On Classical, Real-Time, and Time-Bounded Verification

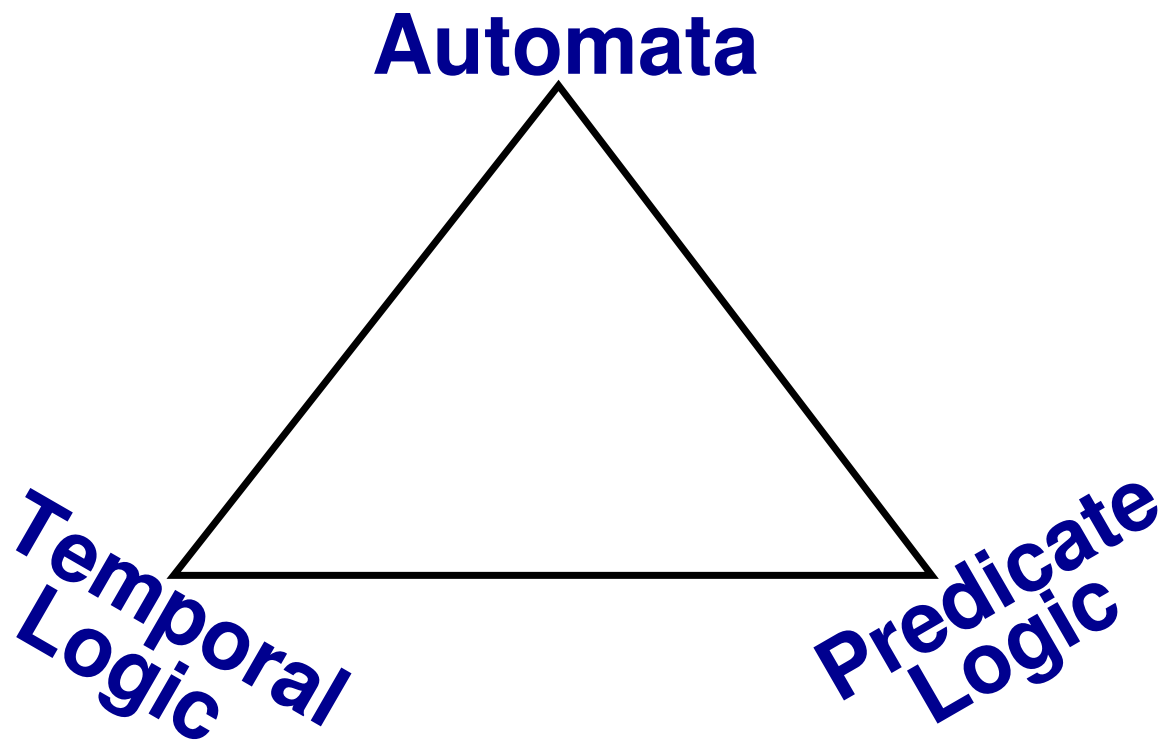
Joël Ouaknine

Oxford University Computing Laboratory

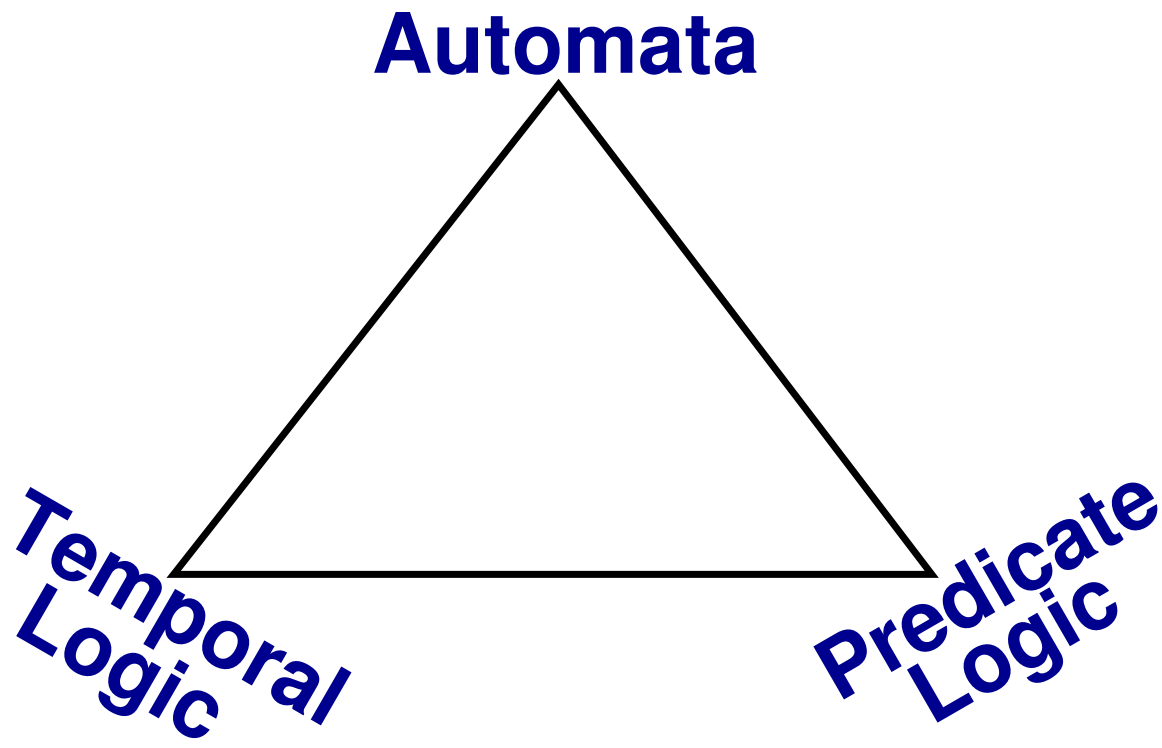
(Joint work with Alex Rabinovich and James Worrell)

ACTS 2010, CMI, Chennai, January 2010

The Classical Theory of Verification

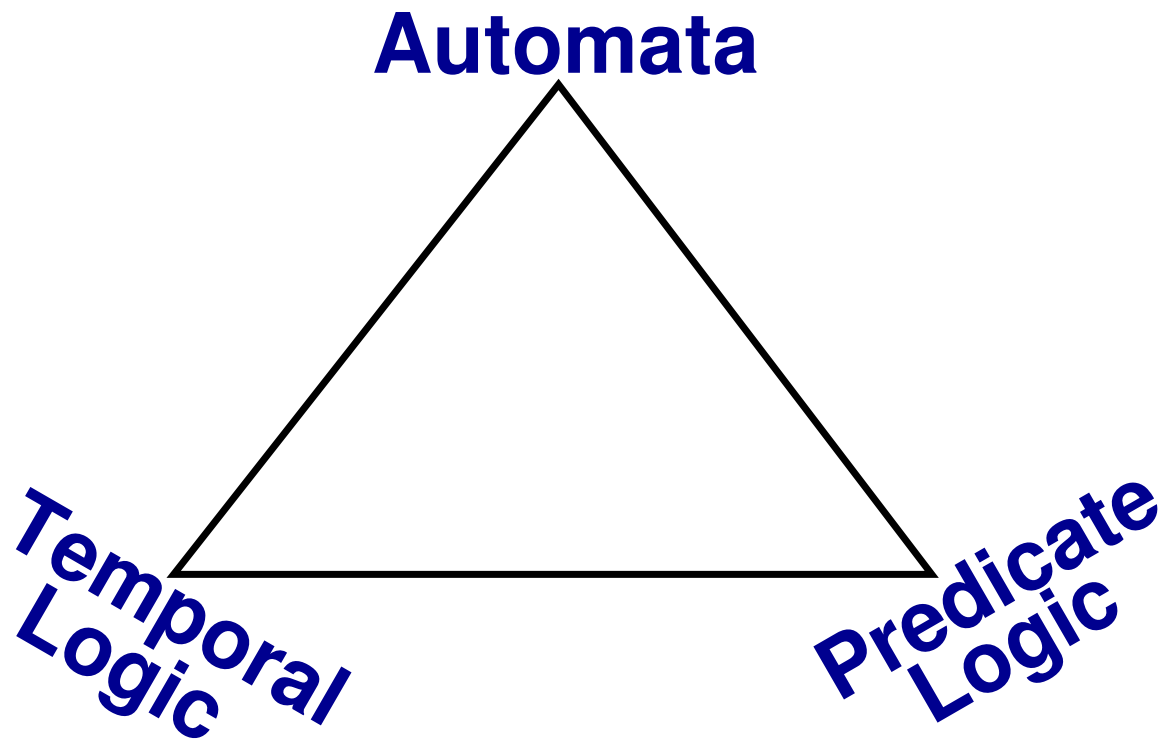


The Classical Theory of Verification



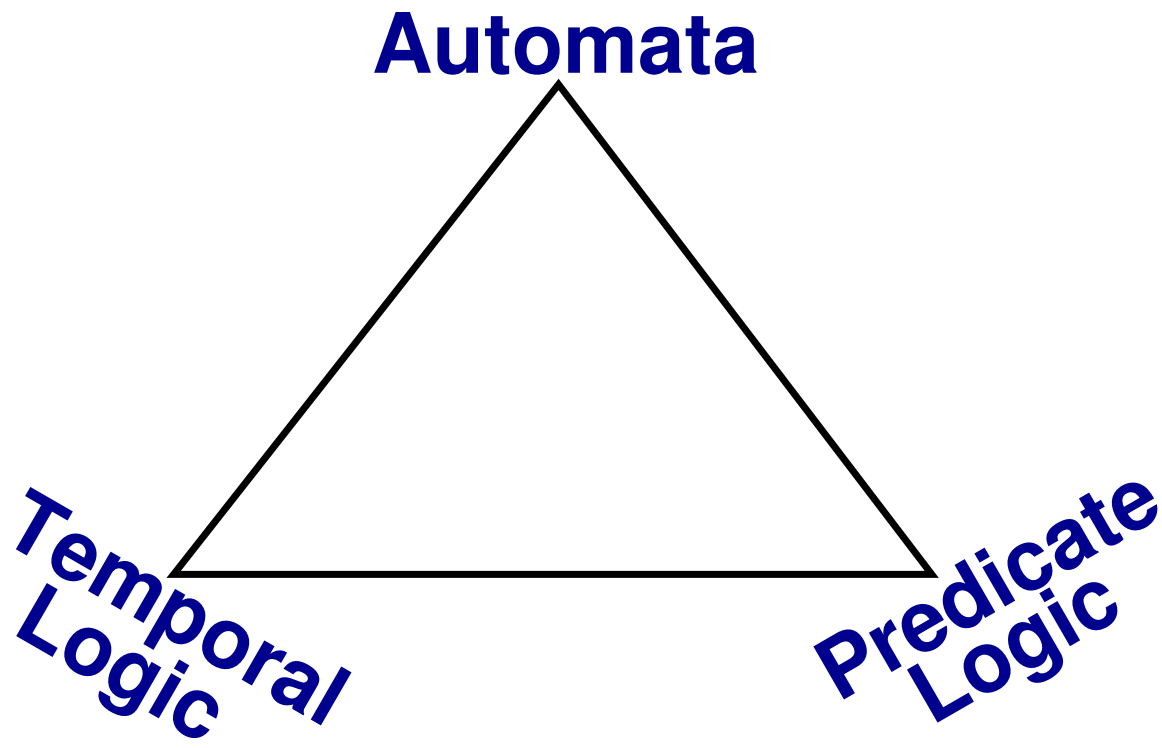
- Qualitative (order-theoretic), rather than quantitative (metric).

The Classical Theory of Verification



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

The Classical Theory of Verification



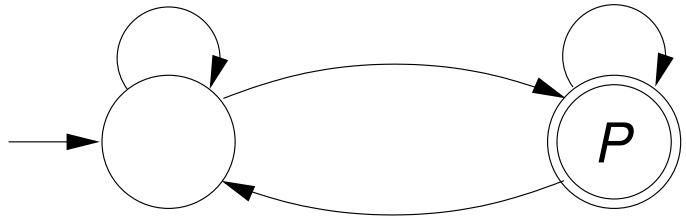
- Qualitative (order-theoretic), rather than quantitative (metric).
- Time is modelled as the naturals $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- Note: focus on linear time (as opposed to branching time).

A Simple Example

‘P occurs infinitely often’

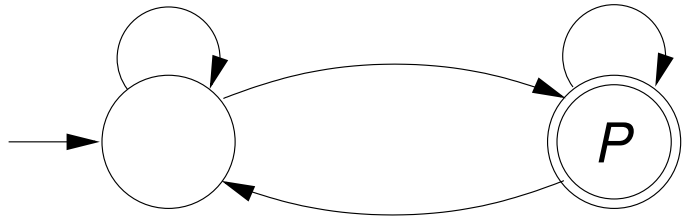
A Simple Example

‘P occurs infinitely often’



A Simple Example

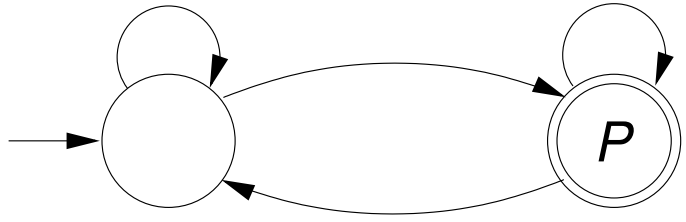
‘P occurs infinitely often’



$\Box\Diamond P$

A Simple Example

‘P occurs infinitely often’



$\Box\Diamond P$

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

Specification and Verification

Assume the system is modelled by an automaton M.

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

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

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

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

  Verification is then model checking: $M \models \theta$?

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

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

  Verification is then model checking: $M \models \theta$?

- A First-Order Logic (FO($<$)) formula φ.

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

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

  Verification is then model checking: $M \models \theta$?

- A First-Order Logic (FO($<$)) formula φ.

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

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

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

  Verification is then model checking: $M \models \theta$?

- A First-Order Logic (FO($<$)) formula φ.

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

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

Specification and Verification

Assume the system is modelled by an automaton M.
The specification can be given by:

- A Linear Temporal Logic (LTL) formula θ.

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

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

  Verification is then model checking: $M \models \theta$?

- A First-Order Logic (FO($<$)) formula φ.

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

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

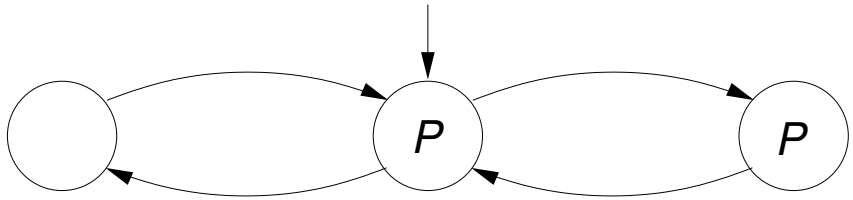
  Verification is again model checking: $M \models \varphi$?

Another Example

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

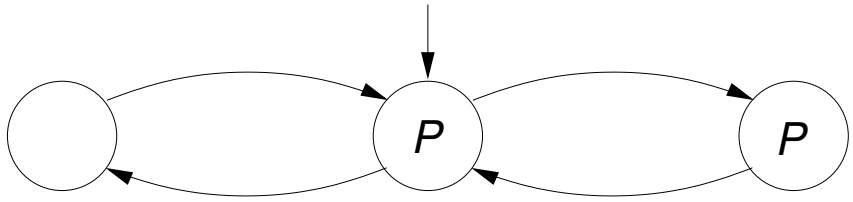
Another Example

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



Another Example

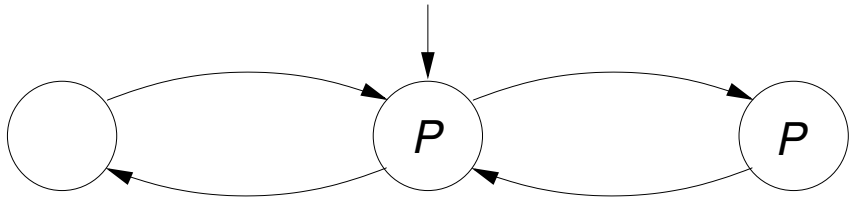
‘P holds at every even position
(and may or may not hold at odd positions)’



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

Another Example

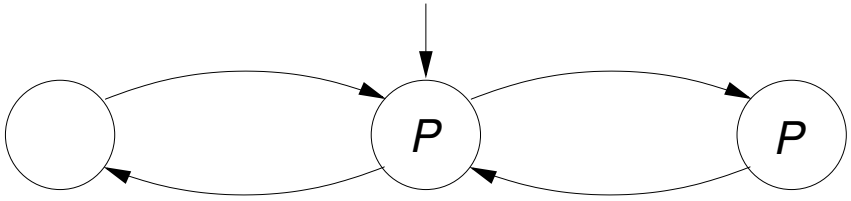
'P holds at every even position
(and may or may not hold at odd positions)'



- 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'*:

Another Example

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

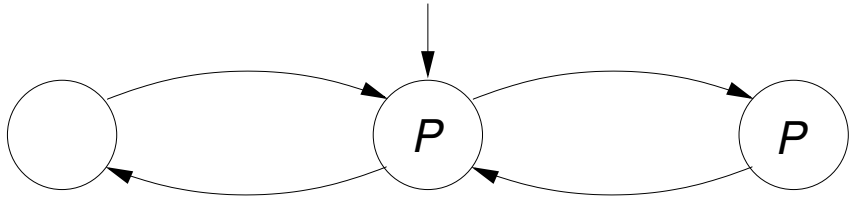


- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

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

Another Example

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



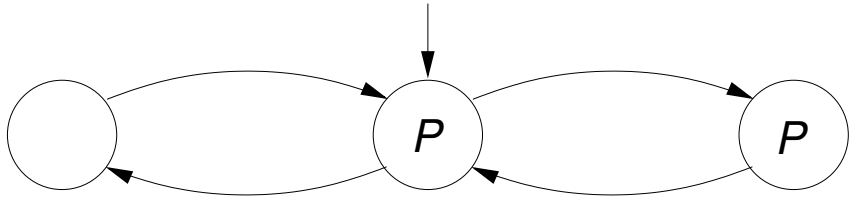
- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

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

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

Another Example

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



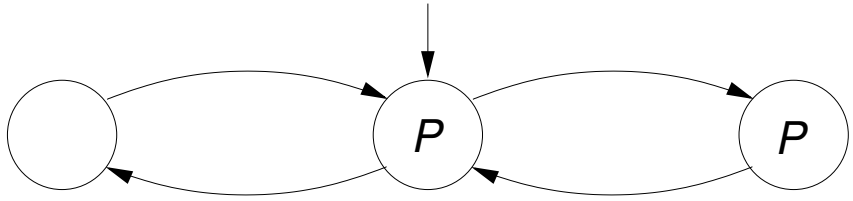
- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

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

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

Another Example

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



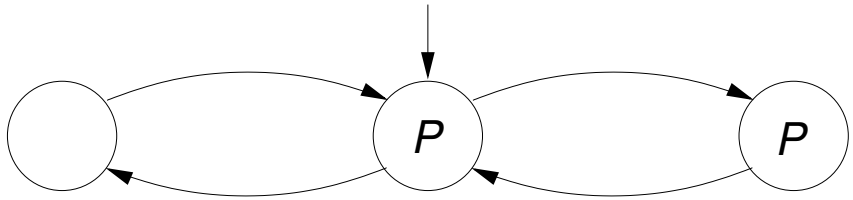
- It turns out it is impossible to capture this requirement using LTL or FO($<$).
- LTL and FO($<$) can however capture the specification: *'Q holds precisely at even positions'*:

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

- 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:

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'*:

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

- 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:

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

More Specification and Verification

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

More Specification and Verification

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

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

More Specification and Verification

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.

More Specification and Verification

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.

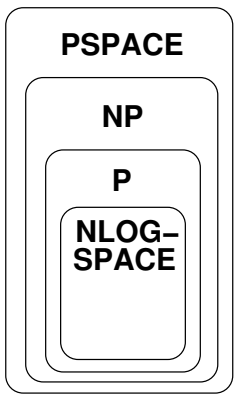
Corollary (Church 1960)

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

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

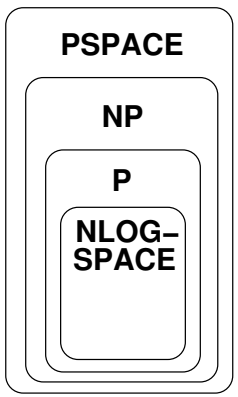
Algorithmic Complexity

- Most problems in Computer Science sit within PSPACE.



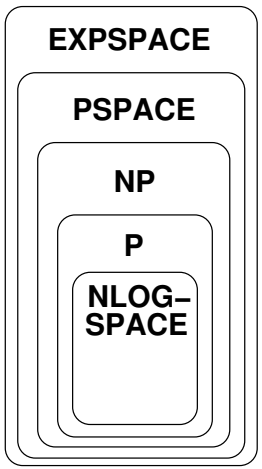
Algorithmic Complexity

- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:



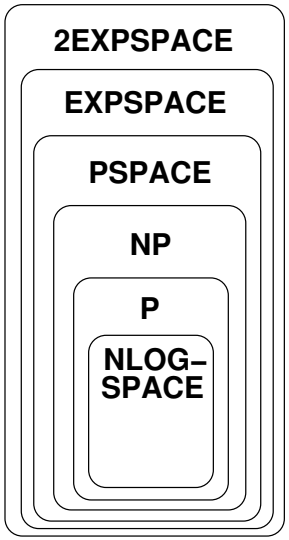
Algorithmic Complexity

- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$



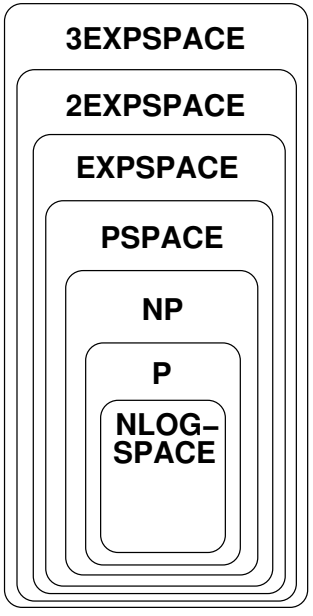
Algorithmic Complexity

- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$

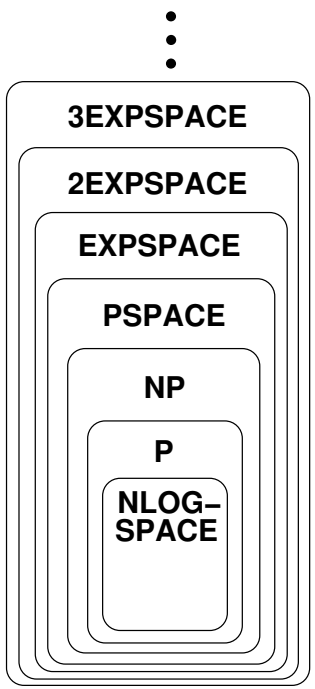


Algorithmic Complexity

- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$

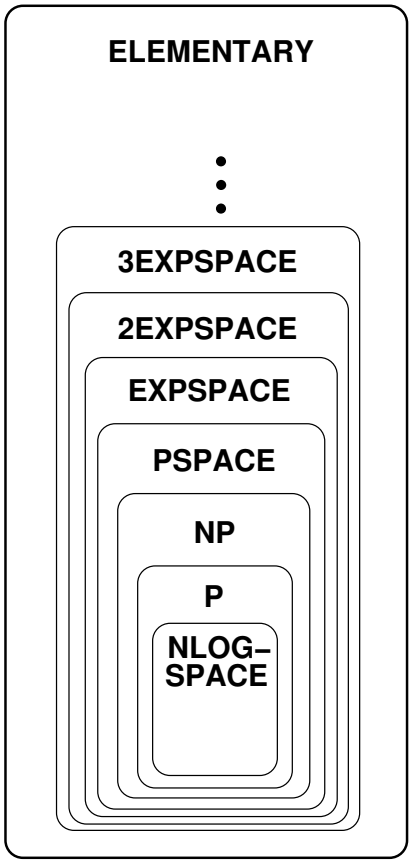


Algorithmic Complexity



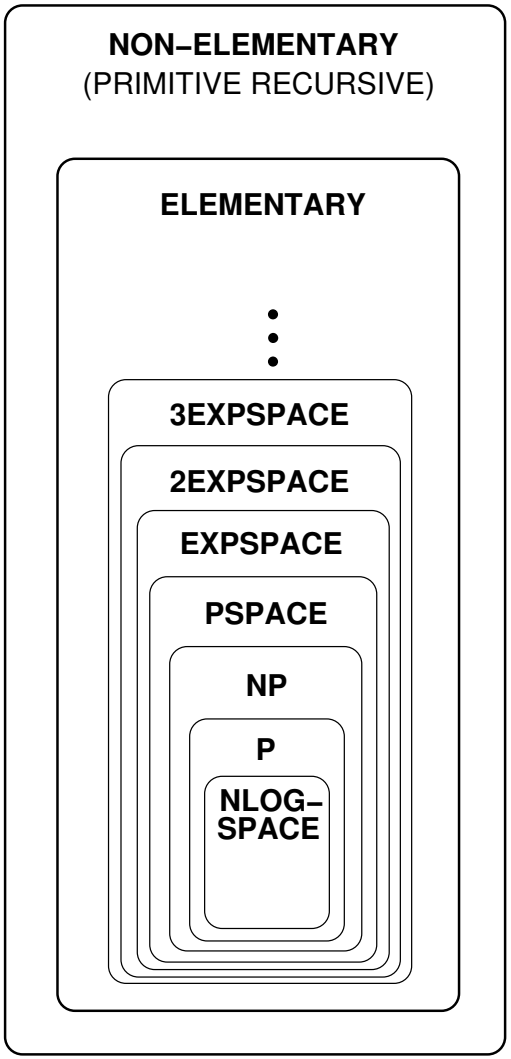
- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$
  - . . .

