NON-ELEMENTARY**
      - **PRIMITIVE RECURSIVE**
        - **ELEMENTARY**
          - **3EXPSPACE**
          - **2EXPSPACE**
          - **EXPSPACE**
            - **PSPACE**
              - **NP**
              - **P**
                - **NLOGSPACE**
              - **NLOGSPACE**-complete
              - **PSPACE**-complete
              - **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

PSPACE

EXPSPACE

2EXPSPACE

3EXPSPACE

PSPACE-complete

NP

P

NLOGSPACE-complete

reachability

language inclusion

LTL model checking

PSPACE-complete

FO(<) model checking

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

NLOGSPACE-complete

reachability

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

FO(<) model checking

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

FO(<) model checking

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

FO(<) model checking

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

FO(<) model checking

NON-ELEMENTARY

MSO(<) model checking

NON-ELEMENTARY

FO(<) model checking

NON-ELEMENTARY
“Lift the classical theory to the real-time world.”

Boris Trakhtenbrot, LICS 1995
Airbus A350 XWB
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
- ...
Timed automata were introduced by Rajeev Alur at Stanford during his PhD thesis under David Dill:

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.

Unfortunately:
Theorem (Alur & Dill 1990) Language inclusion is undecidable for timed automata.
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.
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.*

Unfortunately:

**Theorem (Alur & Dill 1990)**

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

\[
A : \quad \begin{array}{c}
\circ \quad a \quad \circ \quad a \quad \circ \\
\quad \quad \quad x:=0 \quad \quad \quad x=1?
\end{array}
\]

1. \(A\) cannot be complemented: There is no timed automaton \(B\) with \(L(B) = L(A)\).
An Uncomplementable Timed Automaton

$L(A)$:

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

$L(A)$:

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

$A:$

$L(A):$

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

- MTL = LTL + timing constraints on operators:

\[ \Box(PEDAL \rightarrow \diamond [5,10] BRAKE) \]
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 nine hundred papers according to scholar.google.com!).

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 [Ouaknine & Worrell 2005].
Metric Temporal Logic (MTL) [Koymans; de Roever; Pnueli ∼1990] is a central quantitative specification formalism for timed systems.

- MTL = LTL + timing constraints on operators:
  \[ \square (PEDAL \rightarrow \lozenge [5,10] BRAKE) \]

- Widely cited and used (over nine hundred papers according to scholar.google.com!).

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

- Widely cited and used (over nine hundred papers according to scholar.google.com!).

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 [Ouaknine & Worrell 2005].)
The first-order metric logic of order \( \text{FO}(<, +1) \) extends \( \text{FO}(<) \) by the unary function ‘+1’.

Theorem (Hirshfeld & Rabinovich 2007)

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

Corollary: \( \text{FO}(<, +1) \) and \( \text{MSO}(<, +1) \) satisfiability and model checking are undecidable over \( \mathbb{R} \geq 0 \).
Metric Predicate Logic

The first-order metric logic of order (\(\text{FO}(\langle, +1)\)) extends \(\text{FO}(\langle)\) by the unary function ‘+1’.

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

\[
\forall x \ (PEDAL(x) \rightarrow \exists y \ (x + 5 \leq y \leq x + 10 \land BRAKE(y)))
\]
Metric Predicate Logic

The first-order metric logic of order \( \text{FO}(<, +1) \) extends \( \text{FO}(<) \) by the unary function ‘+1’.

For example, \( \Box(PEDAL \rightarrow \Diamond [5,10] \text{ BRAKE}) \) becomes

\[
\forall x \ (PEDAL(x) \rightarrow \exists y \ (x + 5 \leq y \leq x + 10 \land \text{ BRAKE}(y)))
\]

Theorem (Hirshfeld & Rabinovich 2007)

\( \text{FO}(<, +1) \) is strictly more expressive than \( \text{MTL} \) over \( \mathbb{R}_{\geq 0} \).
Metric Predicate Logic

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

For example, □(PEDAL → ◊[5,10] BRAKE) becomes

\[ ∀x \ (PEDAL(x) \rightarrow ∃y \ (x + 5 ≤ y ≤ x + 10 ∧ BRAKE(y))) \]

Theorem (Hirshfeld & Rabinovich 2007)

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

Corollary: FO(<, +1) and MSO(<, +1) satisfiability and model checking are undecidable over \( \mathbb{R}_{≥0} \).
The Real-Time Theory: Expressiveness

\[
\begin{align*}
&\text{MSO}(<,+1) \\
&\text{FO}(<,+1) \\
&\text{MTL}
\end{align*}
\]
The Real-Time Theory: Expressiveness

MSO(\(\leq, +1\))