Algorithmic Complexity



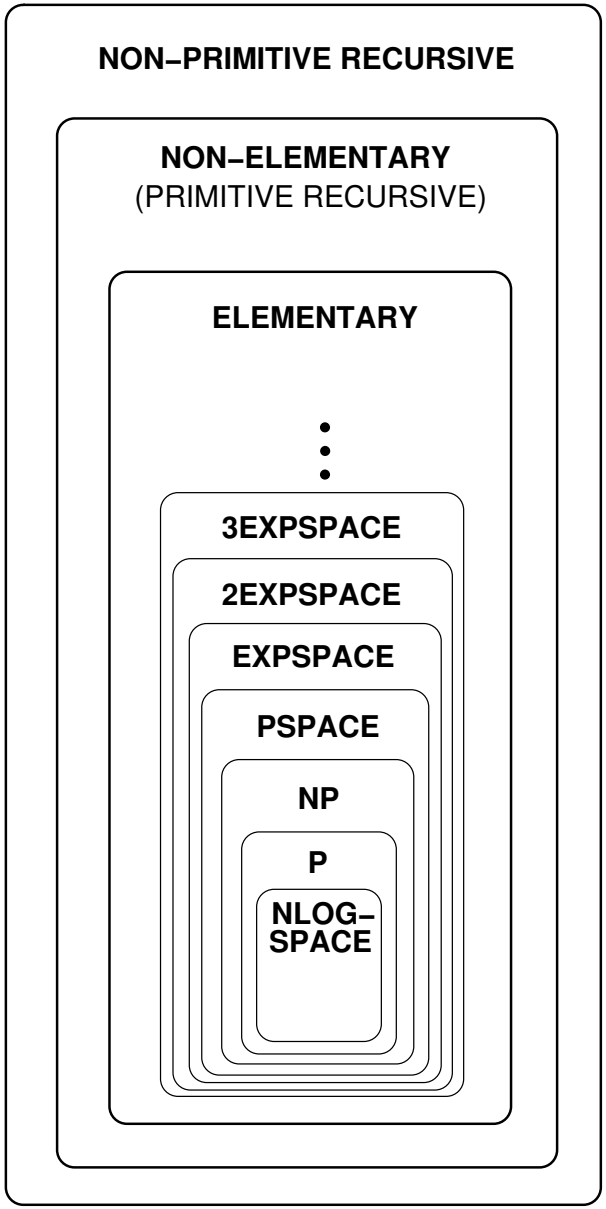
- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$
  - ...
  - ELEMENTARY: $\bigcup_{k \in \mathbb{N}} \{k\text{EXPSPACE}\}$

Algorithmic Complexity



- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$
  - ...
  - ELEMENTARY: $\bigcup_{k \in \mathbb{N}} \{k\text{EXPSPACE}\}$
  - NON-ELEMENTARY: $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{n}}}}}}_{n}$

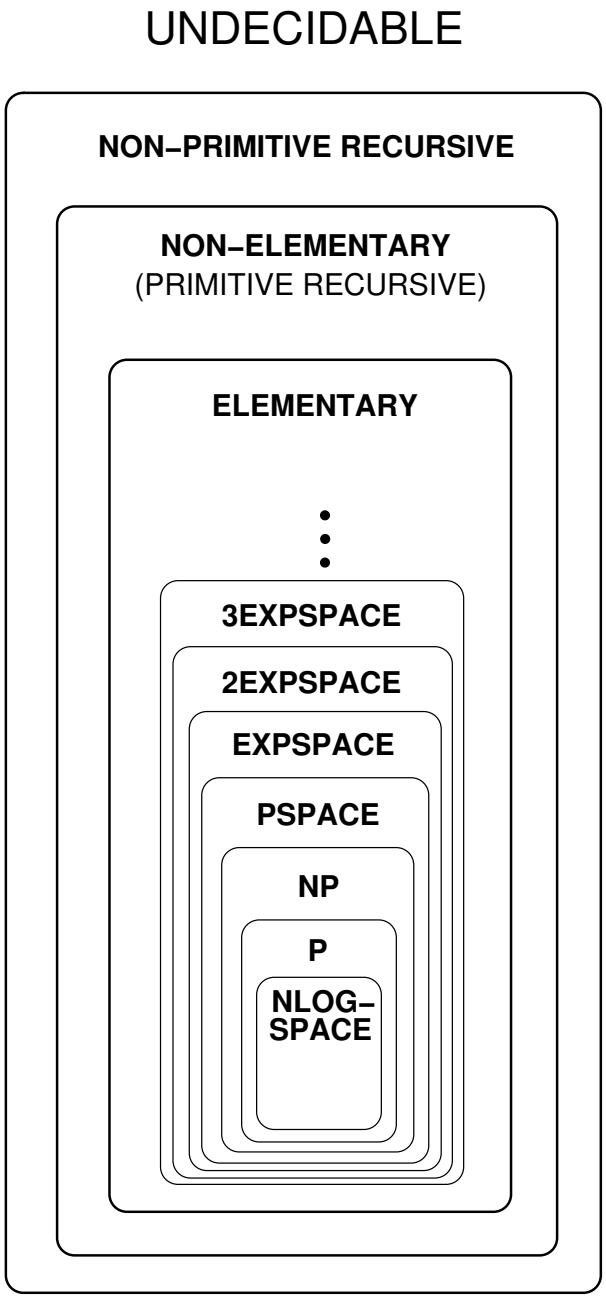
Algorithmic Complexity



- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$
  - ...
  - ELEMENTARY: $\bigcup_{k \in \mathbb{N}} \{k\text{EXPSPACE}\}$
  - NON-ELEMENTARY: $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n}$
  - NON-PRIMITIVE RECURSIVE:

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

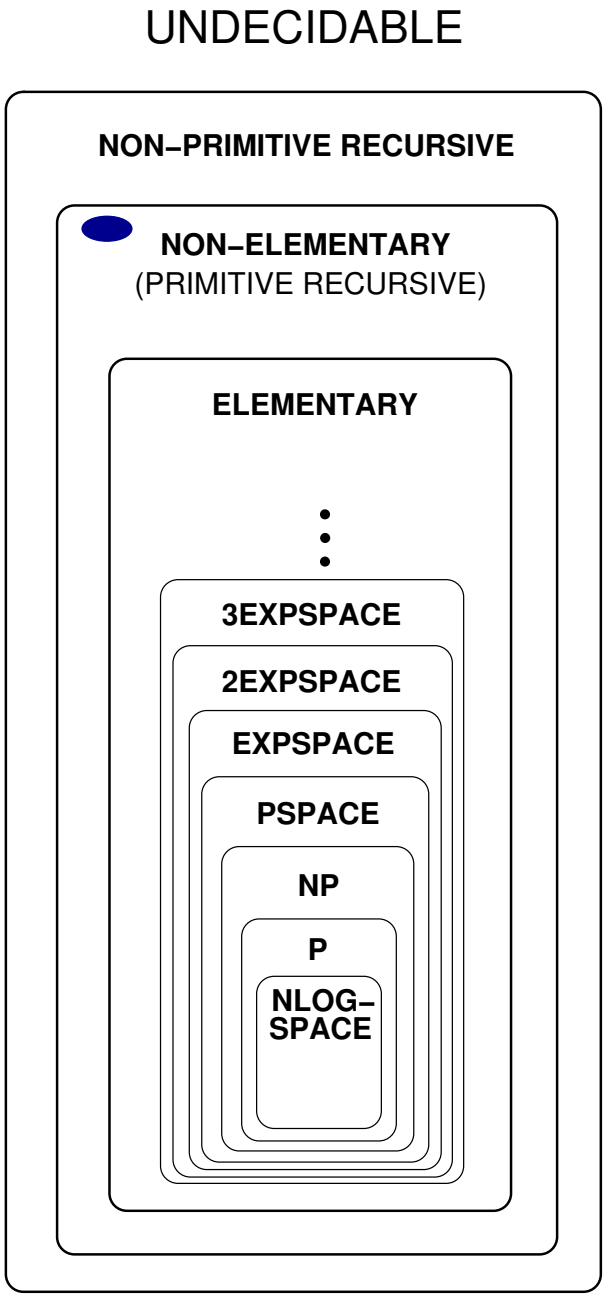
Algorithmic Complexity



- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$
  - ...
  - ELEMENTARY: $\bigcup_{k \in \mathbb{N}} \{k\text{EXPSPACE}\}$
  - NON-ELEMENTARY: $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n}$
  - NON-PRIMITIVE RECURSIVE:

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

Algorithmic Complexity



- Most problems in Computer Science sit within PSPACE.
- Hierarchy extends much beyond:
  - EXPSPACE: $2^{p(n)}$
  - 2EXPSPACE: $2^{2^{p(n)}}$
  - 3EXPSPACE: $2^{2^{2^{p(n)}}}$
  - ...
  - ELEMENTARY: $\bigcup_{k \in \mathbb{N}} \{k\text{EXPSPACE}\}$
  - NON-ELEMENTARY: $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n}$
  - NON-PRIMITIVE RECURSIVE:

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

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.

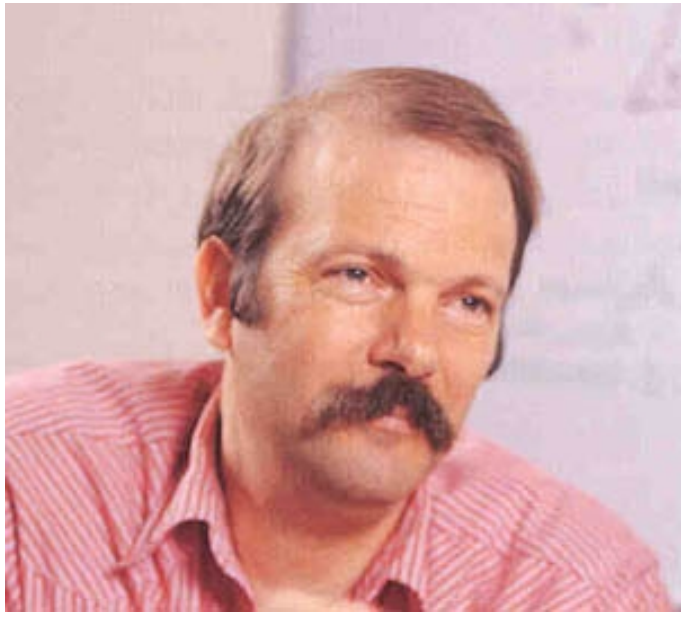
But amazingly:

Theorem (Sistla & Clarke 1982)

LTL satisfiability and model checking are PSPACE-complete.

Logics and Automata

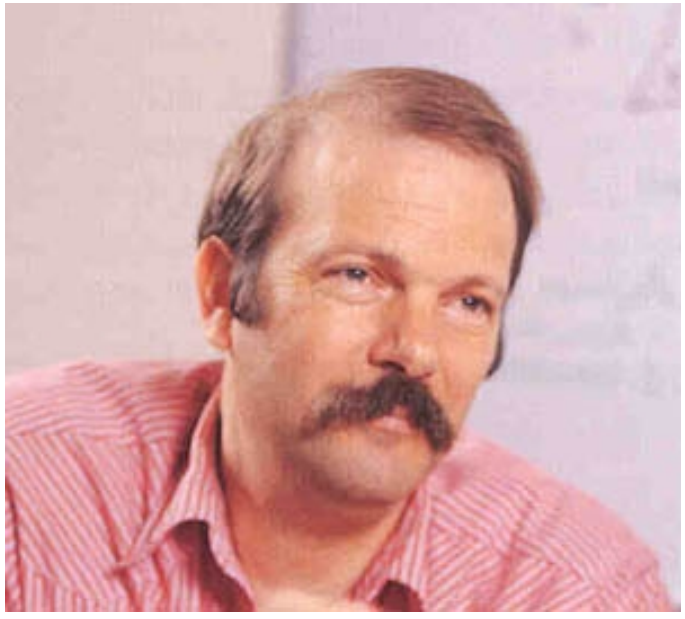
"The paradigmatic idea of the automata-theoretic approach to verification is that we can compile high-level logical specifications into an equivalent low-level finite-state formalism."



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."

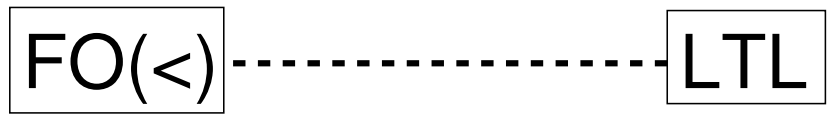


Moshe Vardi

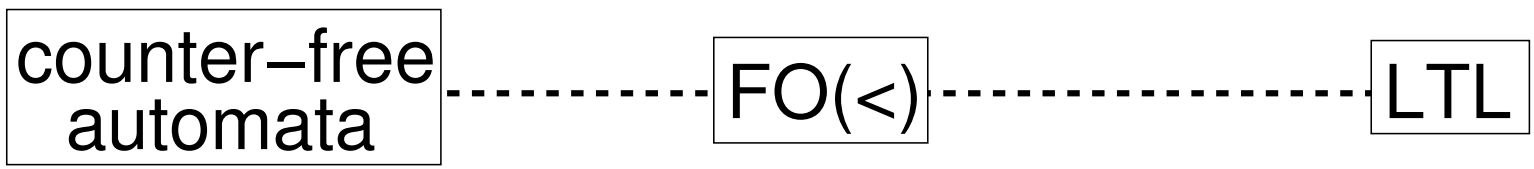
Theorem

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

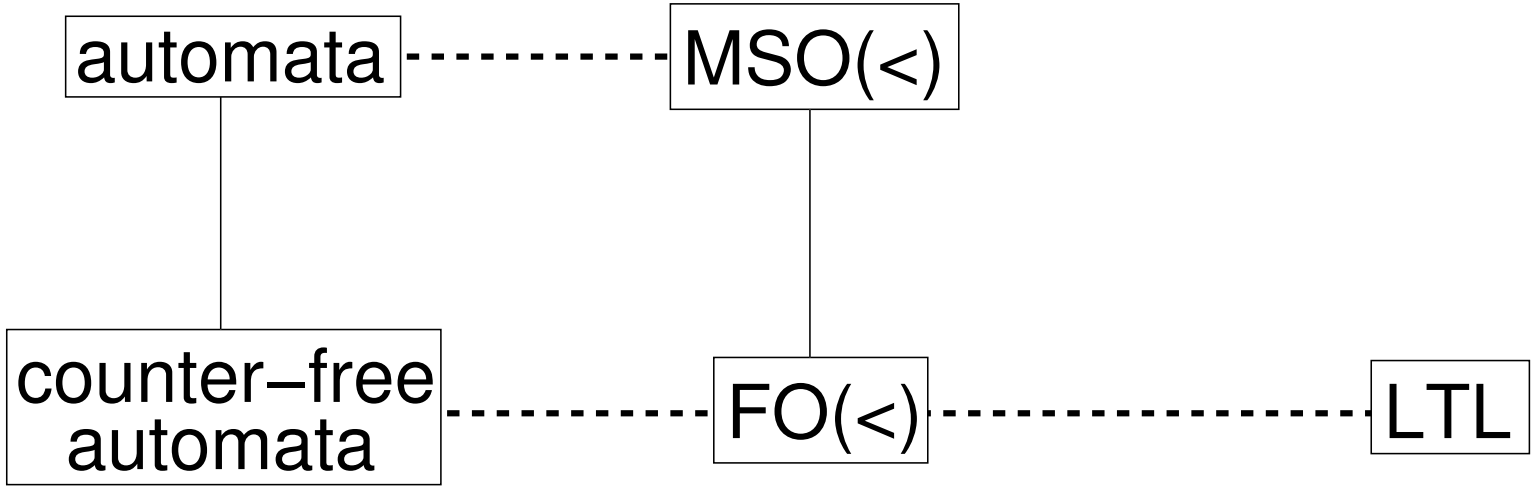
The Classical Theory: Expressiveness



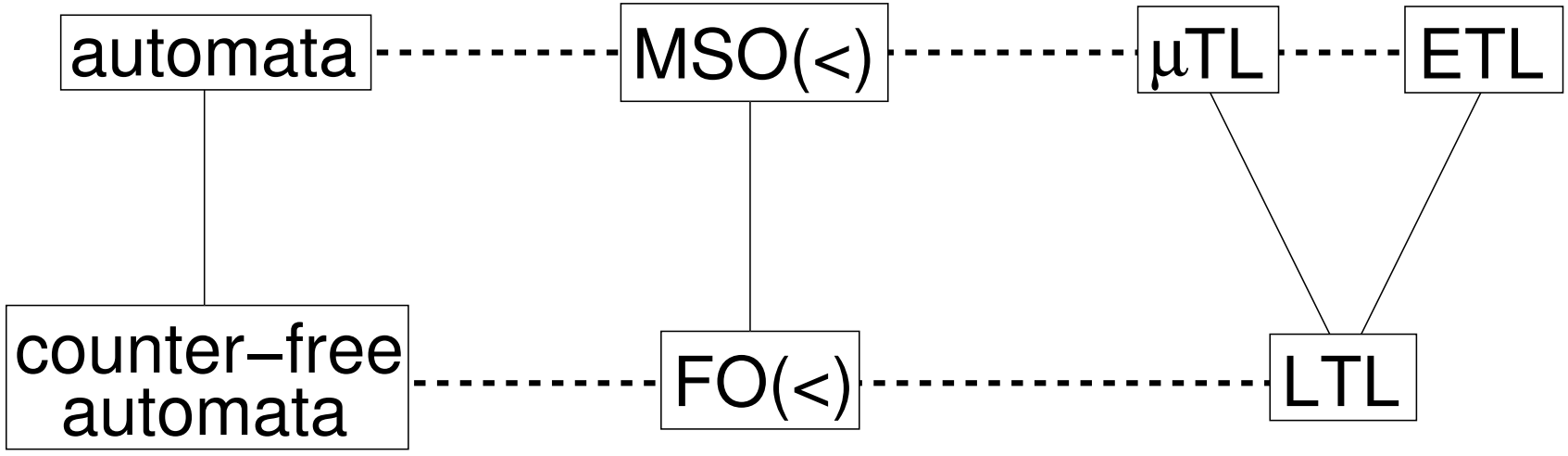
The Classical Theory: Expressiveness



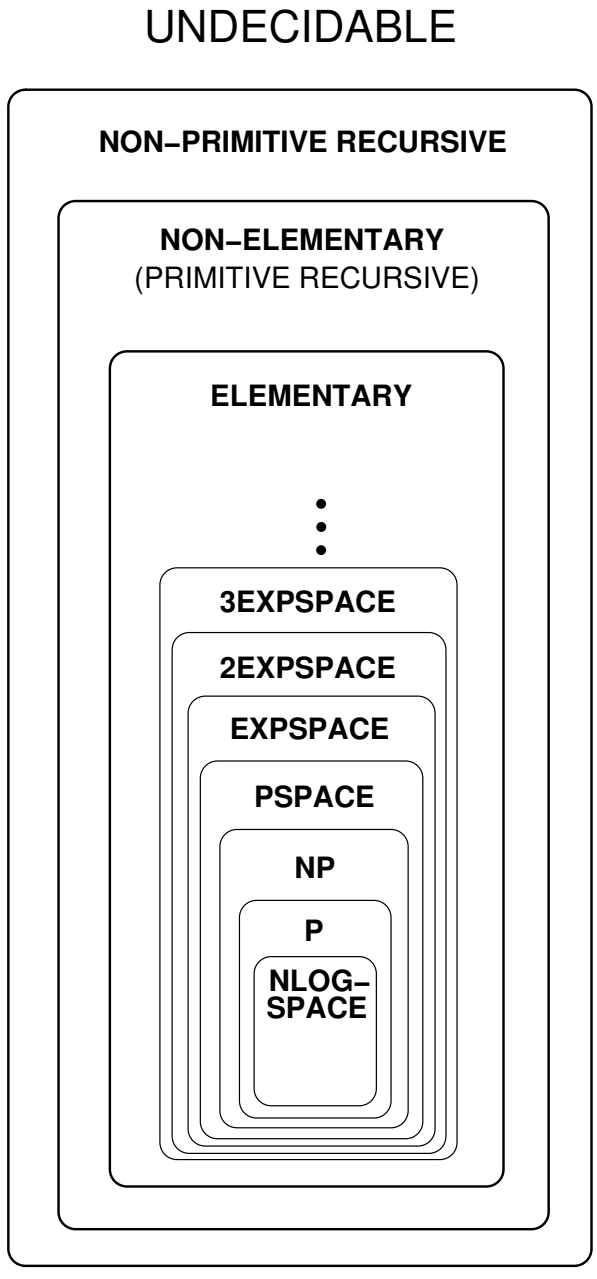
The Classical Theory: Expressiveness



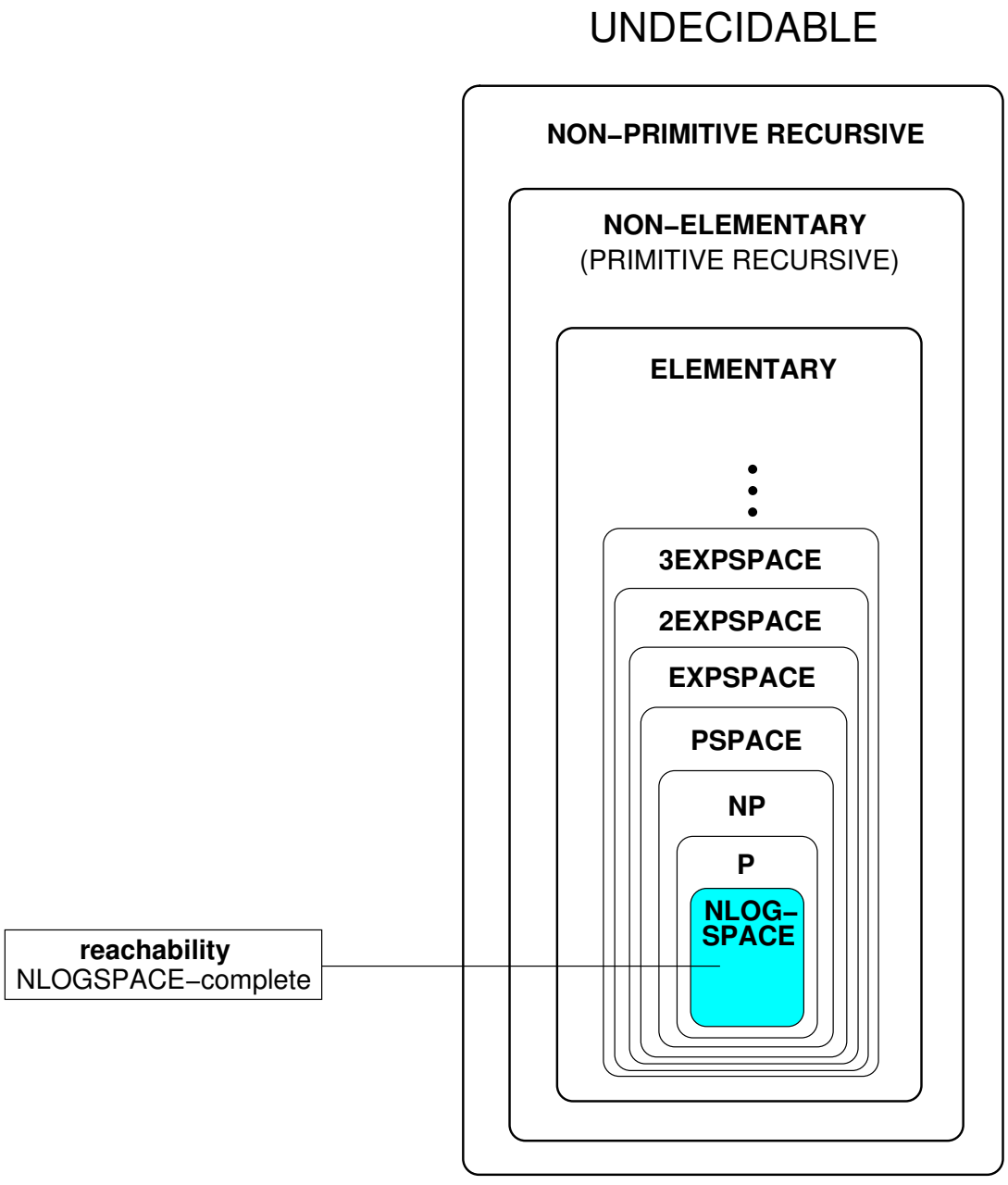
The Classical Theory: Expressiveness



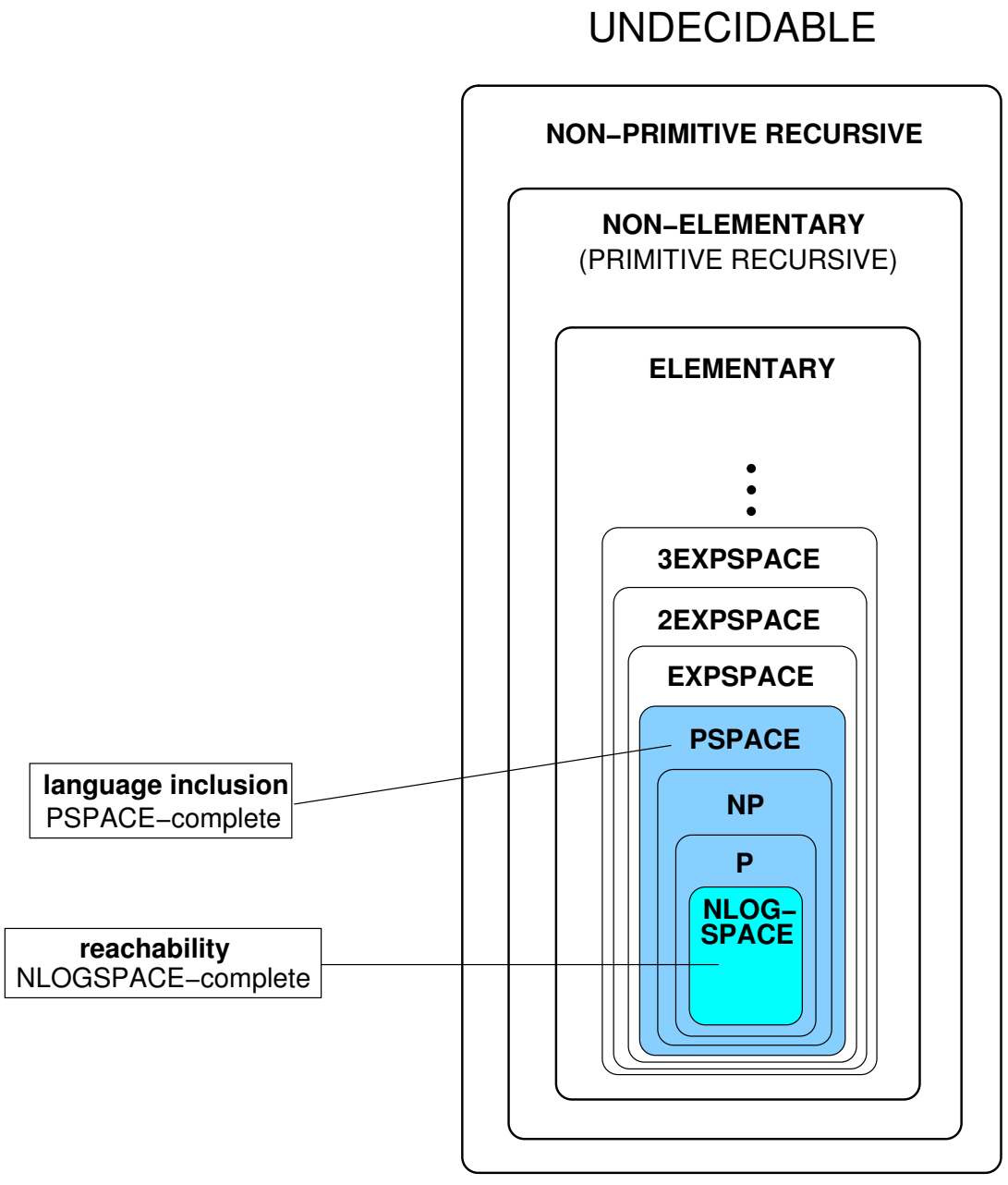
The Classical Theory: Complexity



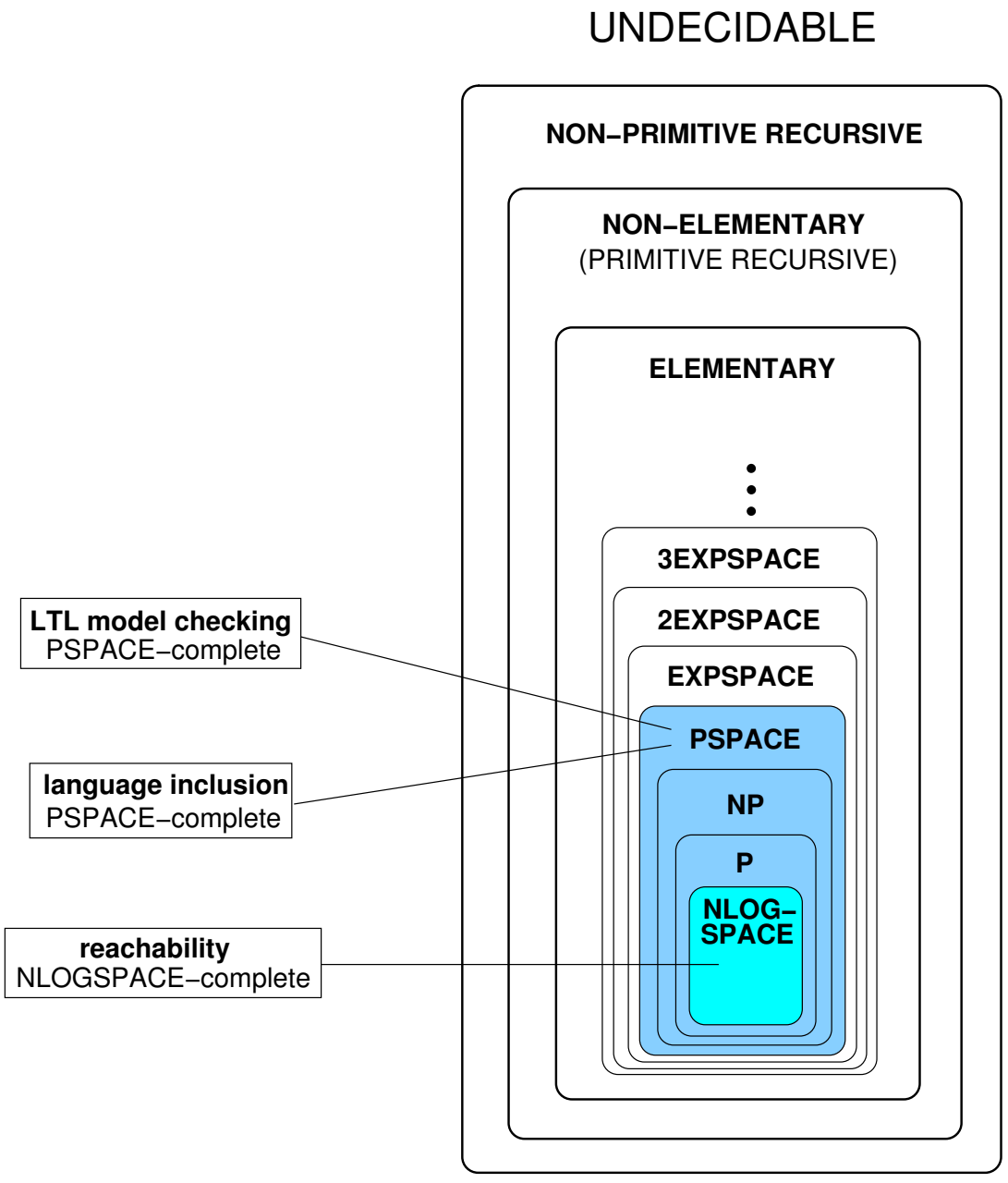
The Classical Theory: Complexity



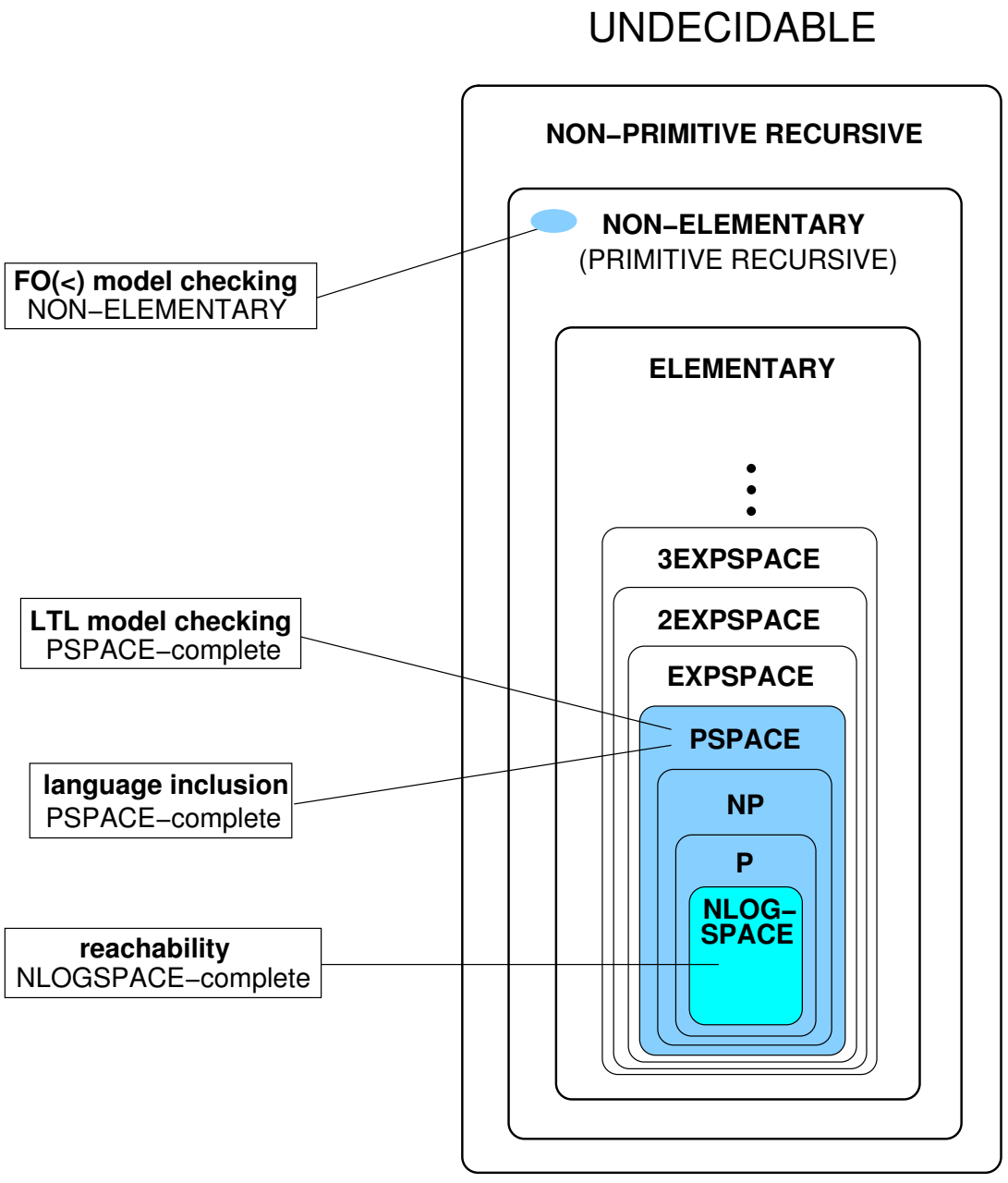
The Classical Theory: Complexity



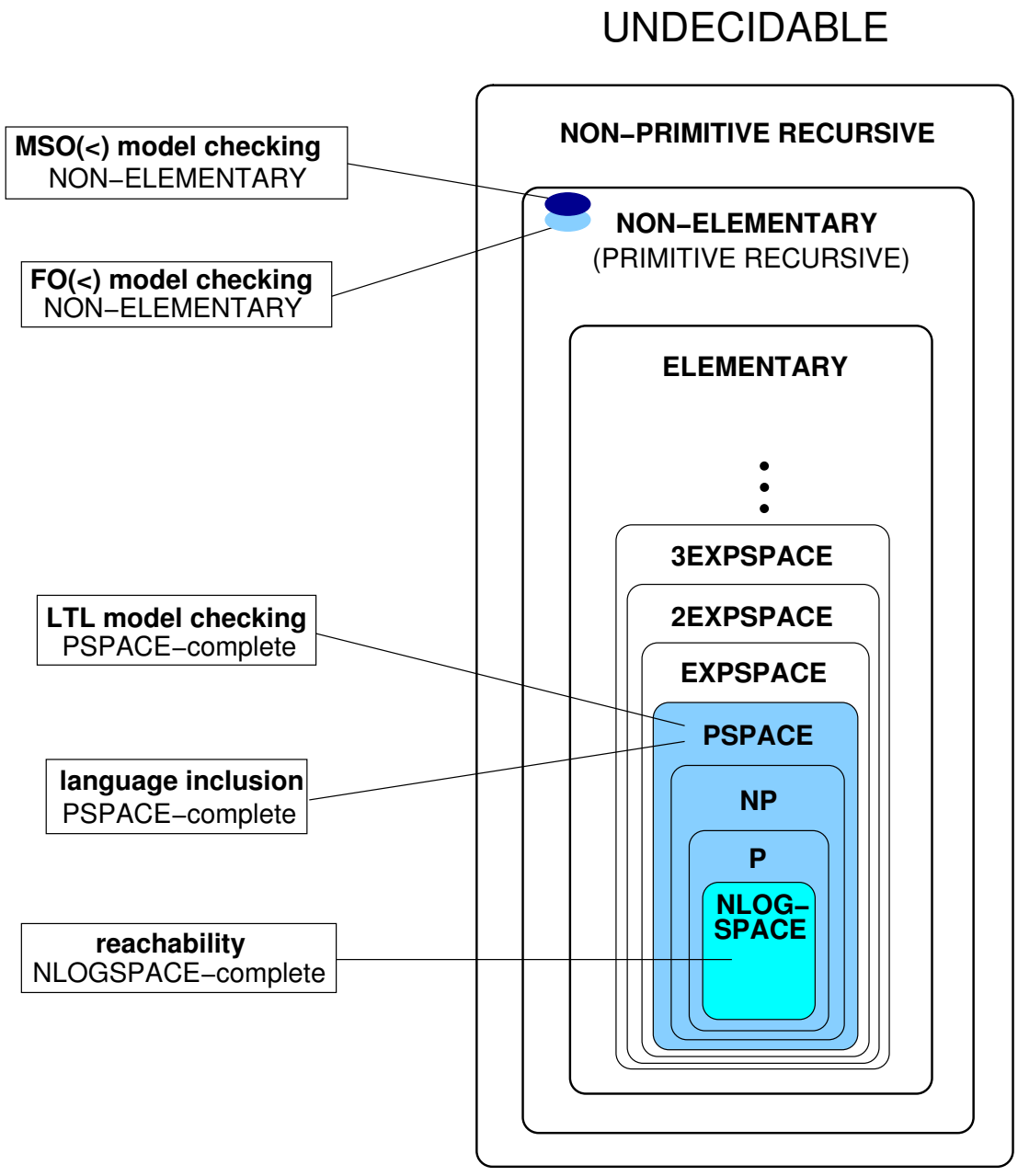
The Classical Theory: Complexity



The Classical Theory: Complexity

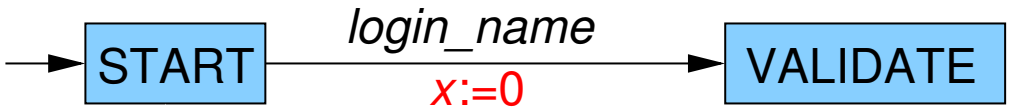


The Classical Theory: Complexity

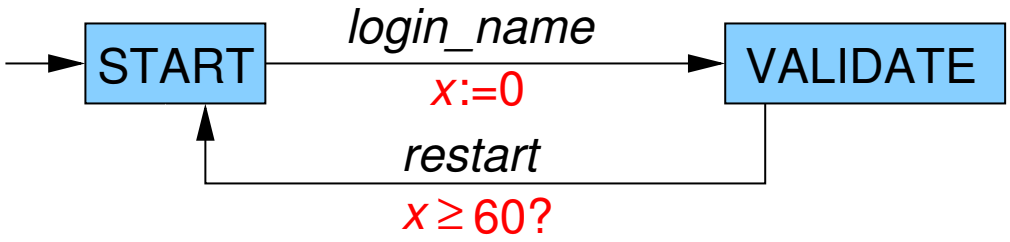


A Login Protocol

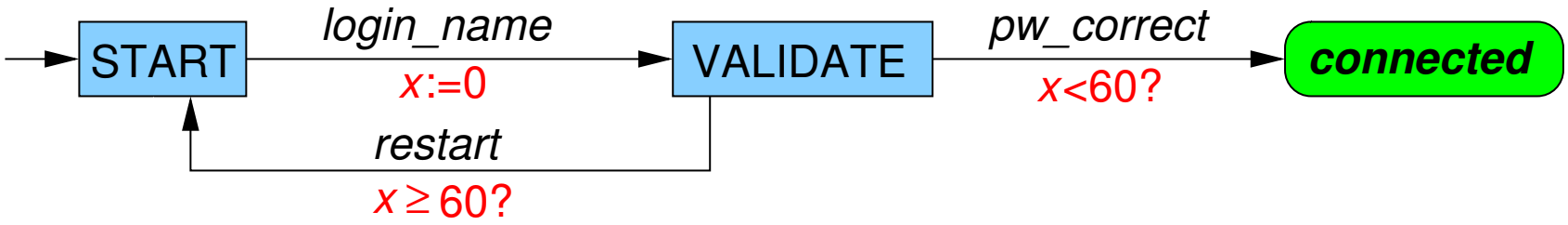
A Login Protocol



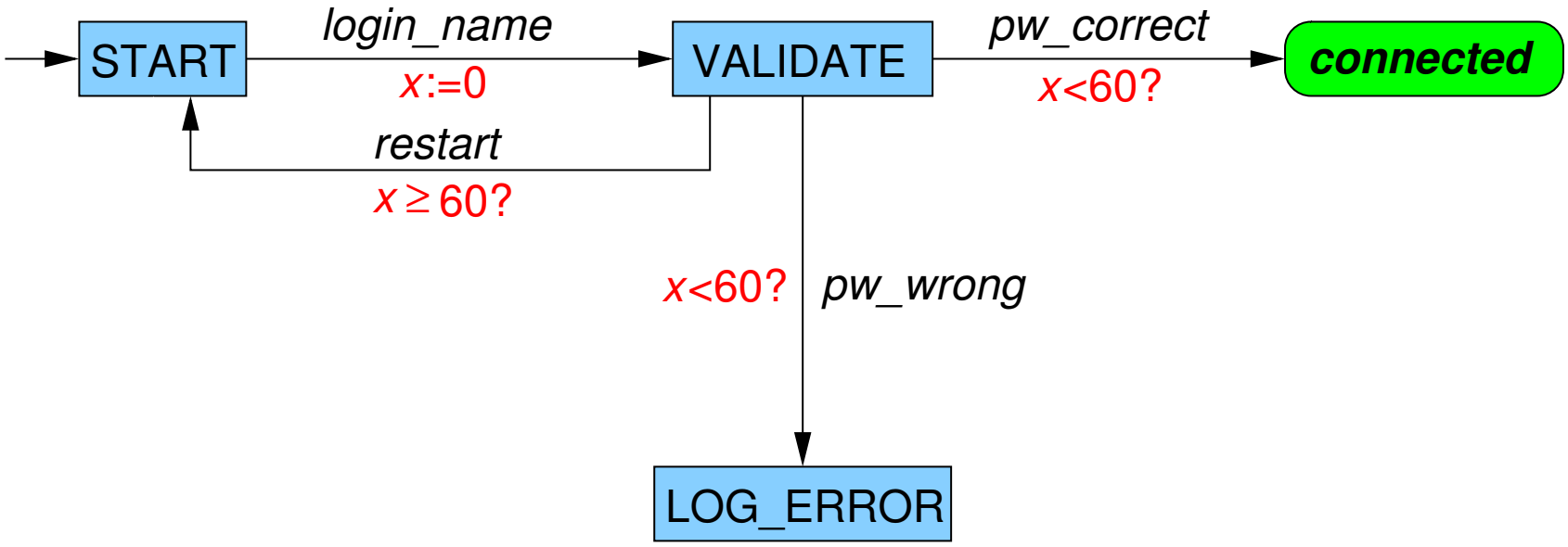
A Login Protocol



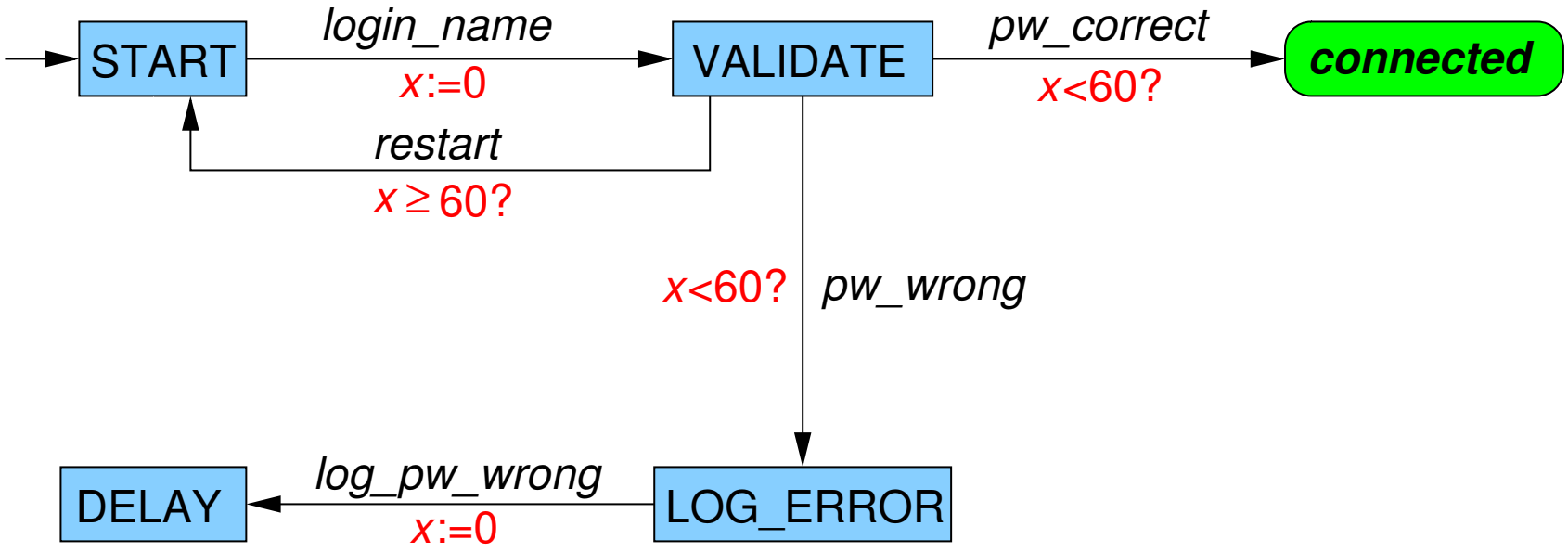
A Login Protocol



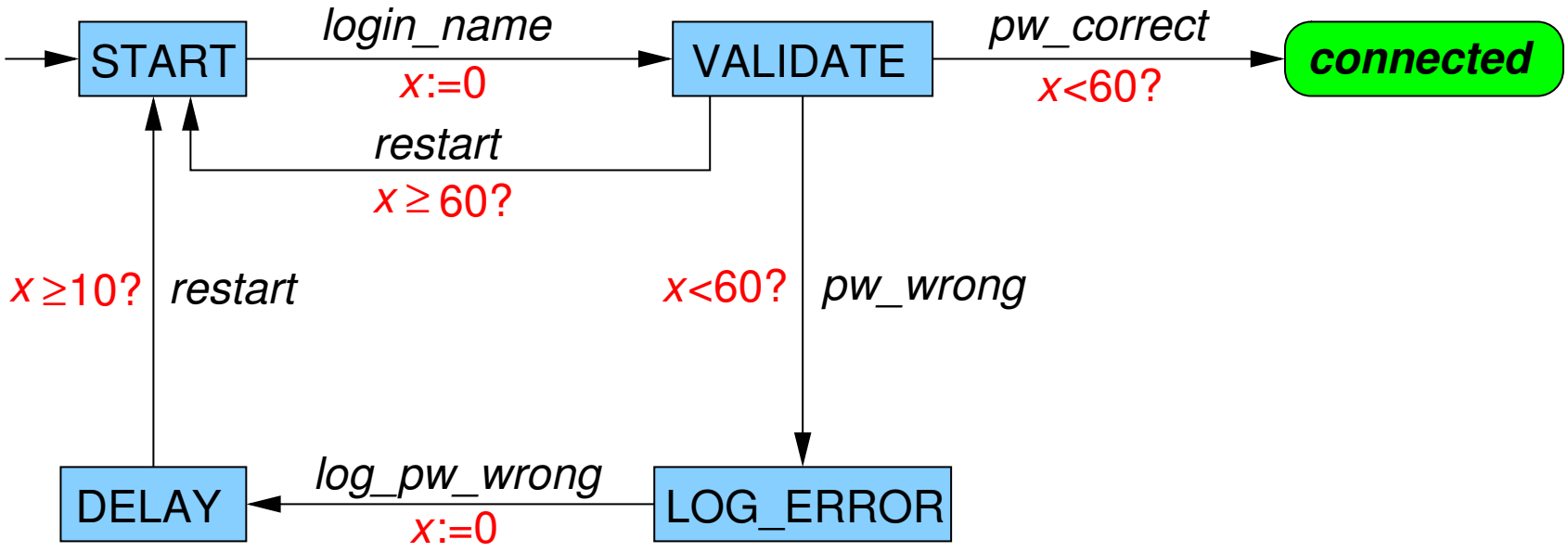
A Login Protocol



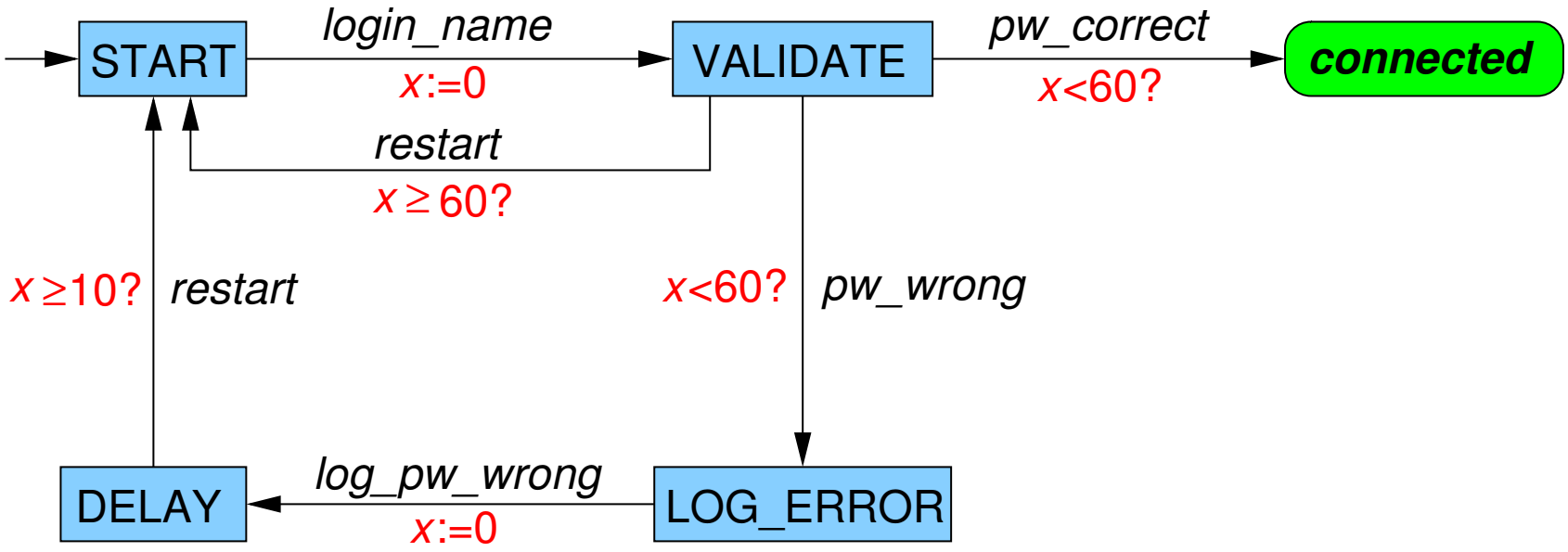
A Login Protocol



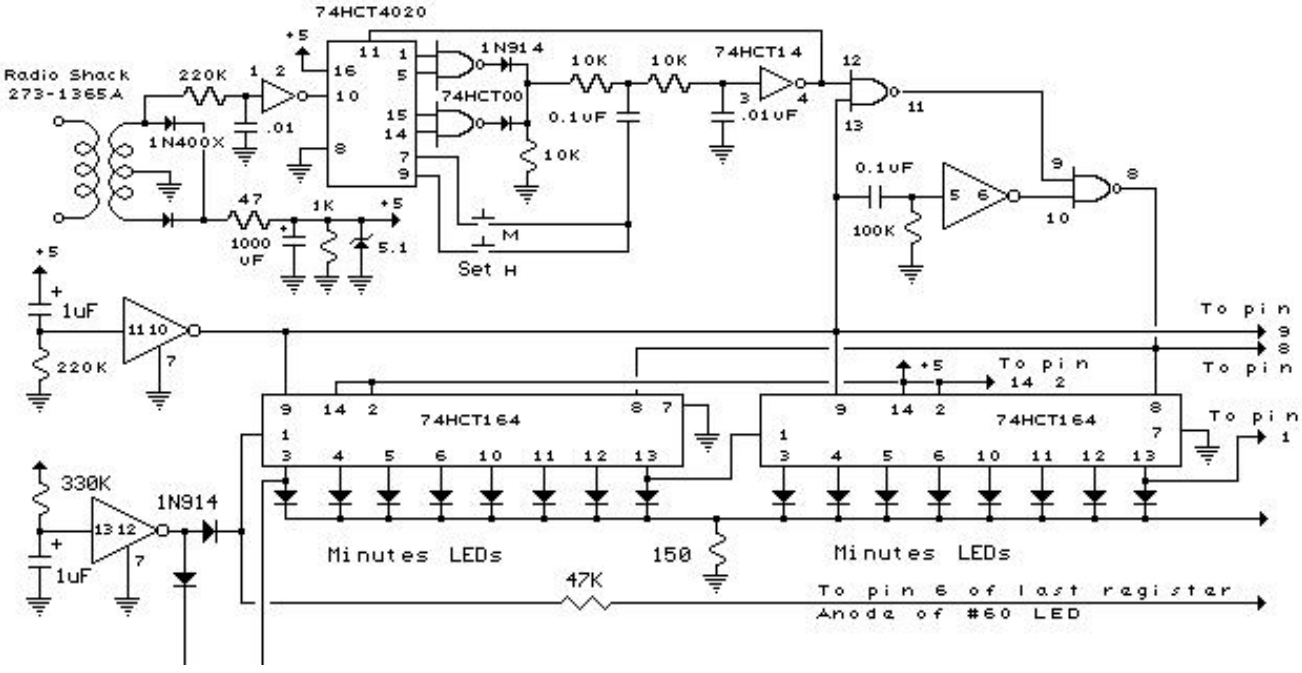
A Login Protocol

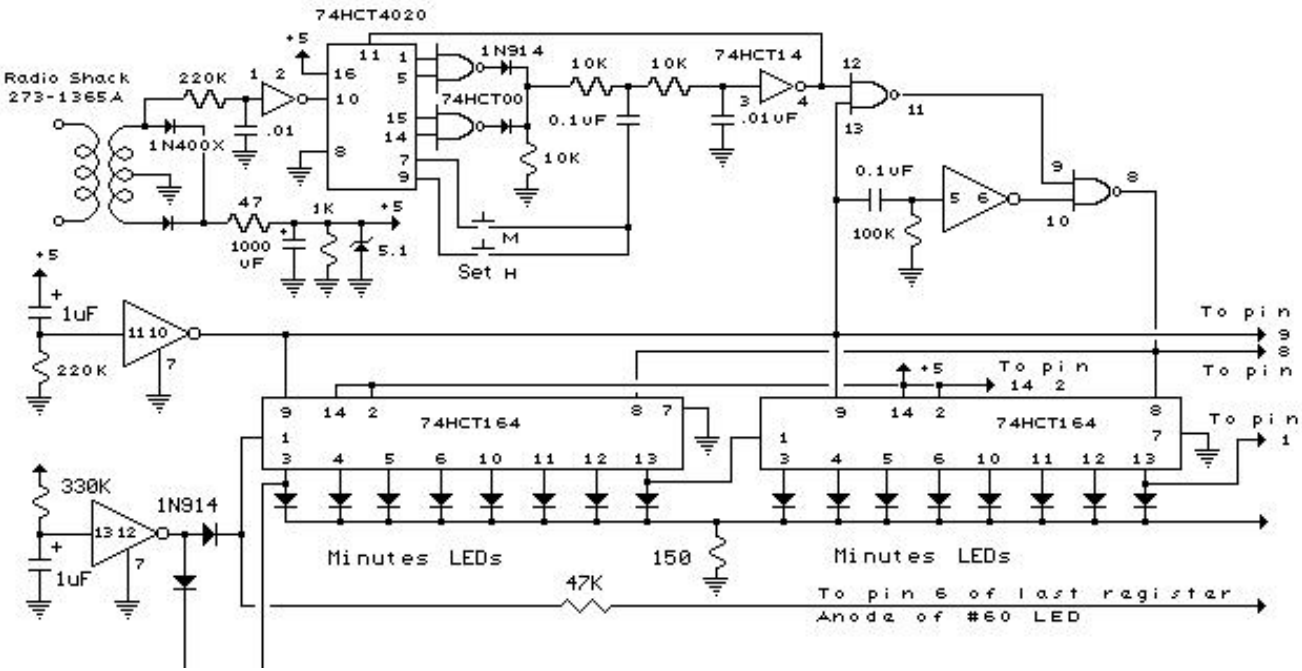


A Login Protocol



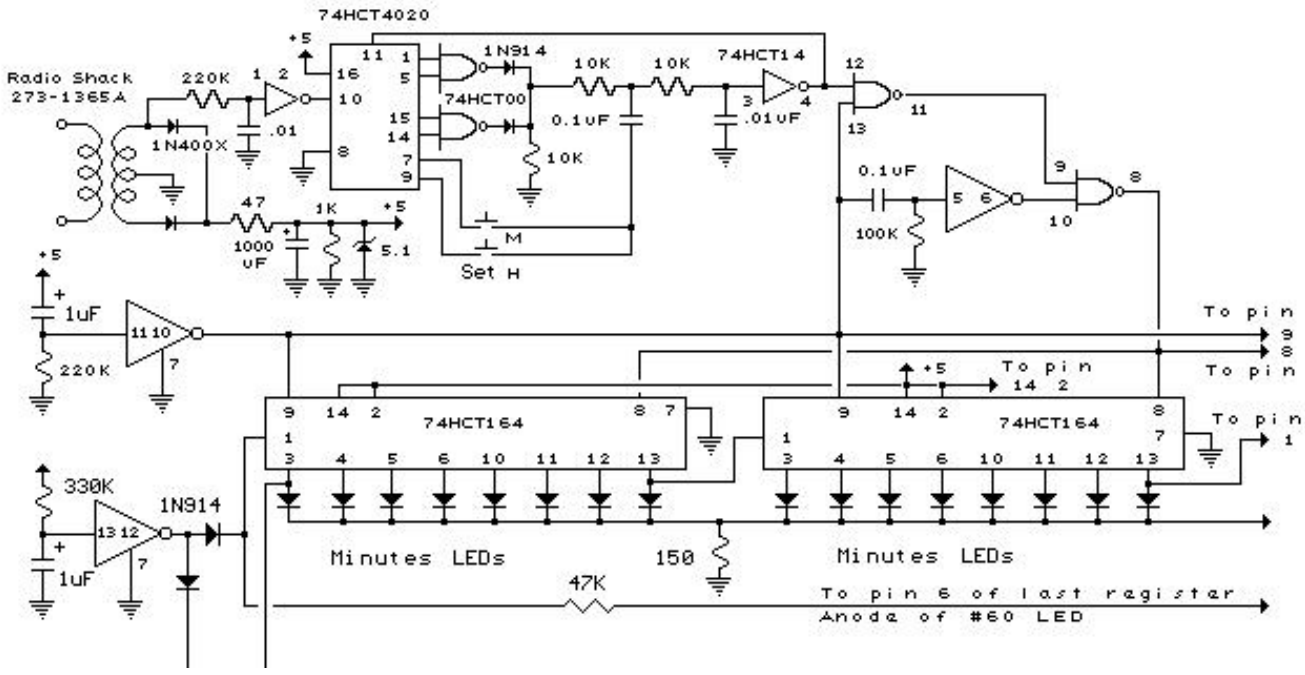
SPECIFICATION: $\Box(pw\_wrong \longrightarrow \Box_{[0,10)} \neg restart)$


74HCT4020
Radio Shack
273-1365A
220K
1N400X
.01
47
1000
uF
1K
5.1
+5
1N914
74HCT00
10K
0.1uF
74HCT14
.01uF
M
Set H
100K
1uF
220K
330K
74HCT164
Minutes LEDs
150
47K
To pin 9
To pin 8
To pin 14 2
To pin 1
To pin 6 of last register
Anode of #60 LED



[. . .] When power is applied, a single ‘1’ bit is loaded into the first stage of both the minutes and hours registers. To accomplish this, a momentary low reset signal is sent to all the registers (at pin 9) and also a NAND gate to lock out any clock transitions at pin 8 of the minutes registers. At the same time, a high level is applied to the data input lines of both minutes and hours registers at pin 1. A single positive going clock pulse is generated at the end of the reset signal which loads a high level into the first stage of the minutes register. The rising edge of first stage output at pin 3 advances the hours and a single bit is also loaded into the hours register. Power should remain off for 3 seconds before being re-applied to allow the filter and timing capacitors to discharge. [. . .]

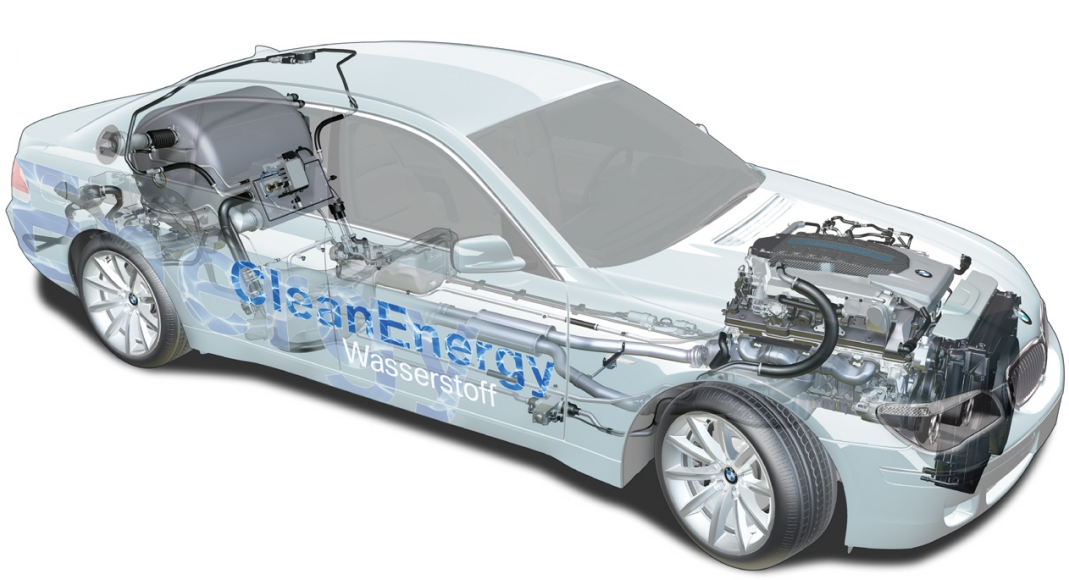
(Bill Bowden, `www.circuitdb.com/circuits/id/98`)