FO(\(\leq, +1\))

MTL

timed automata

MTL
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

- UNDECIDABLE
- 3–clock+ reachability: PSPACE–complete
- 2–clock reachability: NP–hard
- 1–clock reachability: NLOGSPACE–complete
- 1–clock language inclusion: NON–PRIMITIVE RECURSIVE
- MSO(\(<,+1\)) model checking: UNDECIDABLE
- FO(\(<,+1\)) model checking: UNDECIDABLE
- MTL model checking: NON–PRIMITIVE RECURSIVE/ELEMENTARY
The Real-Time Theory: Complexity

Classical Theory

- $1$-clock language inclusion
  - NON-PRIMITIVE RECURSIVE
- $2$-clock reachability
  - NON-ELEMENTARY

Real-Time Theory

- $3$-clock+ language inclusion
  - UNDECIDABLE
- $3$-clock+ reachability
  - PSPACE-complete
- $2$-clock reachability
  - NP-hard

- $1$-clock... RECURSIVE
  - NON-ELEMENTARY
- 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
  - PSPACE-complete

Classical Theory Real-Time Theory
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

UNDECIDABLE

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY
(PRIMITIVE RECURSIVE)

ELEMENTARY

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

P

NP

NLOGSPACE

1–clock reachability
NLOGSPACE–complete

2–clock reachability
NP–hard

PSPACE–complete

EXPSPACE

NP

NON–PRIMITIVE RECURSIVE

NON–ELEMENTARY

UNDECIDABLE
The Real-Time Theory: Complexity

Classical Theory

- MSO(<) model checking
  - UNDECIDABLE

- FO(<) model checking
  - NON–ELEMENTARY

- LTL model checking
  - PSPACE–complete

- Language inclusion
  - PSPACE–complete

- Reachability
  - NLOGSPACE–complete

Real–Time Theory

- 3–clock+ language inclusion
  - UNDECIDABLE

- 1–clock language inclusion
  - NON–PRIMITIVE RECURSIVE

- 2–clock reachability
  - NP-hard

- PSPACE–complete

- EXPSPACE

- 3EXPSPACE

- EXPSPACE

- NP

- 2EXPSPACE

- 2EXPSPACE

- UNDECIDABLE

- NON–ELEMENTARY

- PRIMITIVE RECURSIVE
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
- 3-clock+ reachability: PSPACE-complete
- 2-clock reachability: NP-hard
- 1-clock reachability: NLOGSPACE-complete
The Real-Time Theory: Complexity

Classical Theory

Real-Time Theory

MSO(<) model checking
NON–ELEMENTARY

FO(<) model checking
NON–ELEMENTARY

LTL model checking
PSPACE–complete

language inclusion
PSPACE–complete

reachability
NLOGSPACE–complete

2–clock+ language inclusion
UNDECIDABLE

2–clock+ reachability
NP–hard

1–clock reachability
NLOGSPACE–complete

3–clock+ reachability
PSPACE–complete

1–clock language inclusion
NON–PRIMITIVE RECURSIVE

NP

EXPSPACE

PSPACE

3EXPSPACE

EXPSPACE

2EXPSPACE

PSPACE–complete

P

NLOGSPACE

NP–hard
The Real-Time Theory: Complexity

Classical Theory

Real-Time Theory

- MSO(<) model checking: NON–ELEMENTARY
- FO(<) model checking: NON–ELEMENTARY
- LTL model checking: PSPACE–complete
- Language inclusion: PSPACE–complete
- Reachability: NLOGSPACE–complete
- 1–clock language inclusion: NON–PRIMITIVE RECURSIVE
- 2–clock+ language inclusion: NON–ELEMENTARY
- MTL model checking: NON–PRIMITIVE RECURSIVE/UNDECIDABLE
- 3–clock+ reachability: PSPACE–complete
- 2–clock reachability: NP–hard
- 1–clock reachability: NLOGSPACE–complete

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

- FO(≺,+1) model checking
  - UNDECIDABLE

- MTL model checking
  - NON–PRIMITIVE RECURSIVE
  - NON–ELEMENTARY

- 2–clock+ language inclusion
  - 2EXPSPACE

- 1–clock language inclusion
  - NON–PRIMITIVE RECURSIVE

- 3–clock+ reachability
  - PSPACE–complete

- 2–clock reachability
  - NP–hard

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

Classical Theory

- MSO(\textless, +1) model checking: UNDECIDABLE
- FO(\textless, +1) model checking: UNDECIDABLE
- MTL model checking: NON-PRIMITIVE RECURSIVE
- 1-clock reachability: NP-hard
- 2-clock reachability: NLOGSPACE-complete
- 3-clock+ reachability: 3EXPSPACE