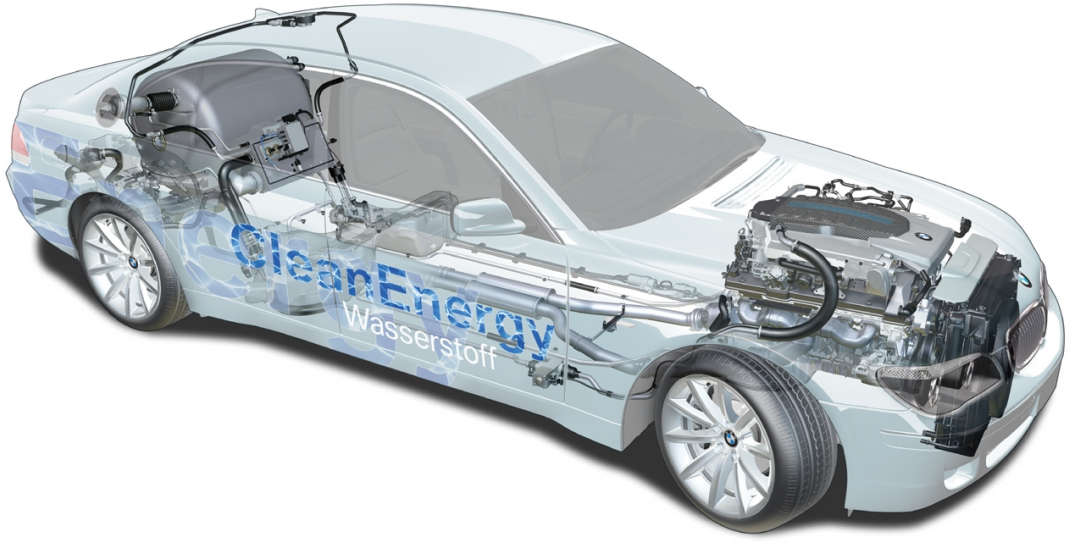
[. . .] When power is applied, a single ‘1’ bit is loaded into the first stage of both the minutes and hours registers. To accomplish this, a momentary low reset signal is sent to all the registers (at pin 9) and also a NAND gate to lock out any clock transitions at pin 8 of the minutes registers. At the same time, a high level is applied to the data input lines of both minutes and hours registers at pin 1. A single positive going clock pulse is generated at the end of the reset signal which loads a high level into the first stage of the minutes register. The rising edge of first stage output at pin 3 advances the hours and a single bit is also loaded into the hours register. Power should remain off for 3 seconds before being re-applied to allow the filter and timing capacitors to discharge. [. . .]

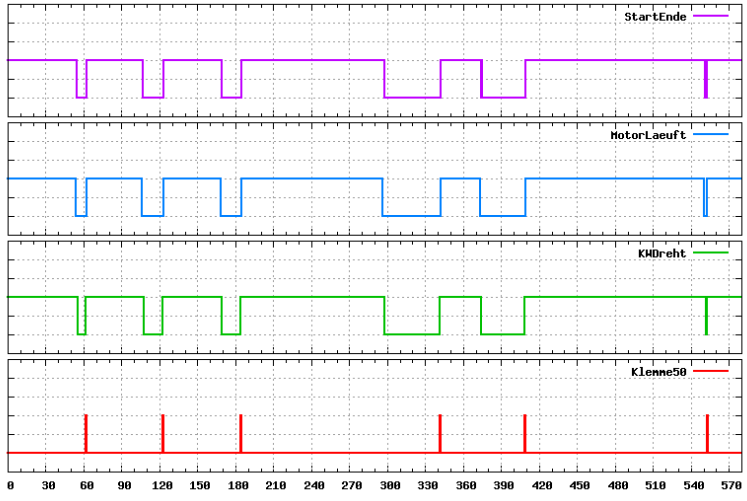
(Bill Bowden, `www.circuitdb.com/circuits/id/98`)

BMW Hydrogen 7



BMW Hydrogen 7





Timed Systems

Timed systems occur in:

- Hardware circuits
- Communication protocols
- Cell phones
- Plant controllers
- Aircraft navigation systems
- . . .

Timed Systems

Timed systems occur in:

- Hardware circuits
- Communication protocols
- Cell phones
- Plant controllers
- Aircraft navigation systems
- . . .

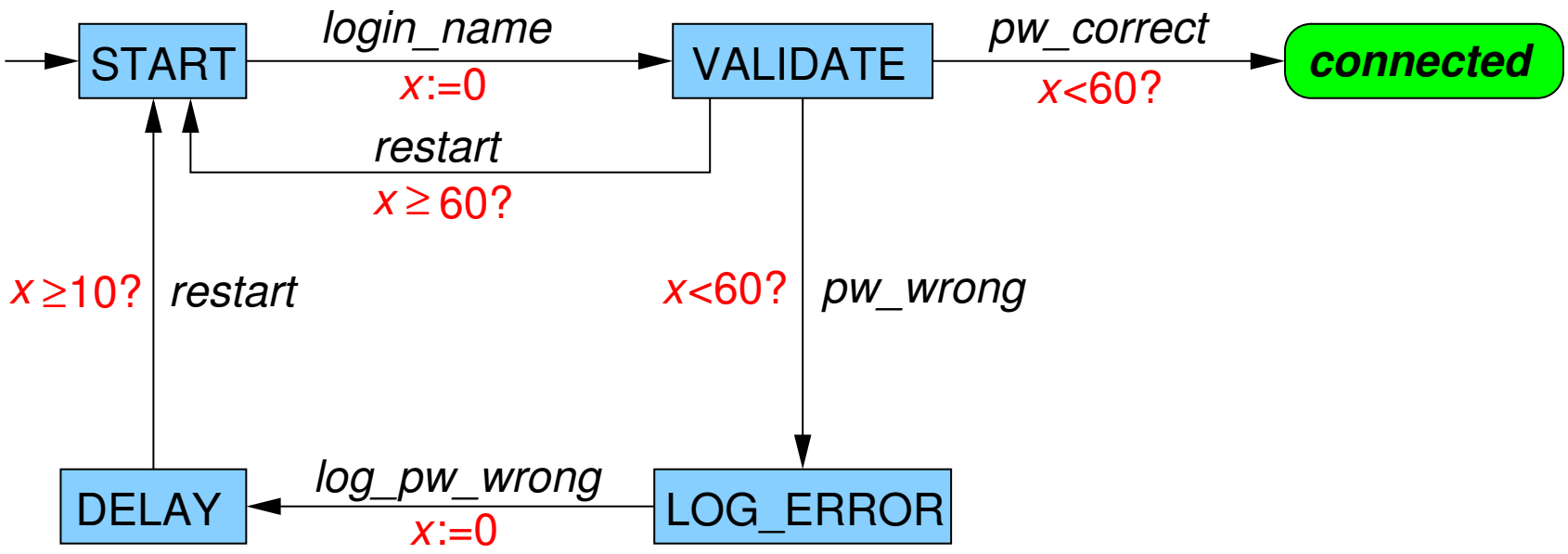
In many instances, it is **crucial** to accurately model the timed behaviour of the system.

From Qualitative to Quantitative

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

Boris Trakhtenbrot, LICS 1995

Timed Automata



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

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)

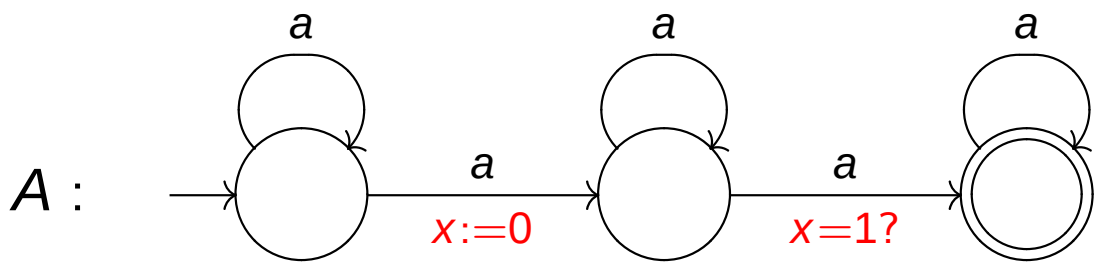
Reachability is decidable, in fact PSPACE-complete.

Unfortunately:

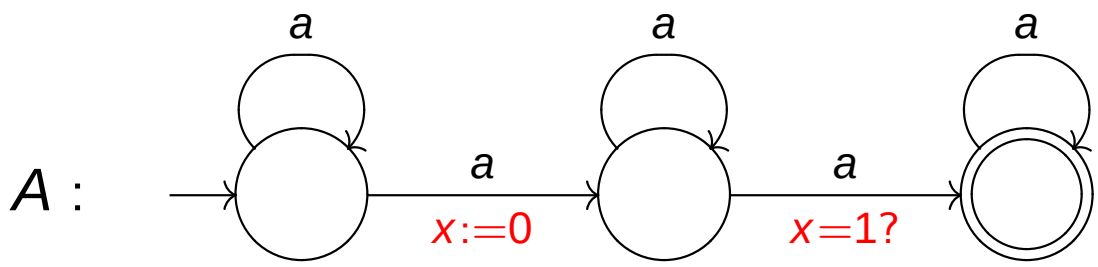
Theorem (Alur & Dill 1990)

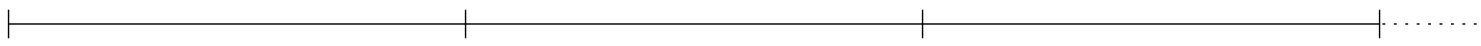
Language inclusion is undecidable for timed automata.

An Uncomplementable Timed Automaton

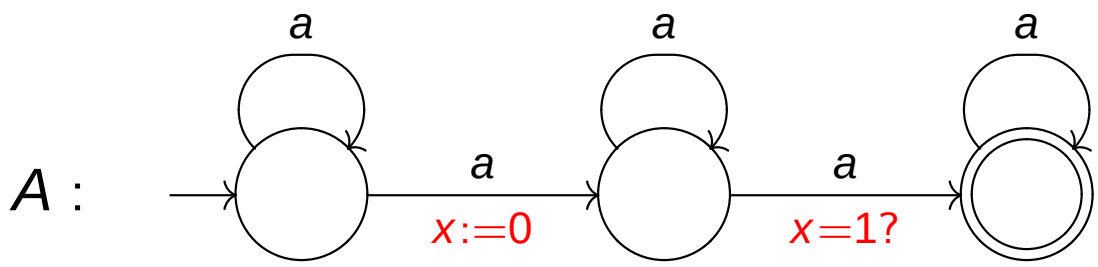


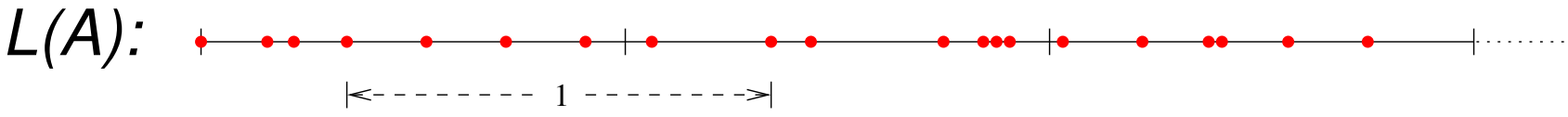
An Uncomplementable Timed Automaton



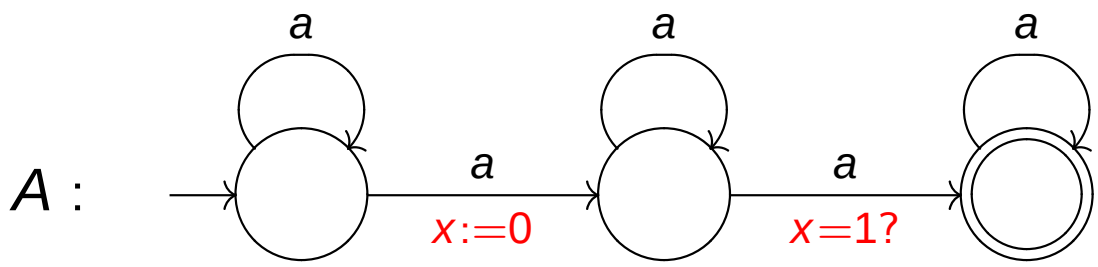


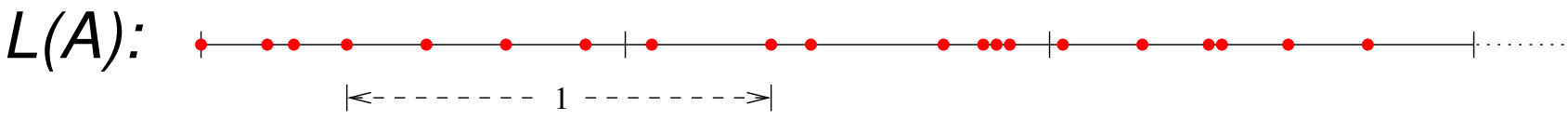
An Uncomplementable Timed Automaton

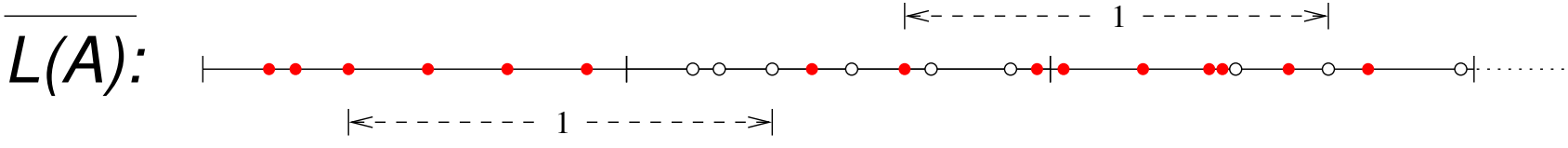




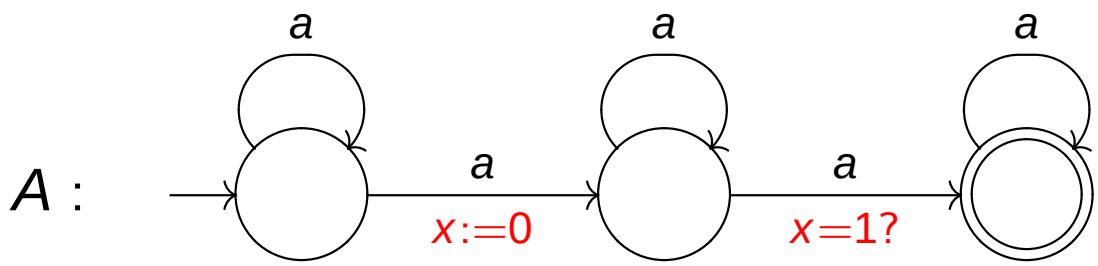
An Uncomplementable Timed Automaton

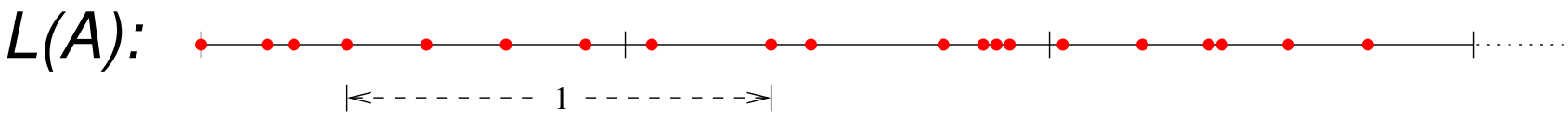


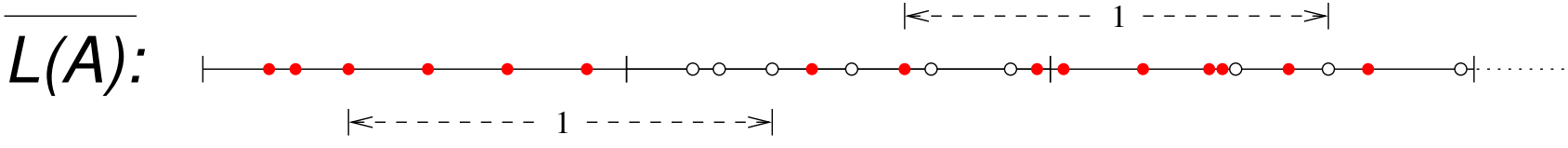




An Uncomplementable Timed Automaton







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:

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 seven hundred papers according to `scholar.google.com`!).

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 seven 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

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 seven 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 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

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

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 $\mathbb{R}_{\geq 0}$.

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

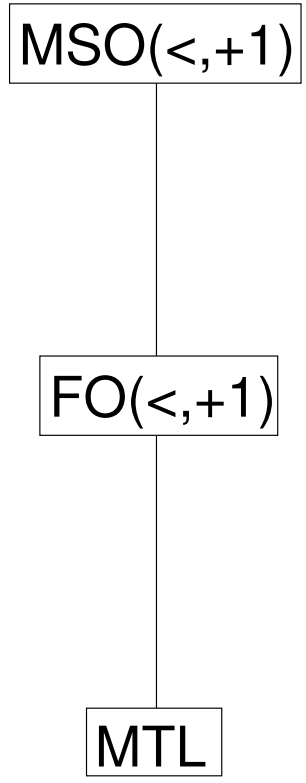
$$\forall x\,(\mathit{PEDAL}(x) \rightarrow \exists y\,(x + 5 \leq y \leq x + 10 \wedge \mathit{BRAKE}(y)))$$

Theorem (Hirshfeld & Rabinovich 2007)

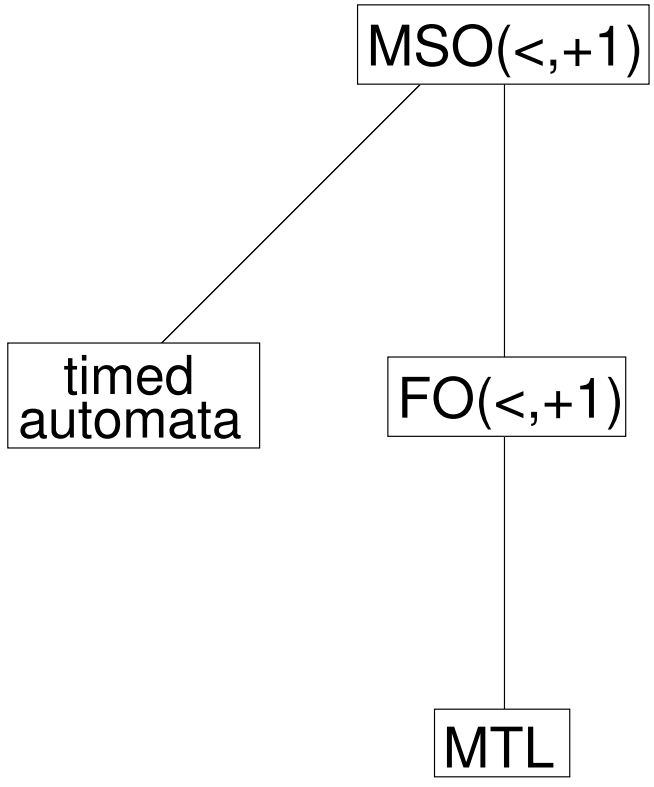
FO($<$,$+1$) is strictly more expressive than MTL over $\mathbb{R}_{\geq 0}$.