Real-Time Theory

- MSO(\textless) model checking: NON-ELEMENTARY
- FO(\textless) model checking: NON-ELEMENTARY
- 1-clock language inclusion: NON-PRIMITIVE RECURSIVE

For more details on the complexity classes and their relations, refer to the diagram above.
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], [Ouaknine & Worrell 2003]

- **Fuzzy semantics / noise-based techniques:**
  [Maass & Orponen 1996],
  [Gupta, Henzinger, Jagadeesan 1997],
  [Fränzle 1999], [Henzinger & Raskin 2000], [Puri 2000],
  [Asarin & Bouajjani 2001], [Ouaknine & 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:**
  [Ouaknine & 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.
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($\lt, +1$)
- This reverses Vardi’s ‘automata-theoretic approach to verification’ paradigm!
Theorem (Shelah 1975)

MSO($<$) is undecidable over $[0, 1)$.
Monadic Second-Order Logic

Theorem (Shelah 1975)

$MSO(<) \text{ is undecidable over } [0, 1)$. 

By contrast,

Theorem

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

Timed behaviours are modelled as flows (or signals):
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

Timed behaviours are modelled as flows (or signals):

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

Predicates must have finite variability: Disallow e.g.

Then:

Theorem (Rabinovich 2002)

MSO(<) satisfiability over finitely-variable flows is decidable.
Finite Variability

Timed behaviours are modelled as \textit{flows} (or \textit{signals}): 

\[ f : [0, T) \rightarrow \mathcal{P}(\mathbb{M}) \]

\( P: \)

\( 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \)

\( Q: \)

\( 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \)

\( R: \)

\( 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \)

Predicates must have finite variability: 

Disallow e.g.

\( P: \)

\( 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \)

Then: 

Theorem (Rabinovich 2002) 

MSO(<) satisfiability over finitely-variable flows is decidable.
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

Predicates must have finite variability:

\[ \text{Disallow e.g.} \]

Then:

Theorem (Rabinovich 2002)

MSO(<) satisfiability over finitely-variable flows is decidable.
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

Predicates must have finite variability:
Finite Variability

Timed behaviours are modelled as flows (or signals):

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

Predicates must have finite variability:

Disallow e.g. \( Q \):

Then:

Theorem (Rabinovich 2002)

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:

Disallow e.g. \( Q \):

Then:

Theorem (Rabinovich 2002)

\( MSO(\prec) \) 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>Logic</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.*
MSO($\prec, +1$) Time-Bounded Satisfiability

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

- Let $\varphi$ be an $\text{MSO}(\prec, +1)$ formula and let $T \in \mathbb{N}$. 
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:

  $\varphi$ is satisfiable over $[0, T)$ $\iff$ $\overline{\varphi}$ is satisfiable over $[0, 1)$
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($<$, $+$1) to MSO($<$)
From MSO($\prec, +1$) to MSO($\prec$)
From MSO($\prec, +1$) to MSO($\prec$)

$P$:

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
\hline
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
\end{array}
\]
From MSO(\(<, +1\)) to MSO(\(<\))

\[ P:\]

\[
\begin{align*}
0 &\quad 1 &\quad 2 &\quad 3 \\
\end{align*}
\]

\[
\begin{align*}
0 &\quad 1 \\
0 &\quad 1 \\
0 &\quad 1 \\
\end{align*}
\]
From MSO($\langle, +1 \rangle$) to MSO($\langle \rangle$)

\[
\begin{align*}
\forall x \psi(x) & \rightarrow \forall x (\psi(x) \land \psi(x + 1) \land \psi(x + 2)) \\
x + k_1 < y + k_2 & \rightarrow \\
P(x + k) & \rightarrow P_k(x) \\
\forall \psi & \rightarrow \forall \forall \forall \psi
\end{align*}
\]

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

$P$:

$P_0$:

$P_1$:

$P_2$:
From MSO(\(<, +1\)) to MSO(\(<\))
From MSO(\(<\),+1) to MSO(\(\langle\))
From MSO($\prec, +1$) to MSO($\prec$)
From MSO(\(<\), +1) to MSO(\(<\))
From MSO($\langle, +1 \rangle$) to MSO($\langle \rangle$)

$P$:

$P_0$:

$P_1$:

$P_2$:
From MSO(\(<, +1\)) to MSO(\(<\))

Replace every:
- \(\forall x \psi(x)\)
From MSO($<, +1$) to MSO($<$)

Replace every:

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

$P$:  

$P_0$:  

$P_1$:  

$P_2$:  

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

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:

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

\[ P: \]
\[ P_0: \]
\[ P_1: \]
\[ P_2: \]