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

The Real-Time Theory: Expressiveness



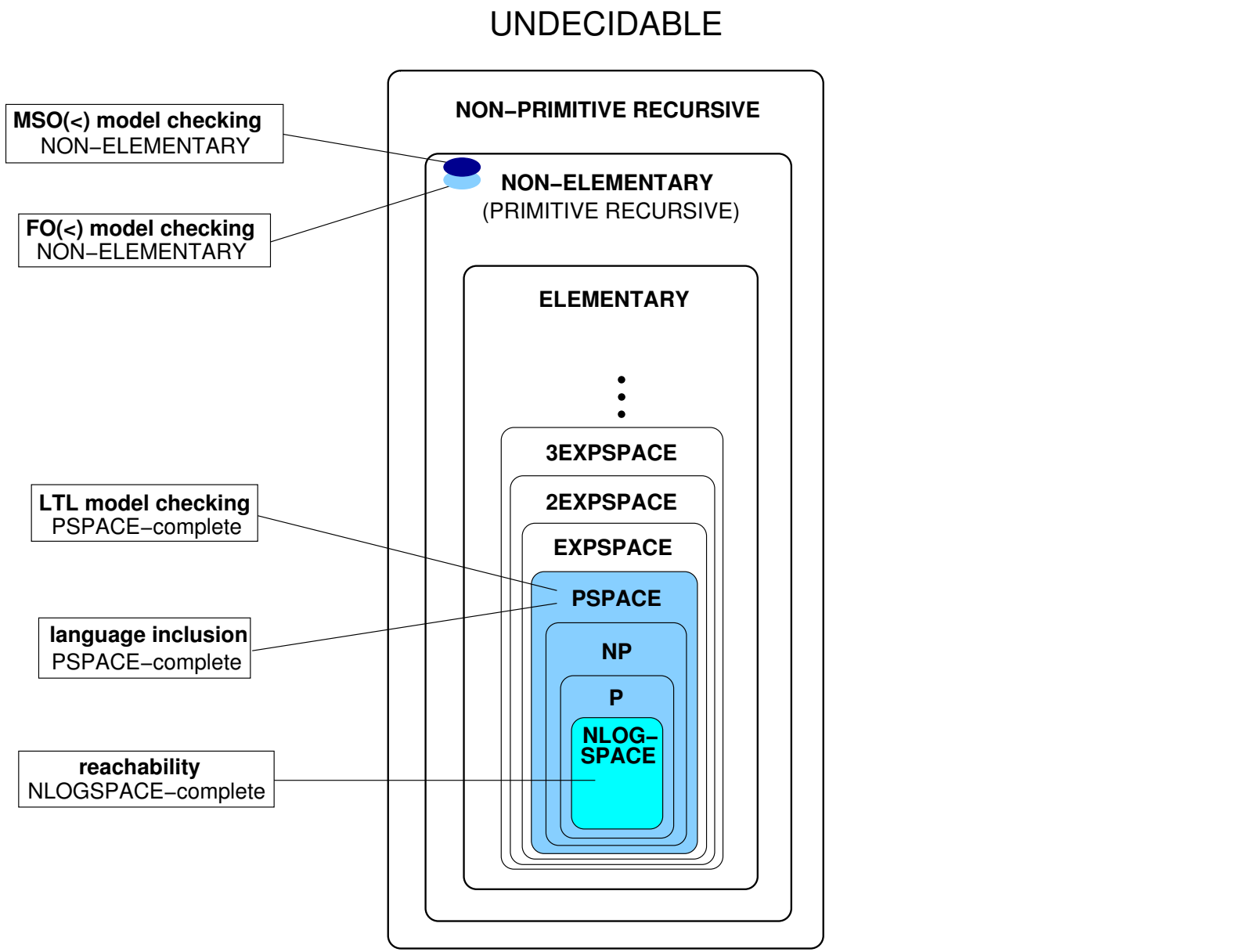
The Real-Time Theory: Expressiveness



The Real-Time Theory: Complexity

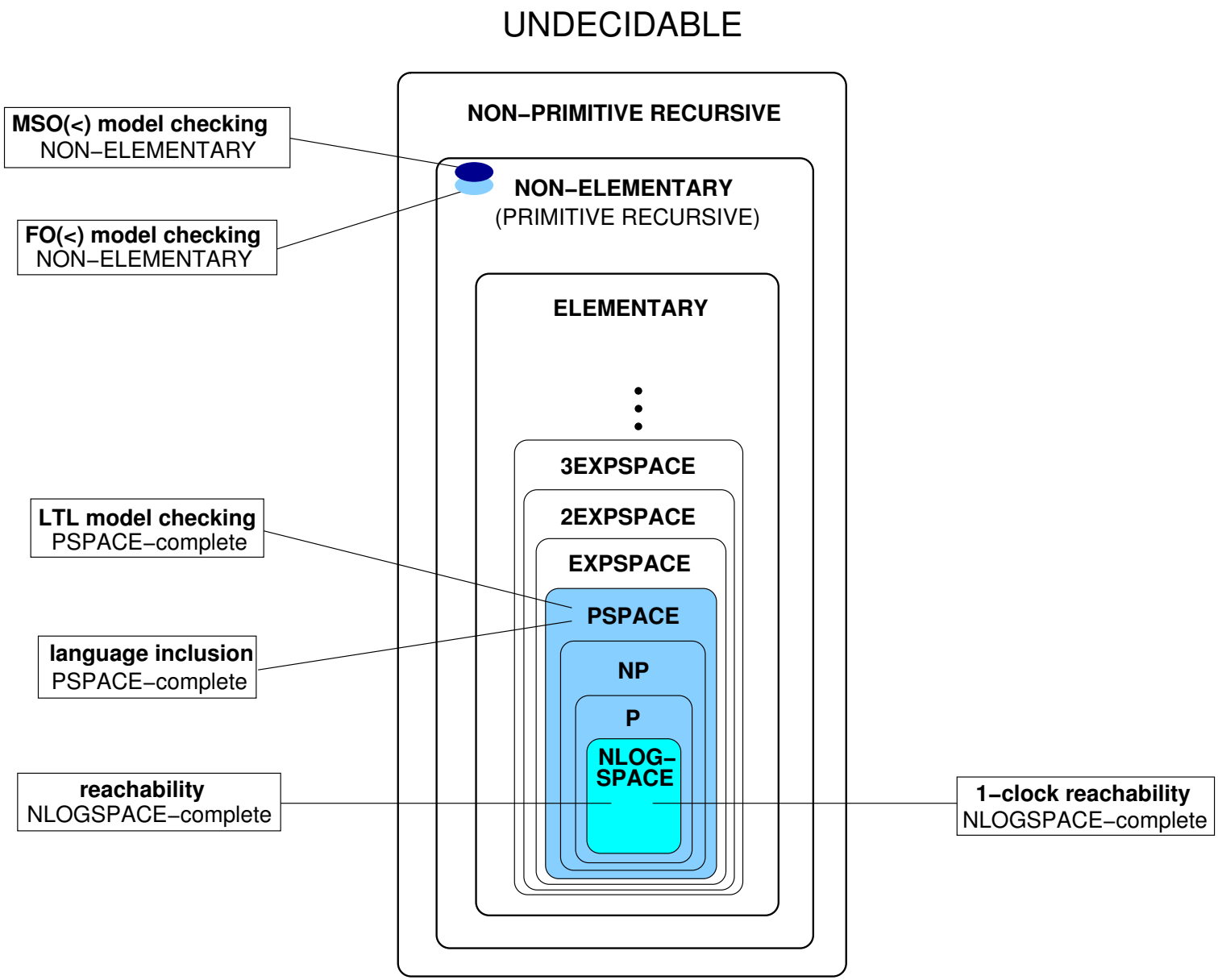
Classical Theory

Real–Time Theory



The Real-Time Theory: Complexity

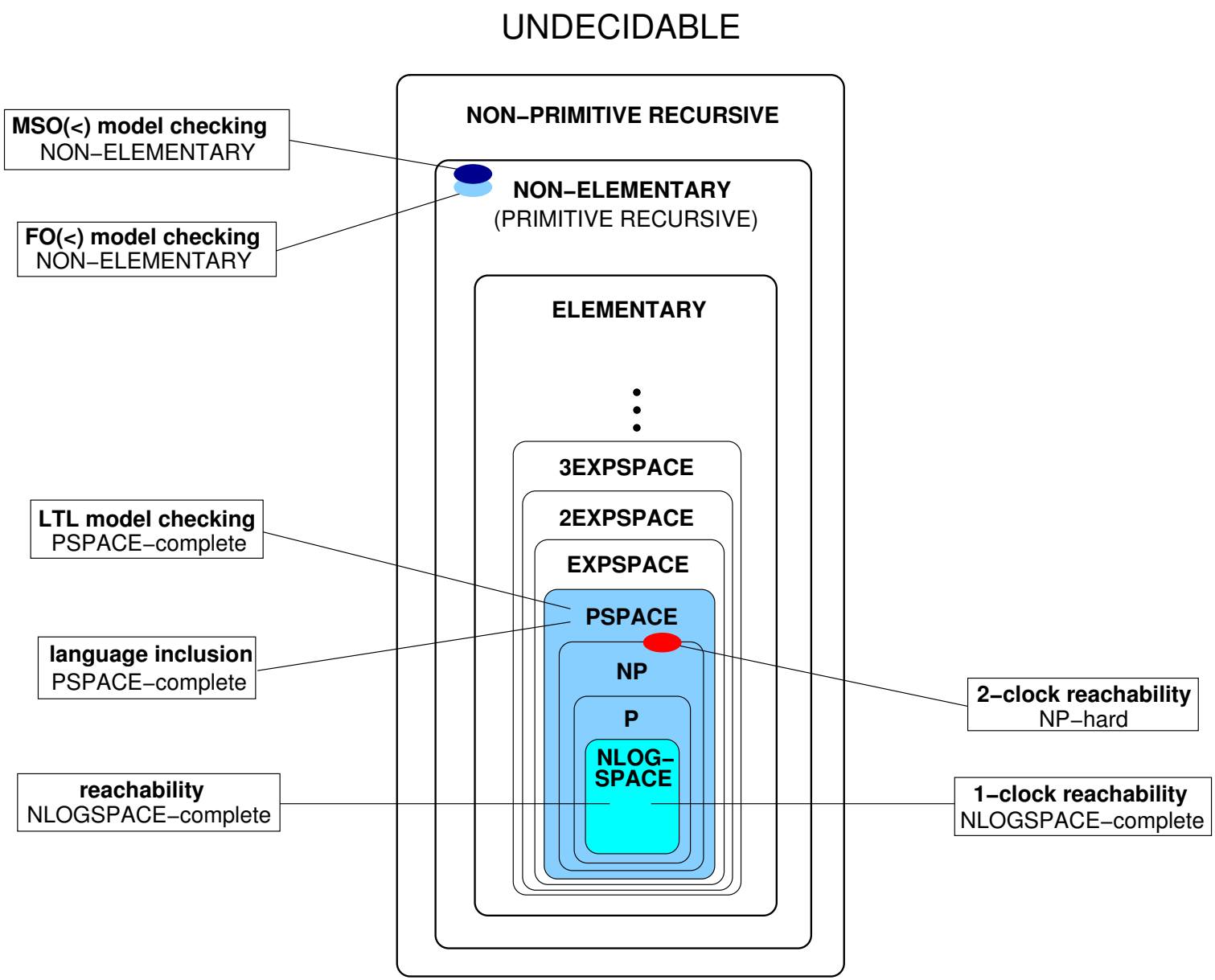
Classical Theory **Real–Time Theory**



The Real-Time Theory: Complexity

Classical Theory

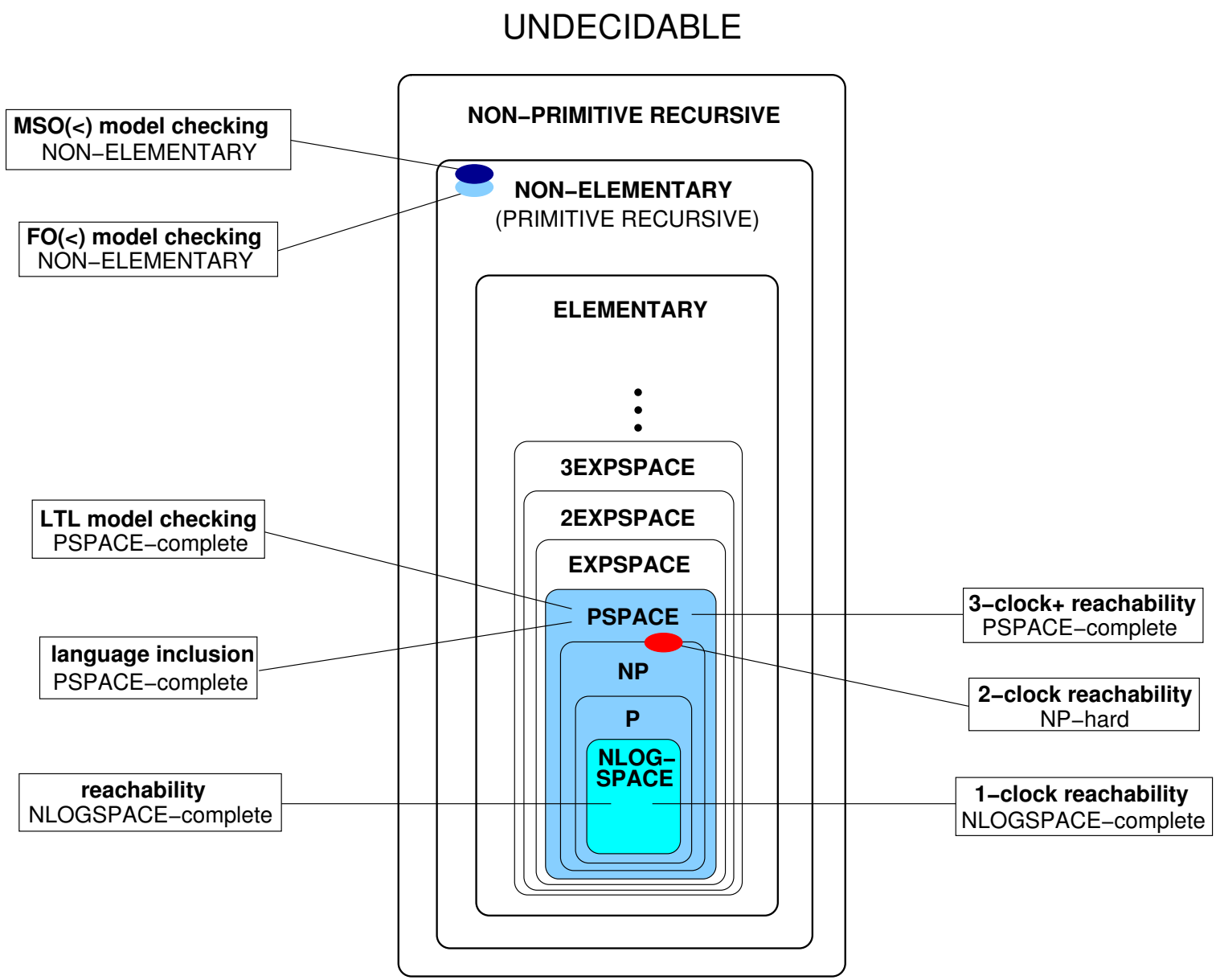
Real–Time Theory



The Real-Time Theory: Complexity

Classical Theory

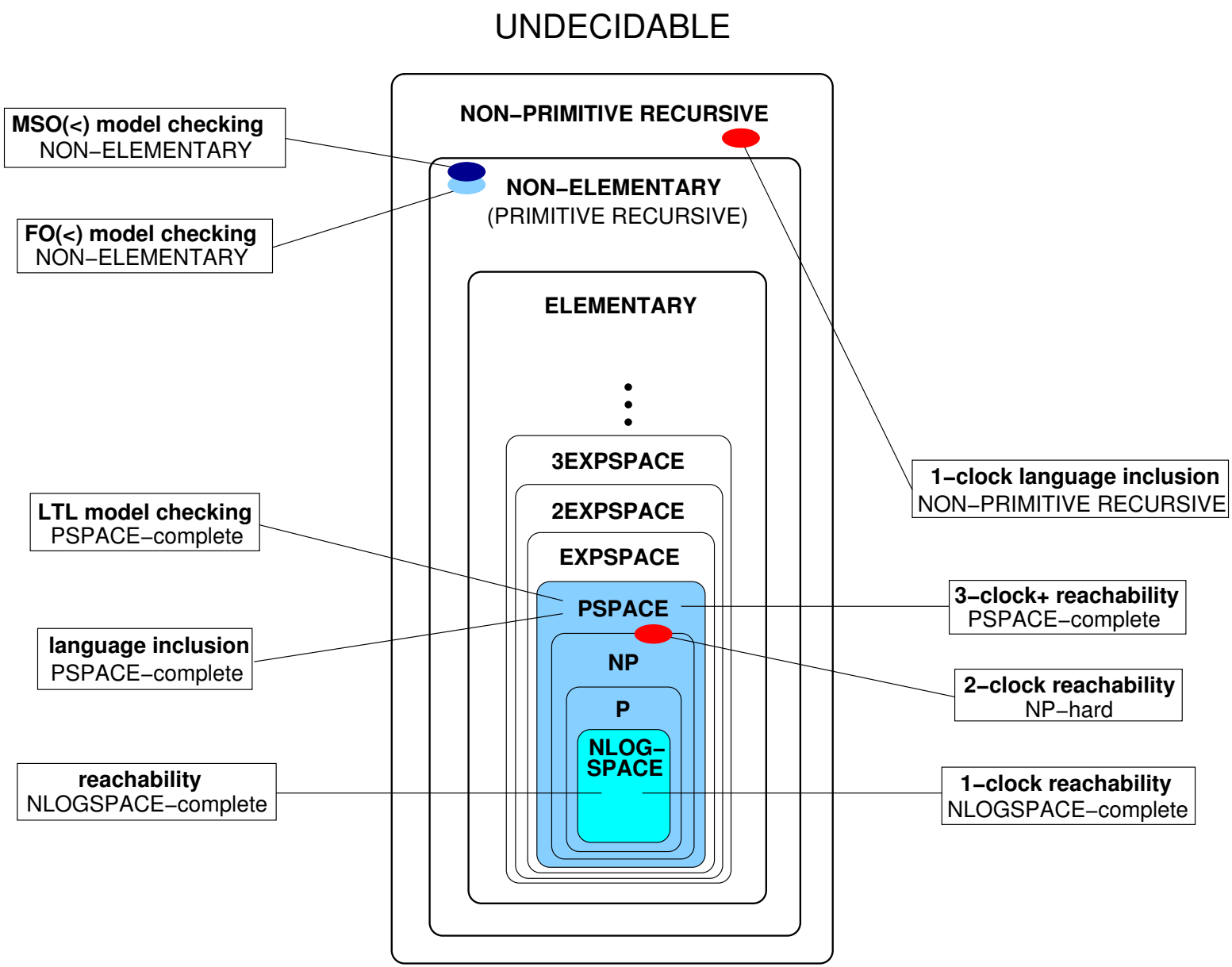
Real–Time Theory



The Real-Time Theory: Complexity

Classical Theory

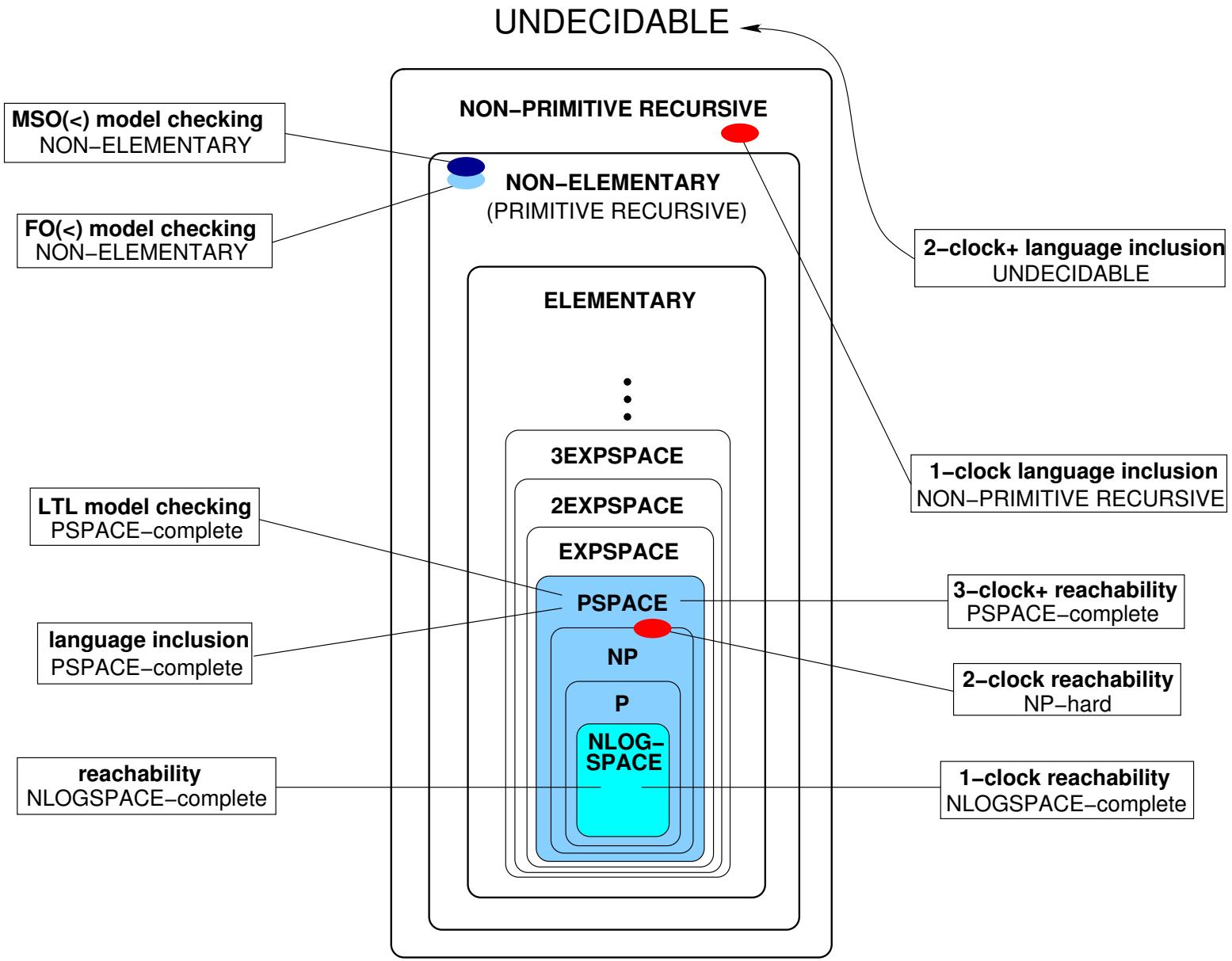
Real–Time Theory



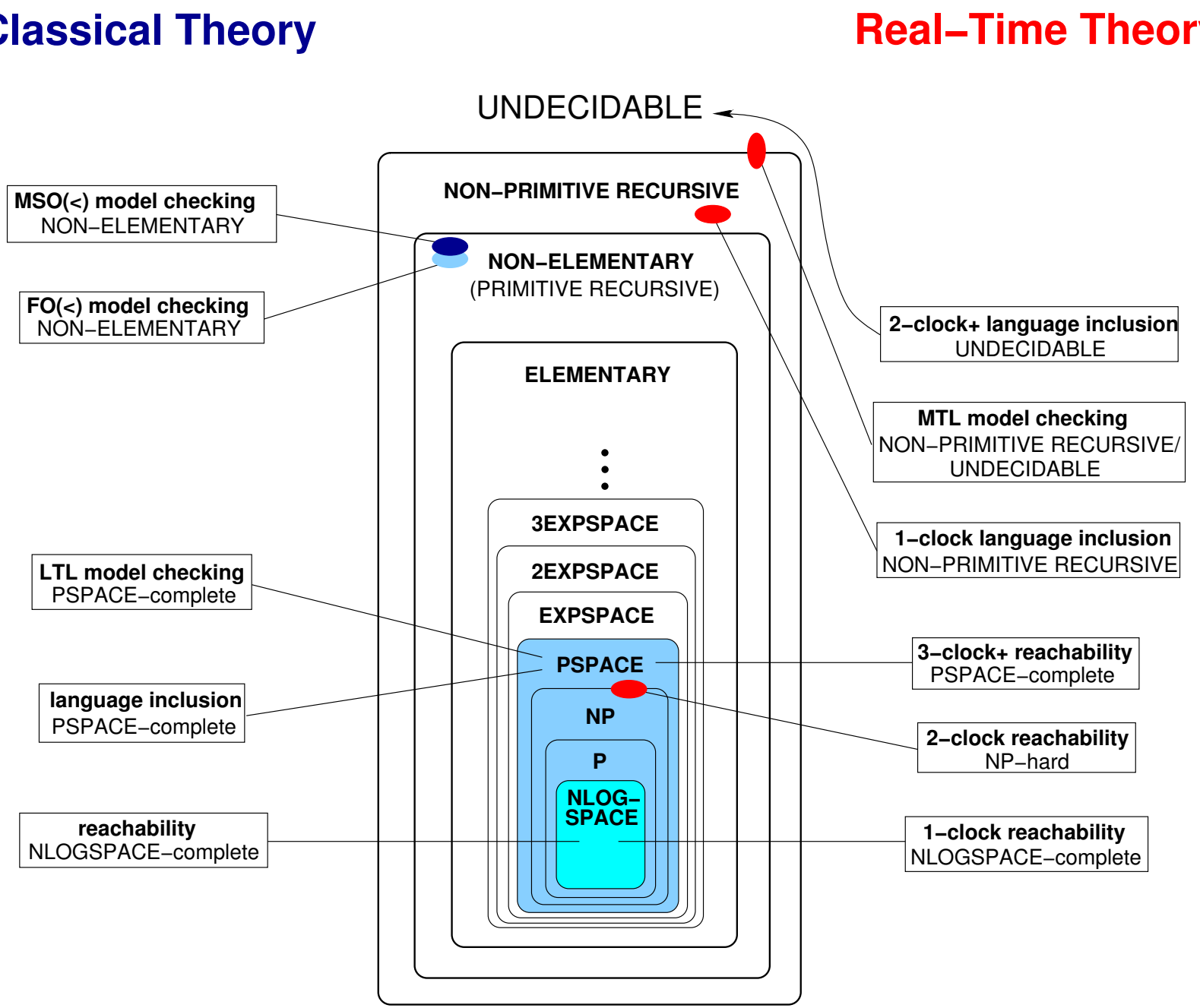
The Real-Time Theory: Complexity

Classical Theory

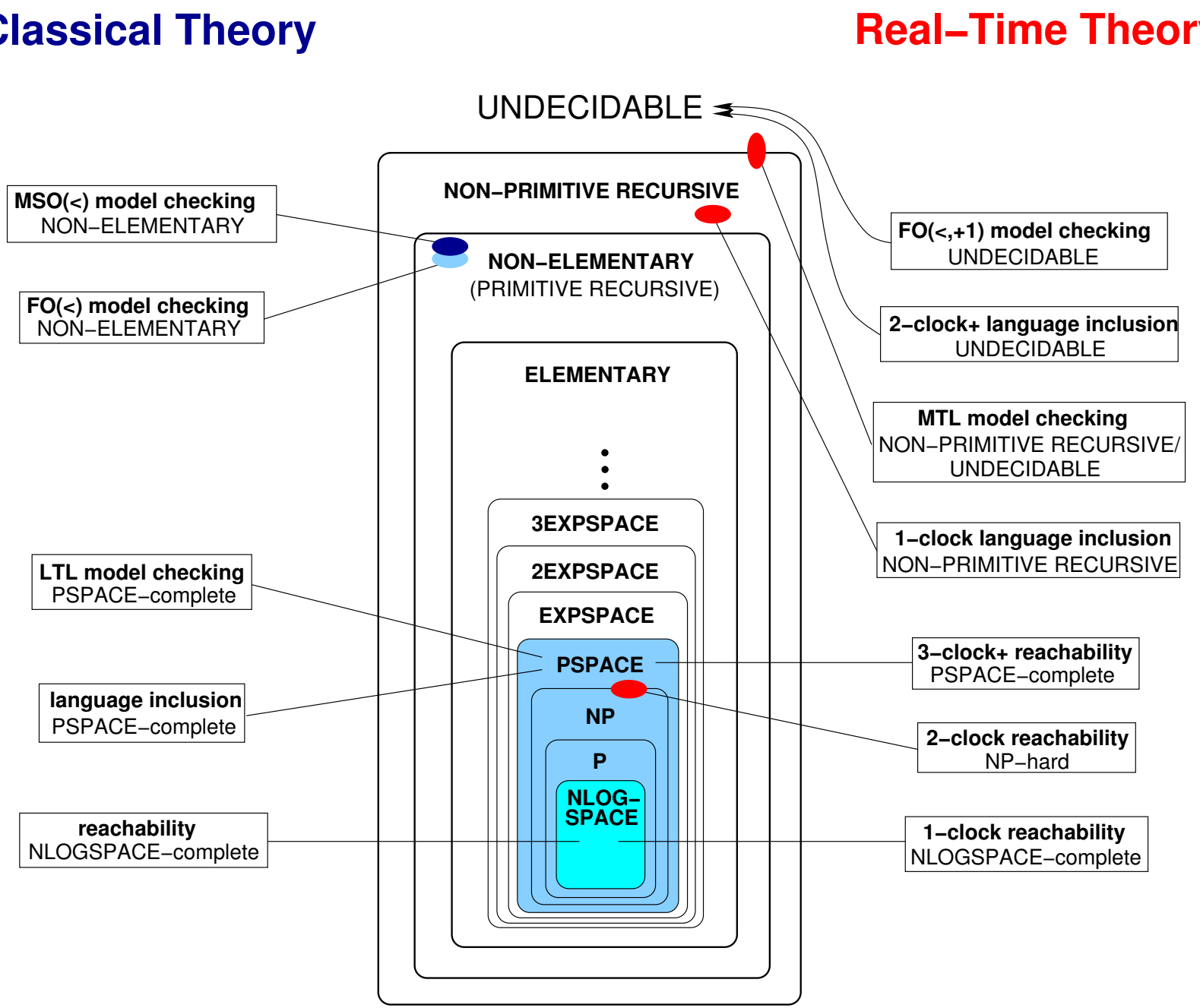
Real–Time Theory



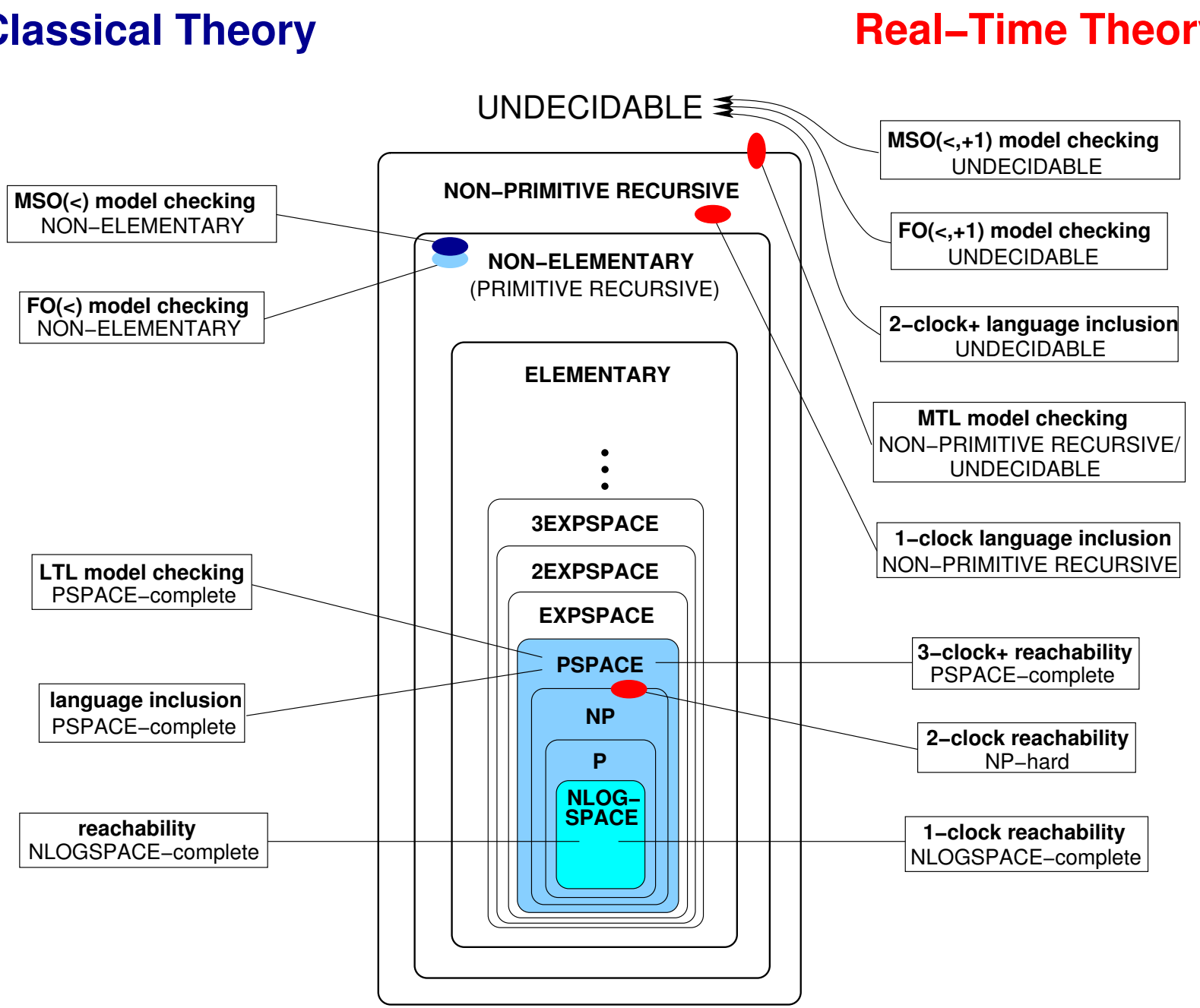
The Real-Time Theory: Complexity



The Real-Time Theory: Complexity



The Real-Time Theory: Complexity



Key Stumbling Block

Theorem (Alur & Dill 1990)

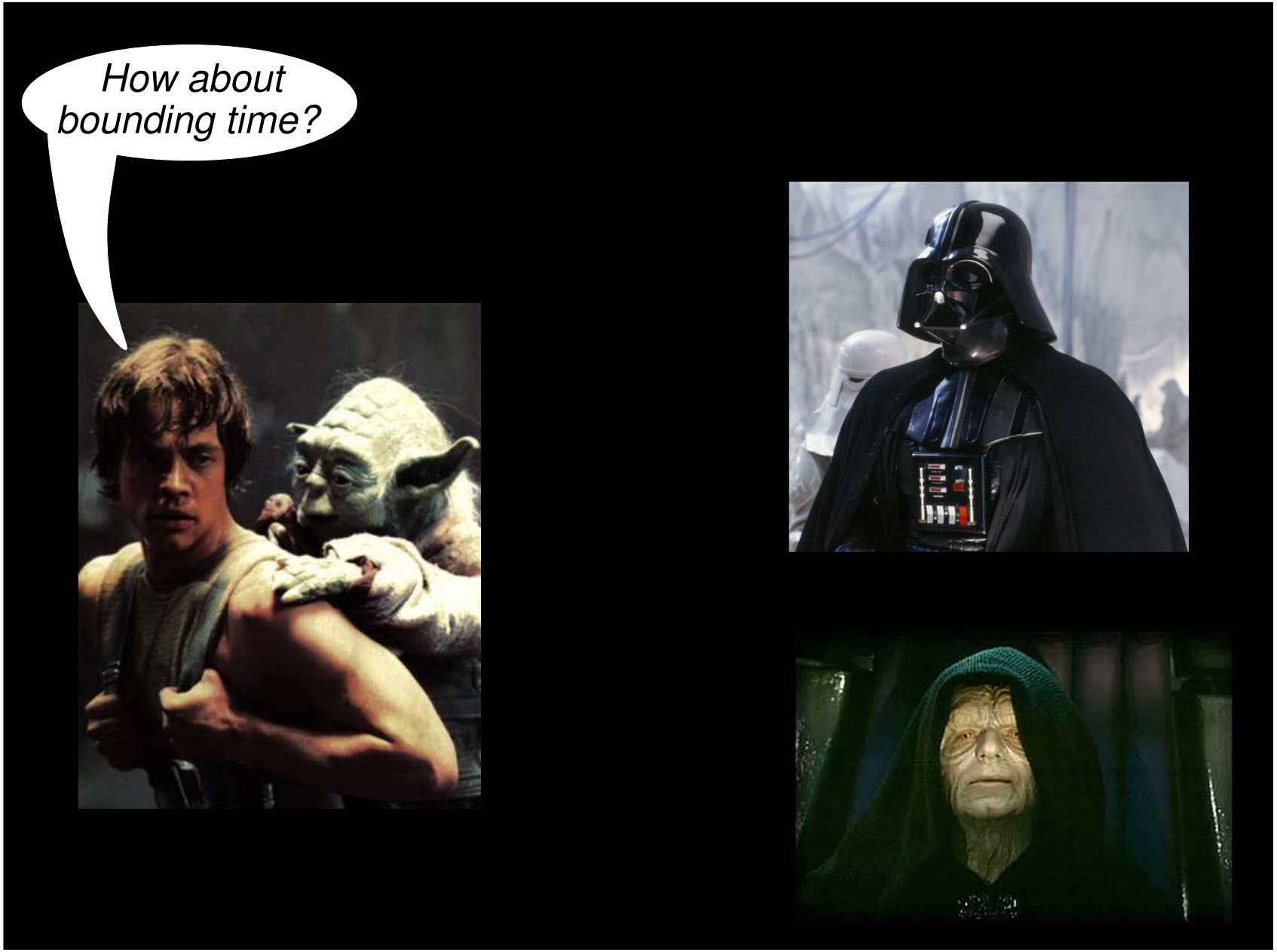
Language inclusion is undecidable for timed automata.

Timed Language Inclusion: Some Related Work

- Topological restrictions and digitization techniques:
  [Henzinger, Manna, Pnueli 1992], [Bošnački 1999],
  [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]

A Long Time Ago, circa 2003. . .

A Long Time Ago, circa 2003. . .



A Long Time Ago, circa 2003. . .

How about bounding time?

Timed automata still cannot be complemented!

A Long Time Ago, circa 2003. . .

A Long Time Ago, circa 2003. . .

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.

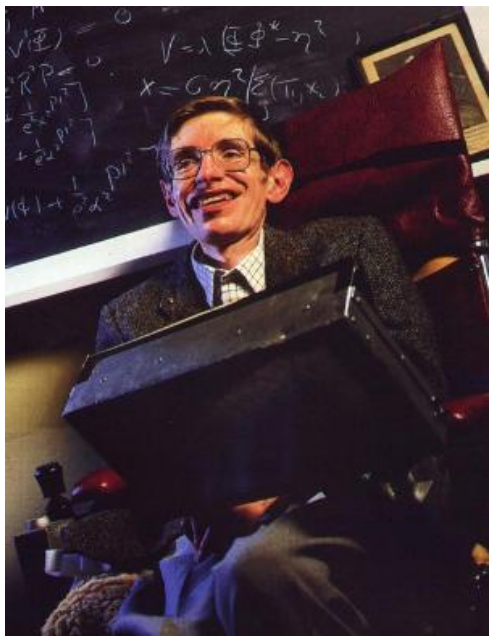
Time-Bounded Language Inclusion

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:

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($<$,$+1$)
  (in fact, even in $\exists$MTL [Henzinger, Raskin, Schobbens 1998]).

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($<$,$+1$)
  (in fact, even in $\exists$MTL [Henzinger, Raskin, Schobbens 1998]).
- This reverses Vardi's 'automata-theoretic approach to verification' paradigm!

Monadic Second-Order Logic

More problems:

Monadic Second-Order Logic

More problems:

Theorem (Shelah 1975)

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

Monadic Second-Order Logic

More problems:

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):

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

P:

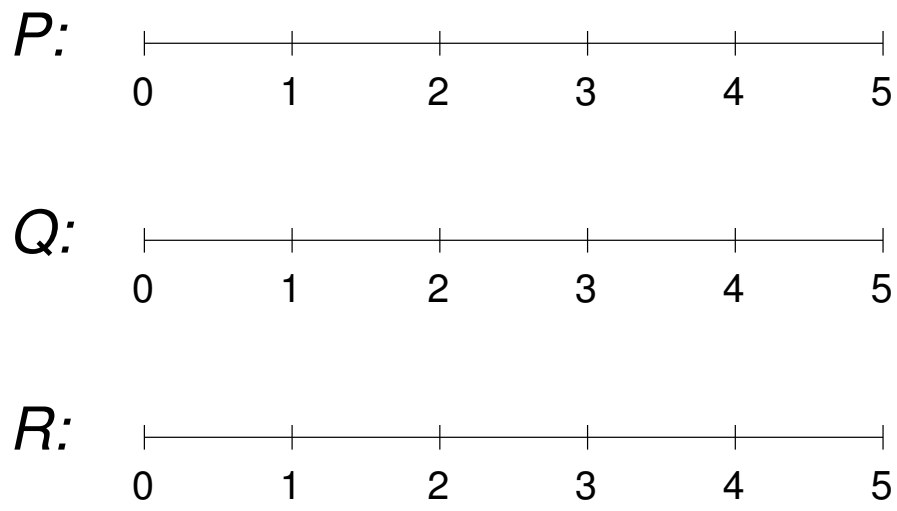
Q:

R:

Finite Variability

Timed behaviours are modelled as flows (or signals):

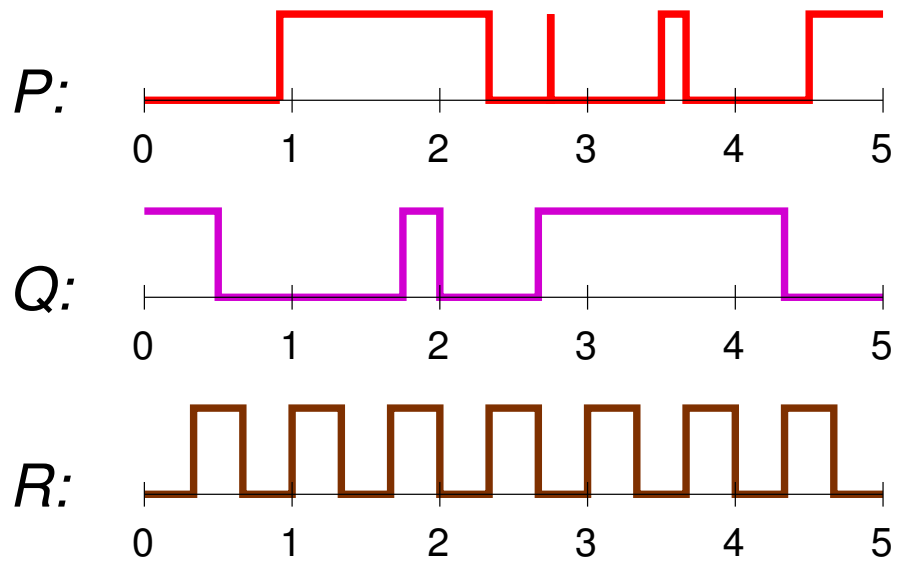
$f : [0, T) \longrightarrow 2^{\mathbf{MP}}$



Finite Variability

Timed behaviours are modelled as flows (or signals):

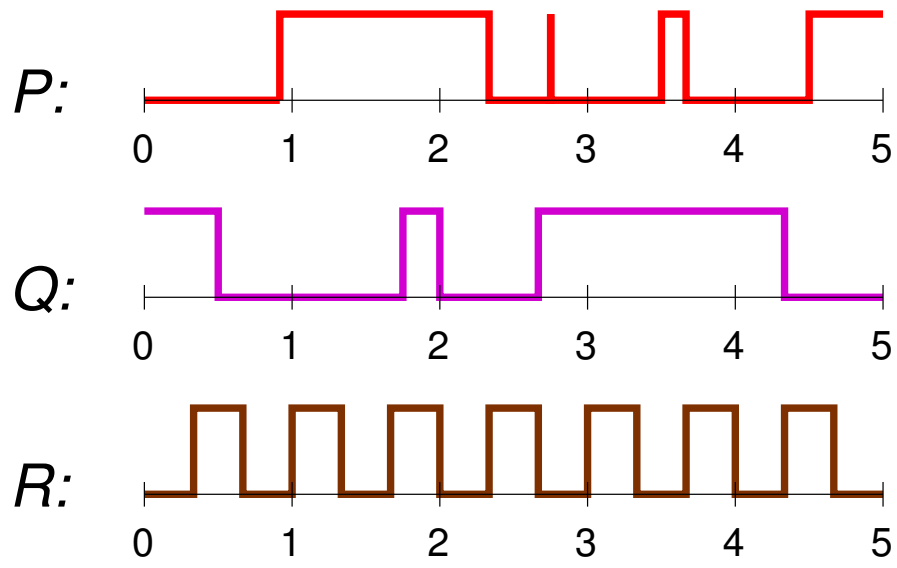
$f : [0, T) \longrightarrow 2^{\mathbf{MP}}$



Finite Variability

Timed behaviours are modelled as flows (or signals):

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

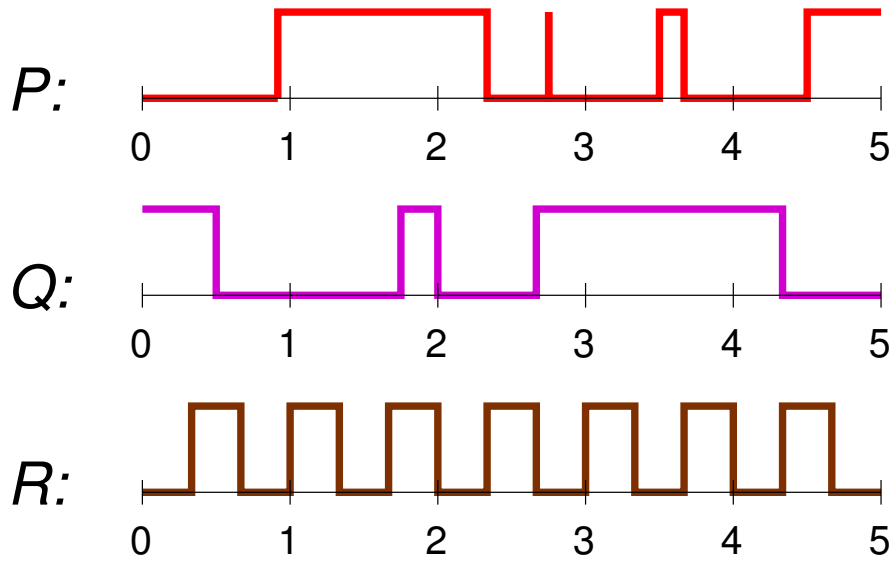


Predicates must have finite variability:

Finite Variability

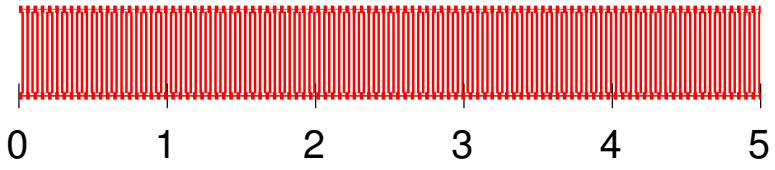
Timed behaviours are modelled as flows (or signals):

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



Predicates must have finite variability:

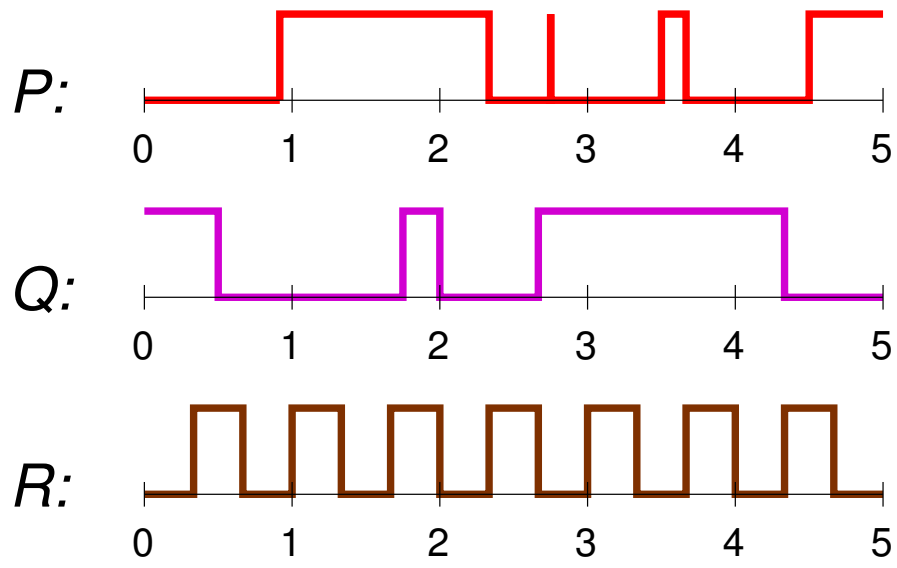
Disallow e.g. $\mathbb{Q}$:



Finite Variability

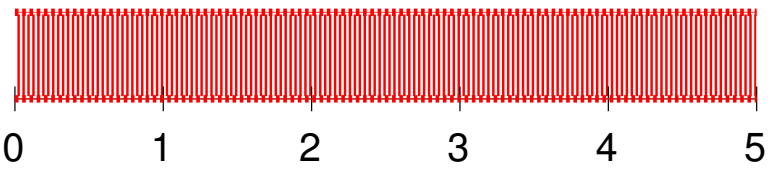
Timed behaviours are modelled as flows (or signals):

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



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 fixed bounded time domain $[0, T)$*, the satisfiability and model-checking problems for MSO(*$<$*,*$+1$*), FO(*$<$*,*$+1$*), and MTL are all decidable, with the following complexities:*

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

The Time-Bounded Theory of Verification

Theorem

For any fixed bounded time domain $[0, T)$*, the satisfiability and model-checking problems for MSO*$(<,+1)$*, FO*$(<,+1)$*, and MTL are all decidable, with the following complexities:*

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 fixed bounded time domain $[0, T)$*, the satisfiability and model-checking problems for MSO*$(<,+1)$*, FO*$(<,+1)$*, and MTL are all decidable, with the following complexities:*

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 language inclusion problem* $L_T(A) \subseteq L_T(B)$ *is decidable and 2EXPSPACE-complete.*

Time-Bounded Language Inclusion

- Let timed automata A, B, and time bound T be given.