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 \(\begin{cases} 
\text{true} & \text{if } k_1 < k_2 \\
\text{false} & \text{if } k_1 > k_2
\end{cases}\)

\(P(x + k)\) 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
The Time-Bounded Theory: Expressiveness

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

- LTL
- MSO(\(<\))
- MSO(\(<,+1\))
- FO(\(<,+1\))
- MTL
- FO(\(<\))
- timed automata
- automata

Diagram:

- MSO(\(<,+1\))
- FO(\(<,+1\))
- MTL
- LTL
- FO(\(<\))
- automata
- timed automata
The Time-Bounded Theory: Expressiveness

- alternating timed automata
- timed automata
- automata
- MSO(<)
- MSO(<,+1)
- FO(<)
- FO(<,+1)
- FO(<)
- 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

- UNDECIDABLE

- NON–PRIMITIVE RECURSIVE
  - NON–ELEMENTARY
    - PRIMITIVE RECURSIVE

- ELEMENTARY
  - ...

- 3EXPSPACE

- 2EXPSPACE

- EXPSPACE

- PSPACE

- NP

- P

- NLOGSPACE
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

NON–ELEMENTARY
(PRIMITIVE RECURSIVE)

NON–PRIMITIVE RECURSIVE

UNDECIDABLE

ELEMENTARY

3EXPSPACE

2EXPSPACE

EXPSPACE

PSPACE

NP

P

NLOGSPACE

NP

NON–PRIMITIVE RECURSIVE

PSPACE

2EXPSPACE

EXPSPACE

PSPACE

NP

P

NLOGSPACE

NP

P

NLOGSPACE
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

- Non-Primitive Recursive
  - UNDECIDABLE

Time-Bounded Theory

- Non-Elementary
  - (PRIMITIVE RECURSIVE)

- Elementary

- 3EXPSPACE

- 2EXPSPACE

- EXPSPACE

- PSPACE

- NP

- P

- NLOGSPACE

- Classical Theory

- Time-Bounded Theory
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

language inclusion

2EXPSPACE–complete

MSO(<) model checking

NON–ELEMENTARY

FO(<) model checking

NON–ELEMENTARY

LTL model checking

PSPACE–complete

language inclusion

PSPACE–complete

reachability

NLOGSPACE–complete

Classical Theory

Time–Bounded Theory
The Time-Bounded Theory: Complexity

Classical Theory

Time-Bounded Theory

UNDECIDABLE

- MSO(<) model checking
  - NON-ELEMENTARY
- FO(<) model checking
  - NON-ELEMENTARY
- LTL model checking
  - PSPACE-complete
- MTL model checking
  - EXPSPACE-complete
- Language inclusion
  - 2EXPSPACE-complete
- Reachability
  - NLOGSPACE-complete

NON-PRIMITIVE RECURSIVE

- ELEMENTARY
- 3EXPSPACE
- 2EXPSPACE
- EXPSPACE
- PSPACE
- NP
- P
- NLOGSPACE

NON-ELEMENTARY (PRIMITIVE RECURSIVE)

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

Language inclusion
- 2EXPSPACE-complete

Reachability
- PSPACE-complete
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
- **MSO(<,+1) model checking**: NON-ELEMENTARY
- **FO(<,+1) model checking**: NON-ELEMENTARY
- **Language inclusion**: 2EXPSPACE-complete
- **MTL model checking**: EXPSPACE-complete
- **Reachability**: PSPACE-complete

**Classical Theory**

- **NLOGSPACE-complete**: reachability
- **PSPACE-complete**: LTL model checking, language inclusion
- **EXPSPACE**: PSPACE
- **3EXPSPACE**: EXPSPACE
- **2EXPSPACE**: EXPSPACE
- **EXPTIME**: PSPACE
- **TIME**: NP
- **NON-PRIMITIVE RECURSIVE**
- **UNDECIDABLE**

**Time-Bounded Theory**

- **TIME**: NLOGSPACE
- **UNDECIDABLE**: ELEMENTARY (PRIMITIVE RECURSIVE)
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

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

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

- alternating timed automata  
  - NON–ELEMENTARY

- MTL model checking  
  - EXPSPACE–complete

- language inclusion  
  - 2EXPSPACE–complete

- reachability  
  - PSPACE–complete

- EXPSPACE  
  - ELEMENTARY (PRIMITIVE RECURSIVE)

- 3EXPSPACE  
  - NON–PRIMITIVE RECURSIVE

- UNDECIDABLE

- NP  
  - P

- PSPACE  
  - NLOGSPACE

- P

- NLOGSPACE

Classical Theory Time–Bounded Theory
Conclusion and Future Work

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

Future work:

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