Time-Bounded Language Inclusion

- Let timed automata A, B, and time bound T be given.
- Define formula $\varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P})$ in MSO($<$,$+1$) such that:

$$A \text{ accepts timed word } w \iff \varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P}) \text{ holds}$$

Time-Bounded Language Inclusion

- Let timed automata A, B, and time bound T be given.
- Define formula $\varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P})$ in MSO($<$,$+1$) such that:

$$A \text{ accepts timed word } w \iff \varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P}) \text{ holds}$$

where

- $\mathbf{W}$ encodes w
- $\mathbf{P}$ encodes a corresponding run of A.

Time-Bounded Language Inclusion

- Let timed automata A, B, and time bound T be given.
- Define formula $\varphi_A^{\text{acc}}(\mathbf{W}, \mathbf{P})$ in MSO($<$,$+1$) such that:

$$A \text{ accepts timed word } w \iff \varphi_A^{\text{acc}}(\mathbf{W}, \mathbf{P}) \text{ holds}$$

  where
  - $\mathbf{W}$ encodes w
  - $\mathbf{P}$ encodes a corresponding run of A.
- Define likewise $\varphi_B^{\text{acc}}(\mathbf{W}, \mathbf{Q})$ for timed automaton B.

Time-Bounded Language Inclusion

- Let timed automata A, B, and time bound T be given.
- Define formula $\varphi_A^{\text{acc}}(\mathbf{W}, \mathbf{P})$ in MSO(<,+1) such that:

$$A \text{ accepts timed word } w \iff \varphi_A^{\text{acc}}(\mathbf{W}, \mathbf{P}) \text{ holds}$$

  where
  - $\mathbf{W}$ encodes w
  - $\mathbf{P}$ encodes a corresponding run of A.
- Define likewise $\varphi_B^{\text{acc}}(\mathbf{W}, \mathbf{Q})$ for timed automaton B.
- Then $L_T(A) \subseteq L_T(B)$ iff:

$$\forall \mathbf{W} \forall \mathbf{P} \; (\varphi_A^{\text{acc}}(\mathbf{W}, \mathbf{P}) \rightarrow \exists \mathbf{Q} \, \varphi_B^{\text{acc}}(\mathbf{W}, \mathbf{Q}))$$

  holds over time domain $[0, T)$.

Time-Bounded Language Inclusion

- Let timed automata A, B, and time bound T be given.
- Define formula $\varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P})$ in MSO(<,+1) such that:

$$A \text{ accepts timed word } w \iff \varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P}) \text{ holds}$$

  where
  - $\mathbf{W}$ encodes w
  - $\mathbf{P}$ encodes a corresponding run of A.
- Define likewise $\varphi_B^{\mathrm{acc}}(\mathbf{W}, \mathbf{Q})$ for timed automaton B.
- Then $L_T(A) \subseteq L_T(B)$ iff:

$$\forall \mathbf{W} \forall \mathbf{P} \; (\varphi_A^{\mathrm{acc}}(\mathbf{W}, \mathbf{P}) \rightarrow \exists \mathbf{Q} \, \varphi_B^{\mathrm{acc}}(\mathbf{W}, \mathbf{Q}))$$

  holds over time domain $[0, T)$.
- This can be decided in 2EXPSPACE.

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

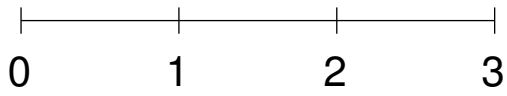
Key idea: eliminate the metric by 'vertical stacking'.

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

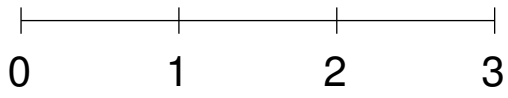
  φ is satisfiable over $[0, T)$ $\iff$ $\overline{\varphi}$ is satisfiable over $[0, 1)$

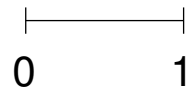
- Conclude by invoking decidability of MSO($<$).

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

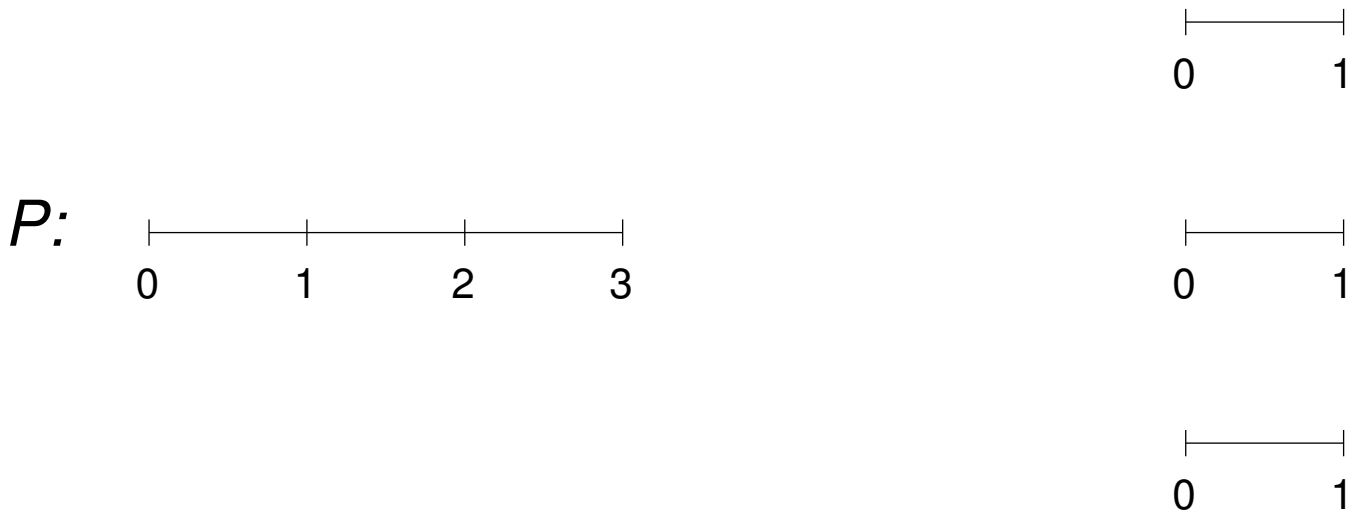


From MSO($<$,$+1$) to MSO($<$)



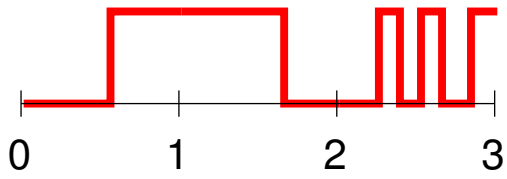


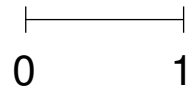
From MSO(<,+1) to MSO(<)



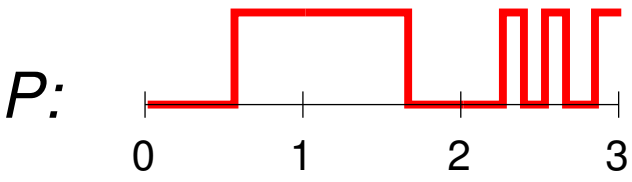
From MSO(<,+1) to MSO(<)

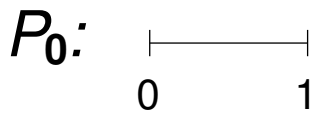
P:

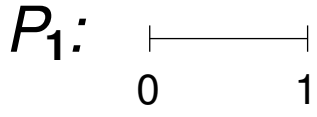


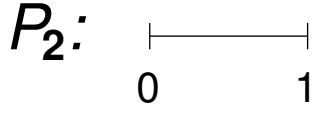


From MSO(<,+1) to MSO(<)

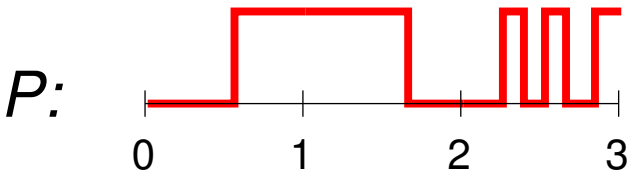


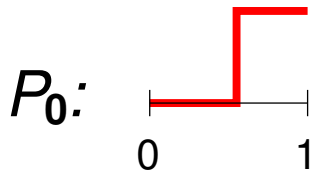


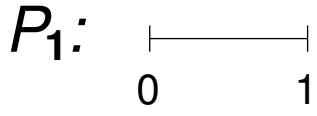


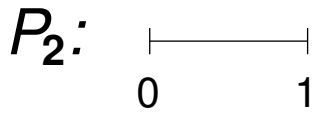


From MSO(<,+1) to MSO(<)

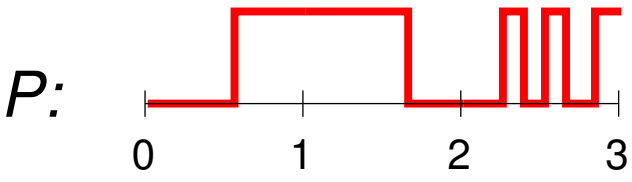


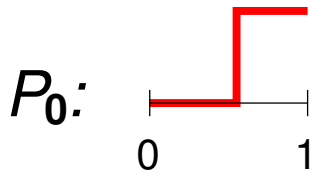


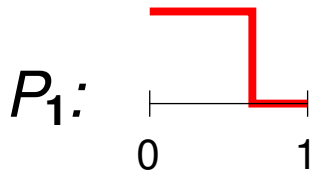


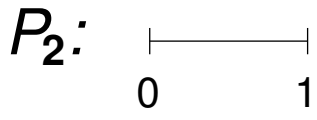


From MSO(<,+1) to MSO(<)

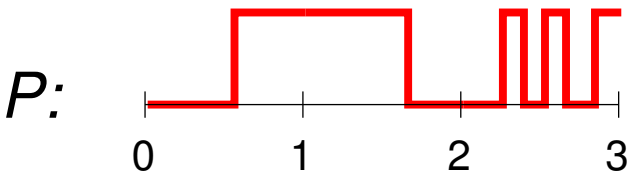


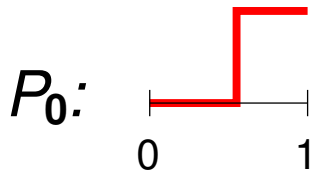


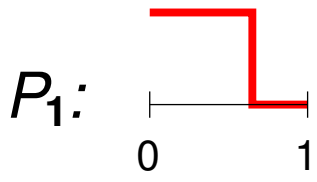


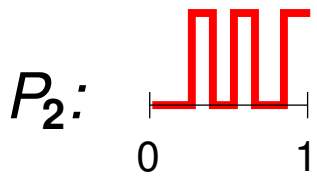


From MSO(<,+1) to MSO(<)

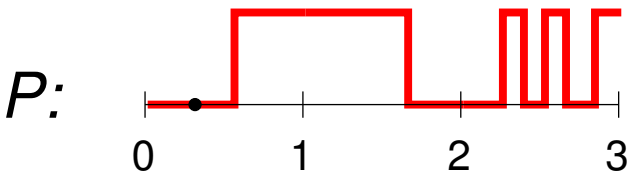


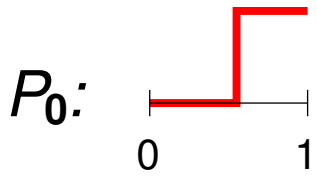


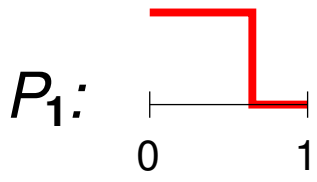


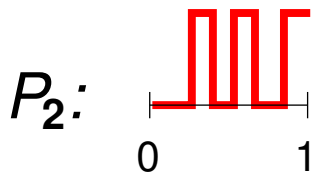


From MSO(<,+1) to MSO(<)

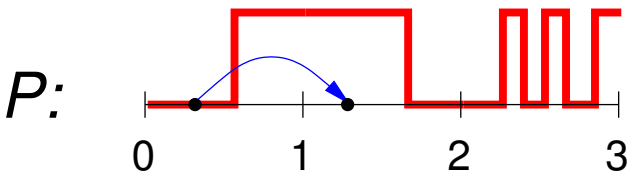


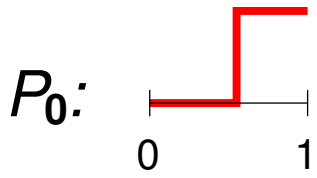


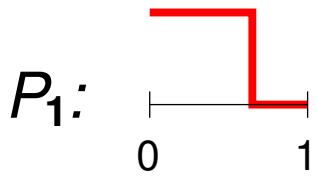


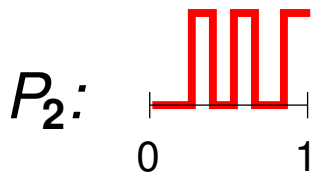


From MSO(<,+1) to MSO(<)

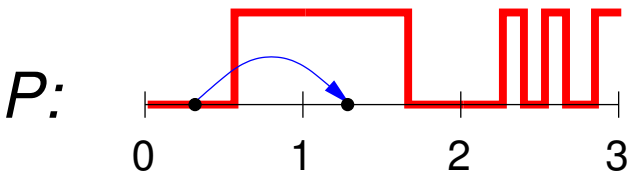


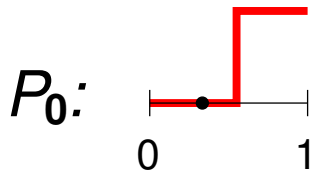


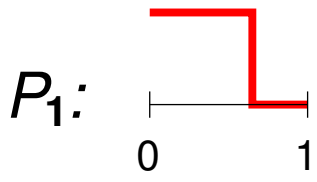


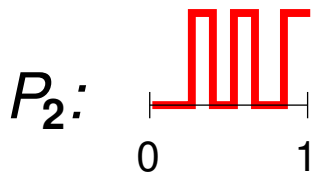


From MSO(<,+1) to MSO(<)

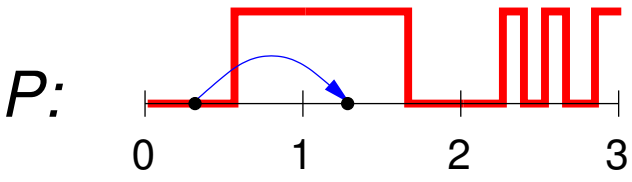


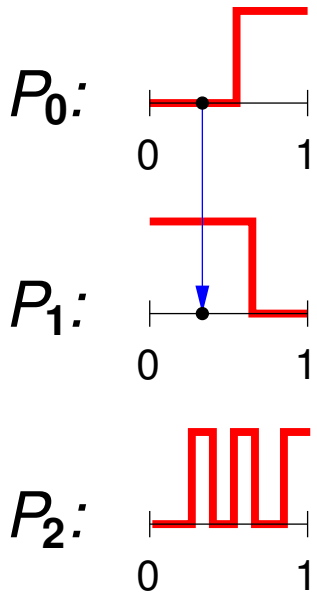




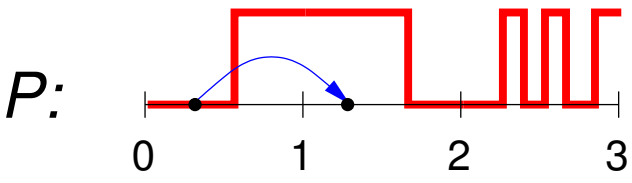


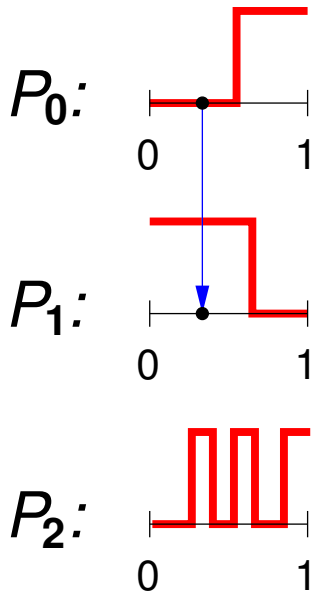
From MSO(<,+1) to MSO(<)





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

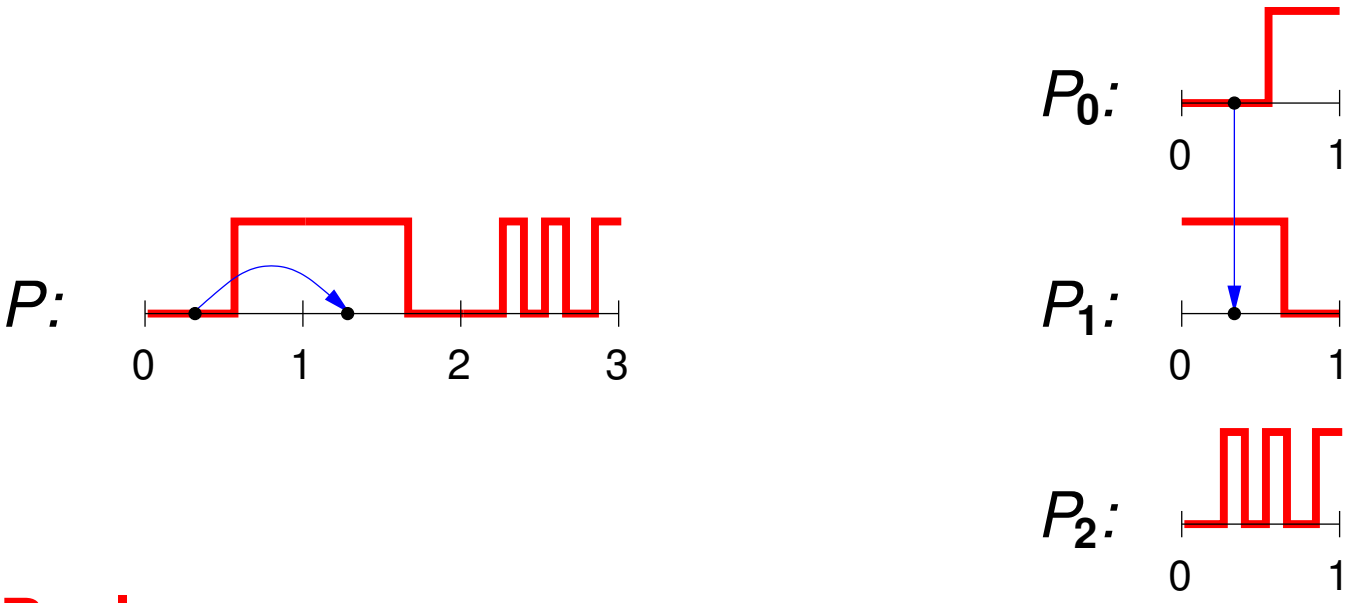




Replace every:

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

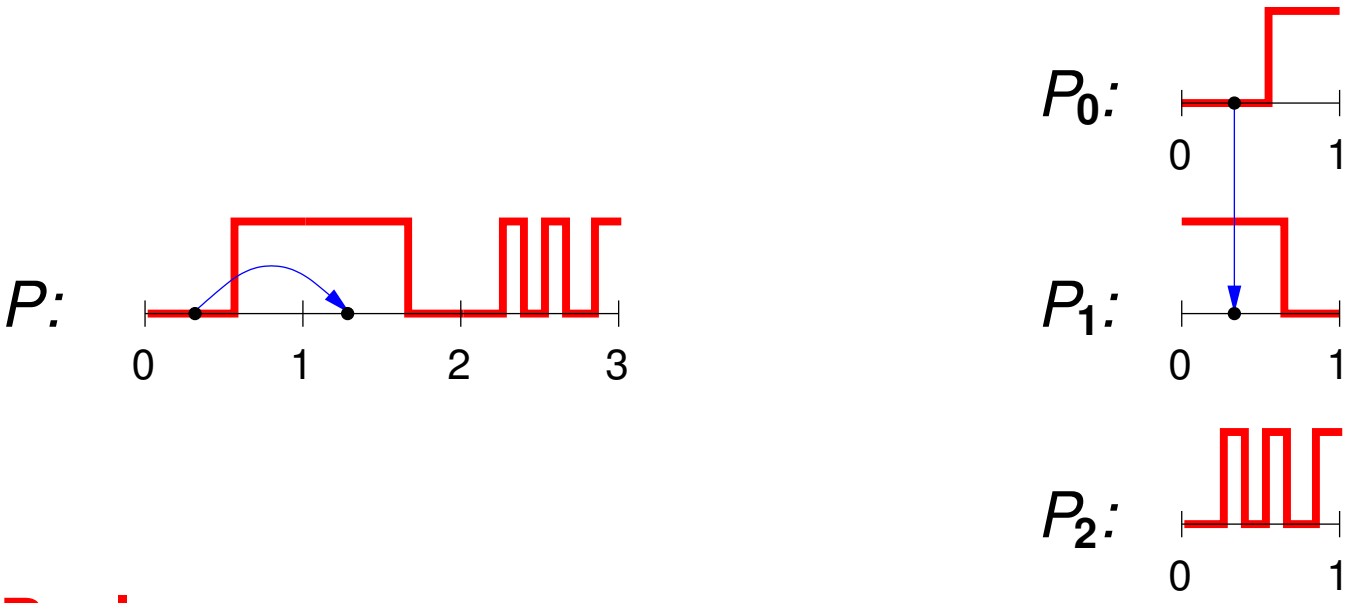
From MSO(<,+1) to MSO(<)



Replace every:

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

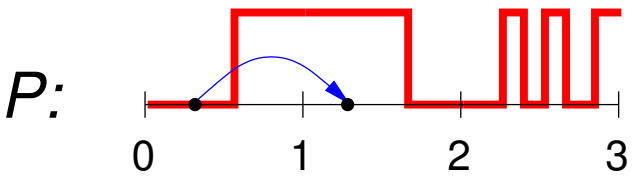
From MSO(<,+1) to MSO(<)

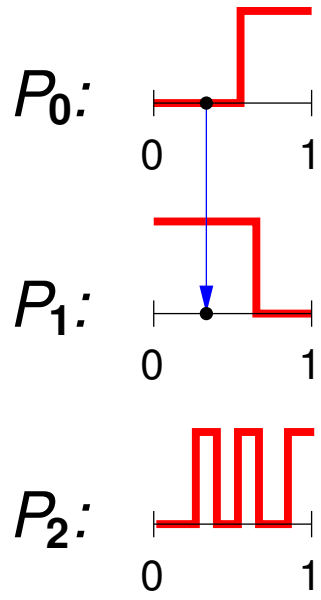


Replace every:

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

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

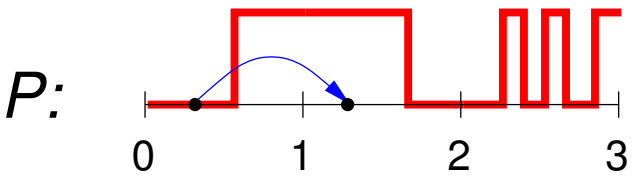


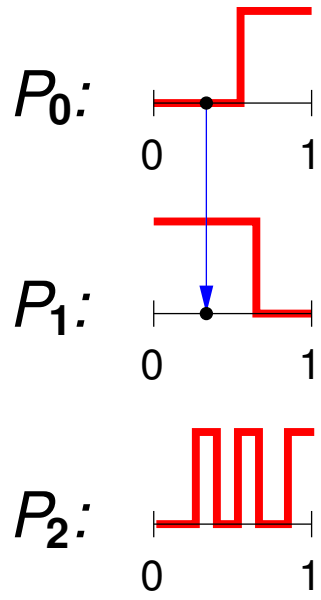


Replace every:

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

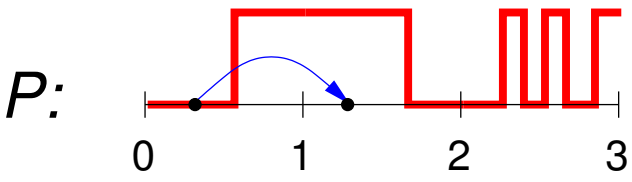


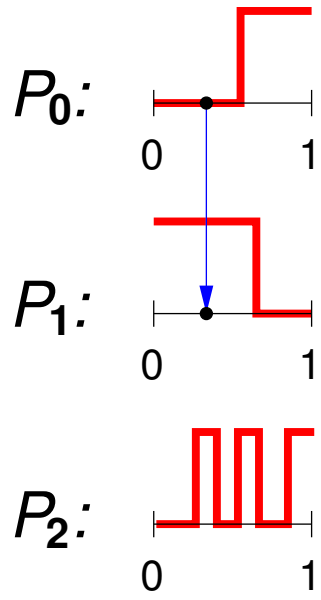


Replace every:

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

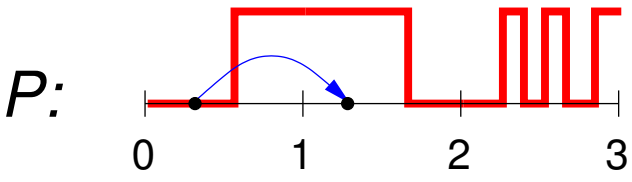


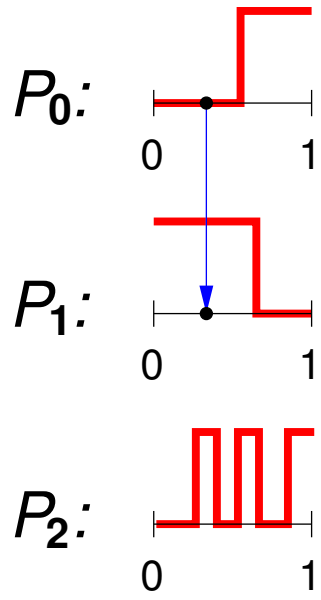


Replace every:

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

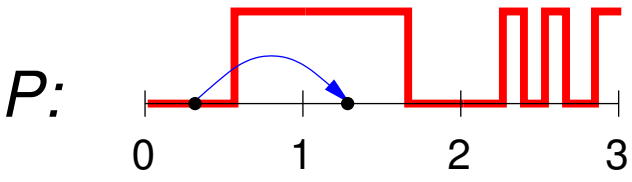


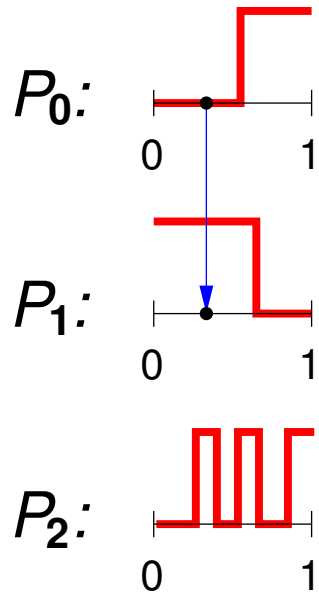


Replace every:

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

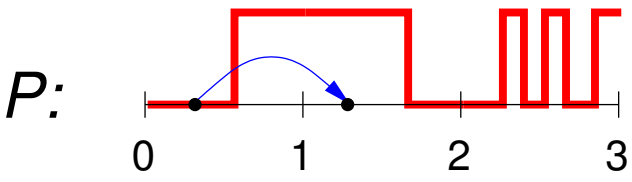


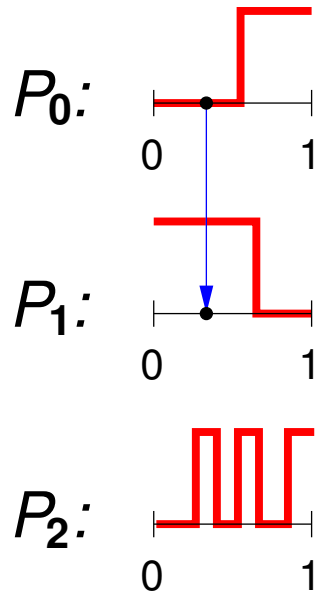


Replace every:

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





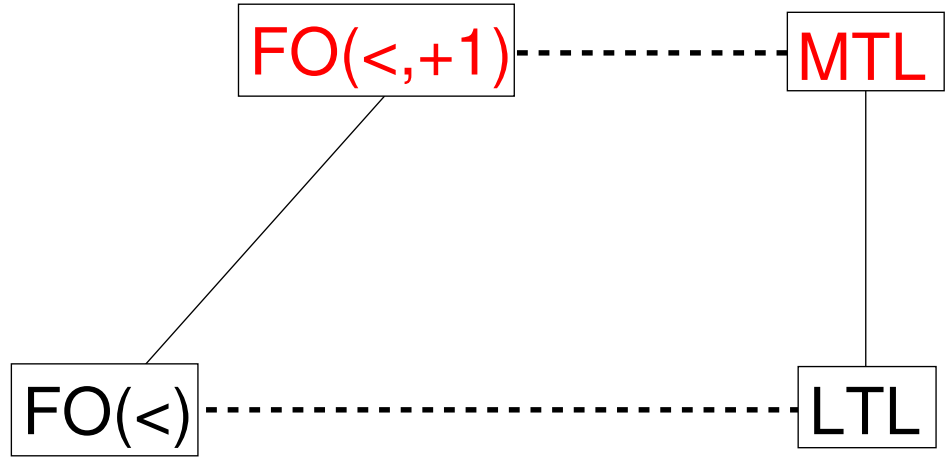
Replace every:

- $\forall x\, \psi(x)$ **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** $\forall P_0 \forall P_1 \forall P_2\, \psi$

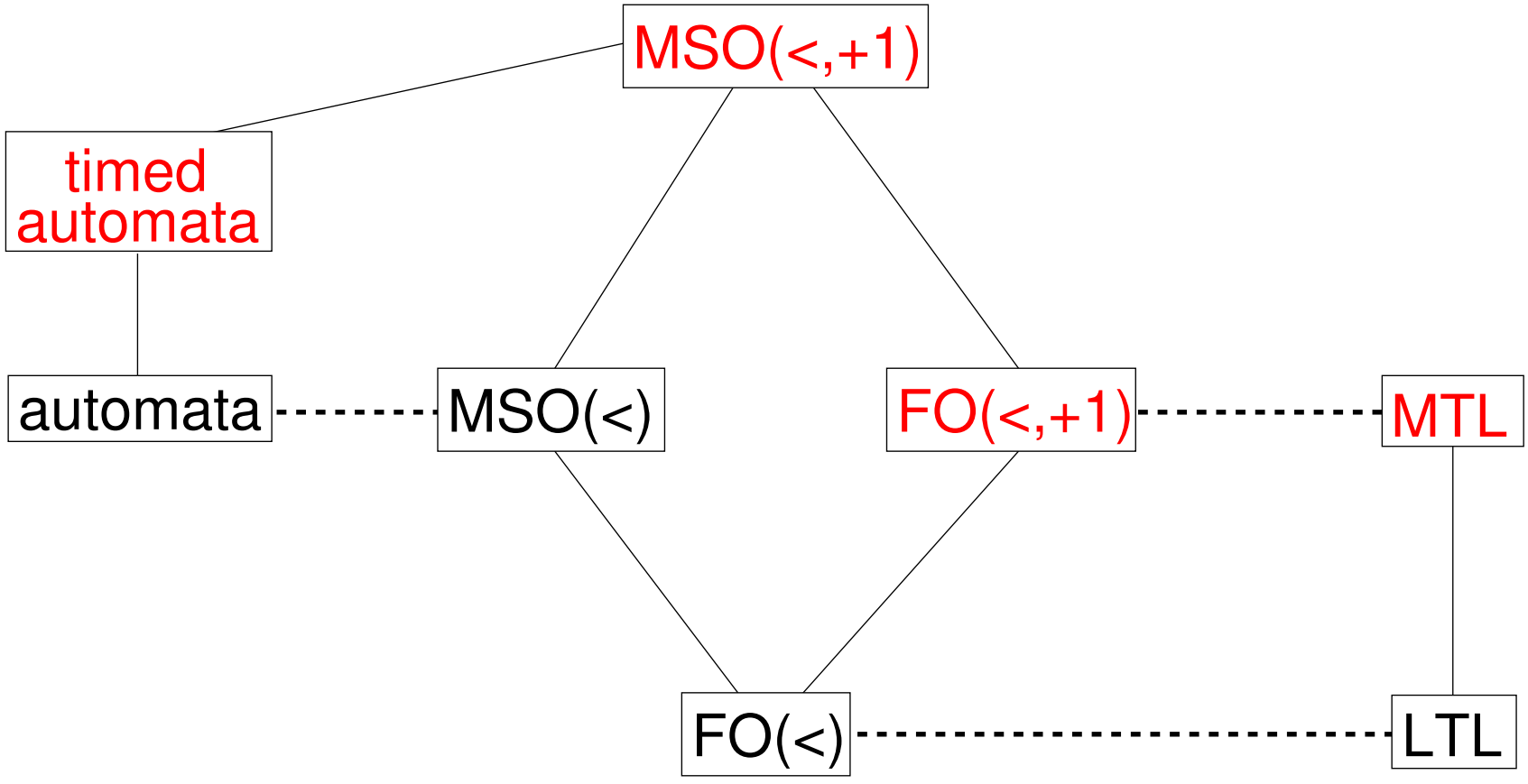
Then φ is satisfiable over $[0, T)$ $\iff$ $\overline{\varphi}$ is satisfiable over $[0, 1)$.

The Time-Bounded Theory: Expressiveness

The Time-Bounded Theory: Expressiveness



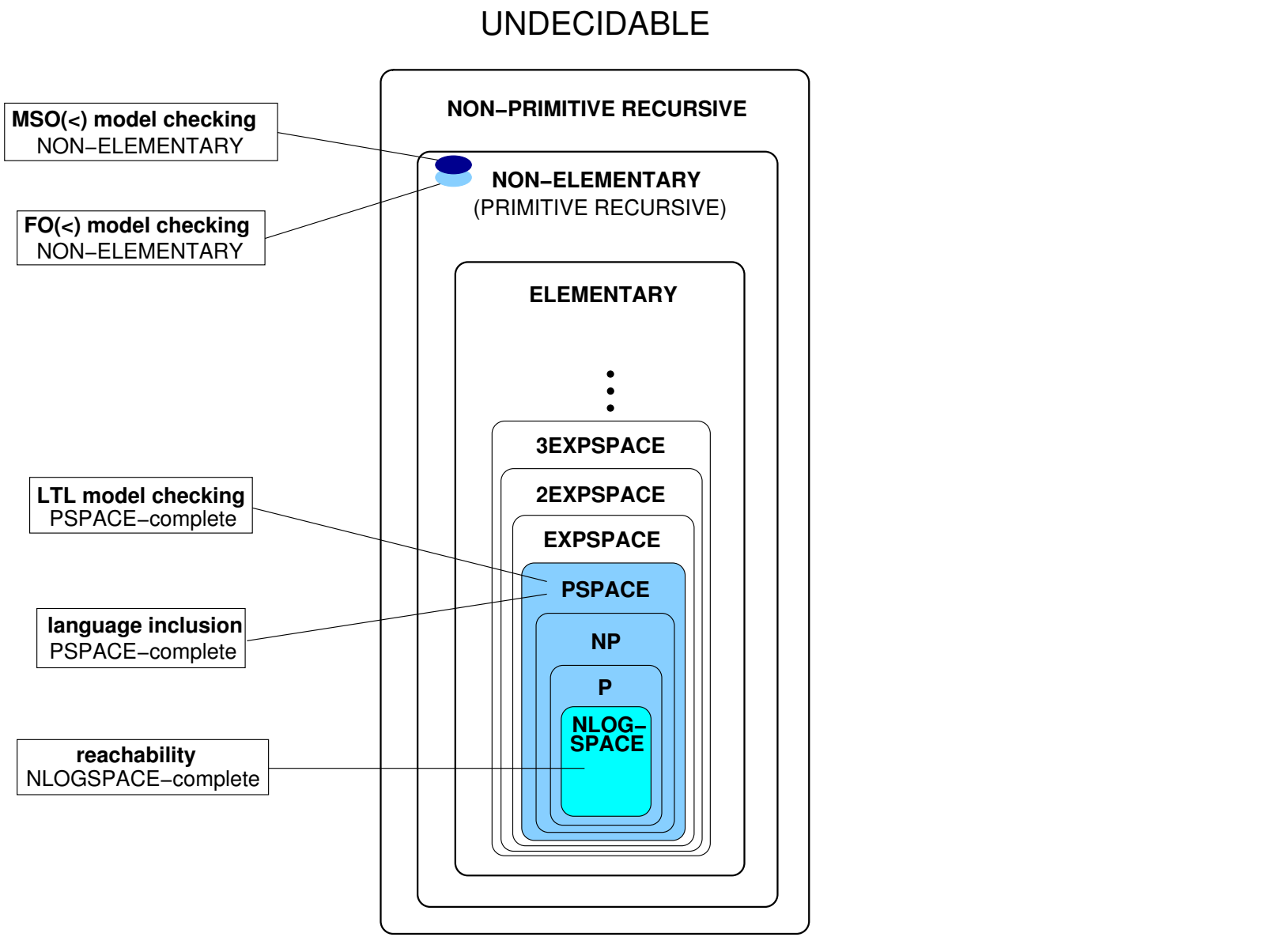
The Time-Bounded Theory: Expressiveness



The Time-Bounded Theory: Complexity

Classical Theory

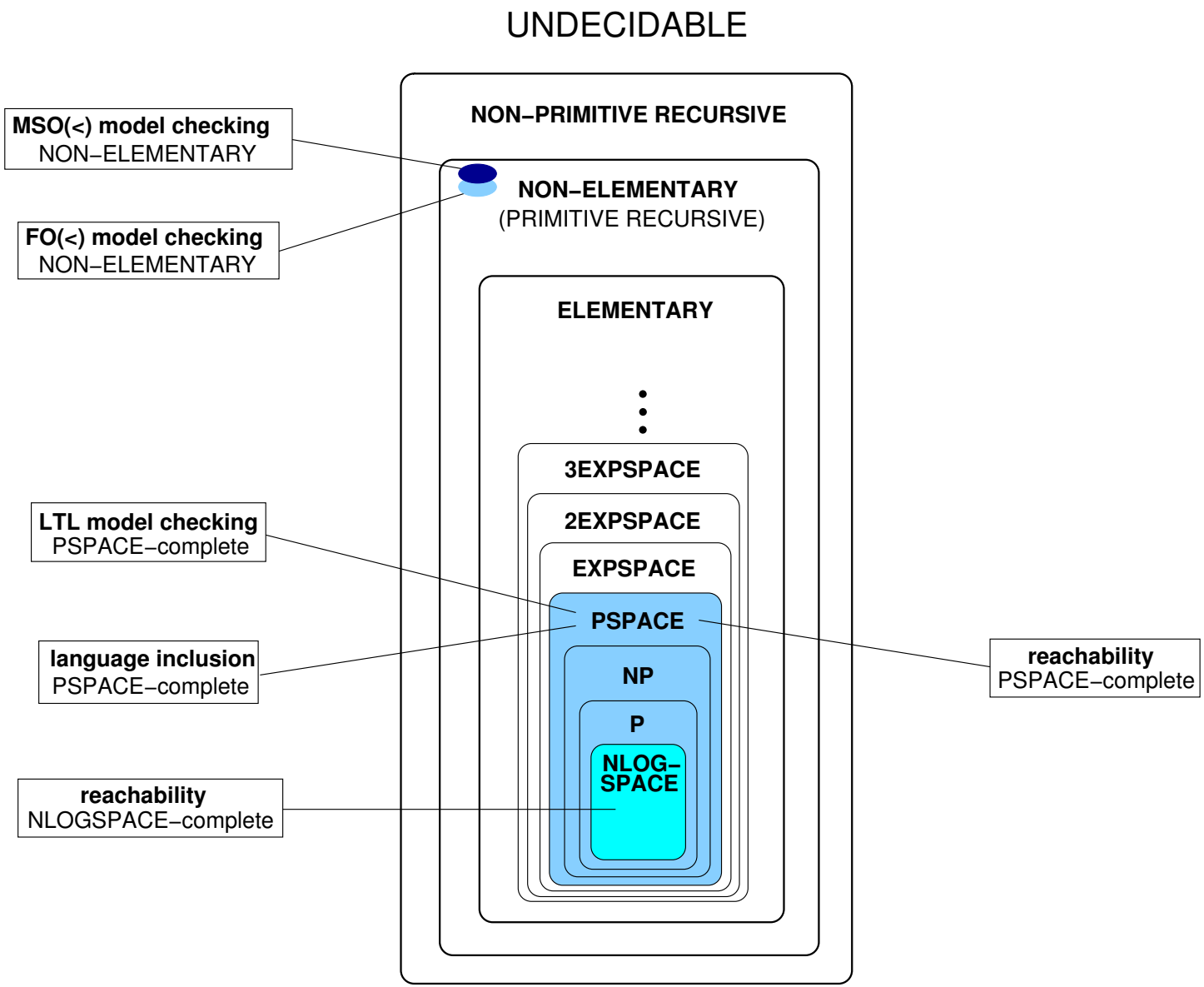
Time–Bounded Theory



The Time-Bounded Theory: Complexity

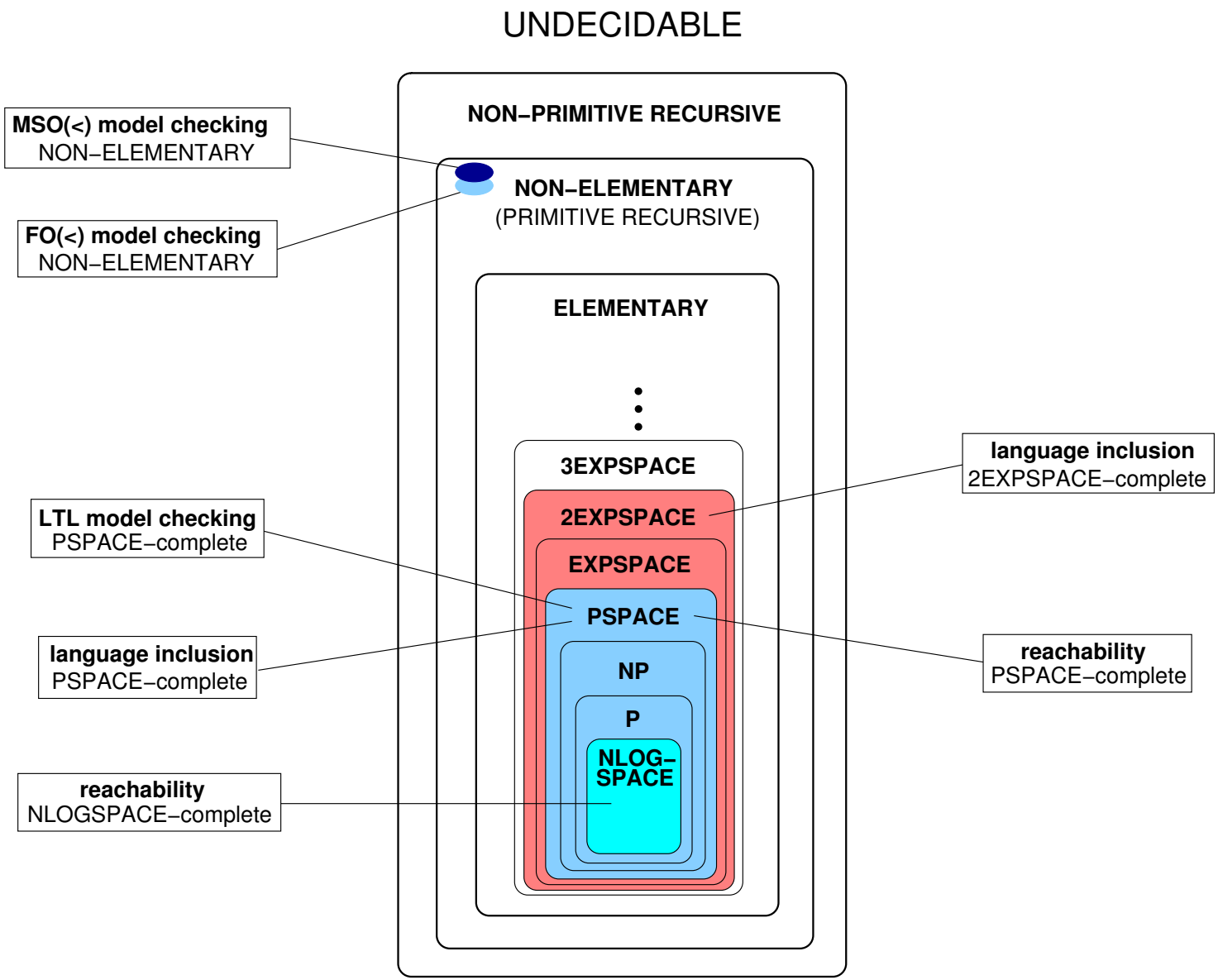
Classical Theory

Time–Bounded Theory



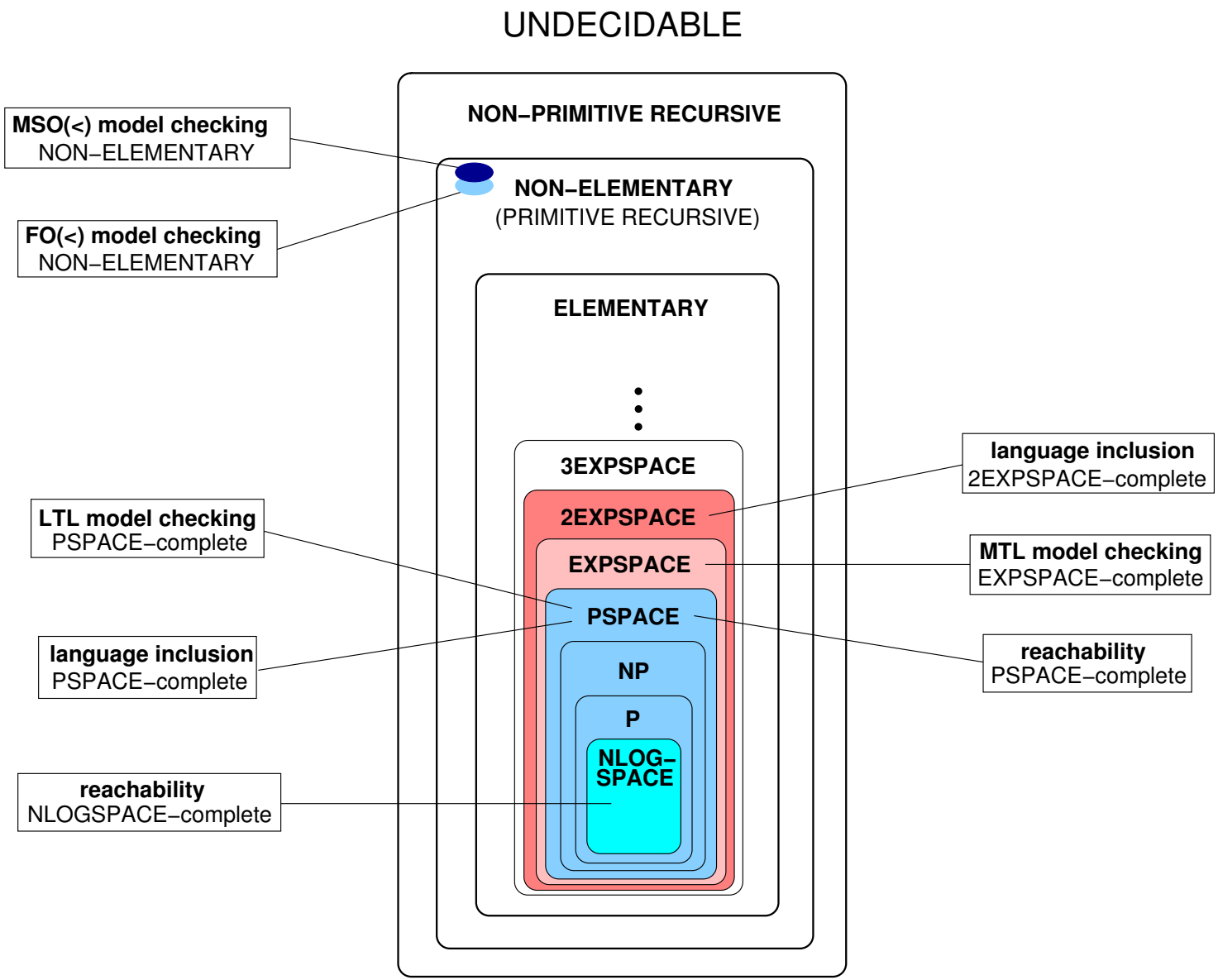
The Time-Bounded Theory: Complexity

Classical Theory **Time–Bounded Theory**



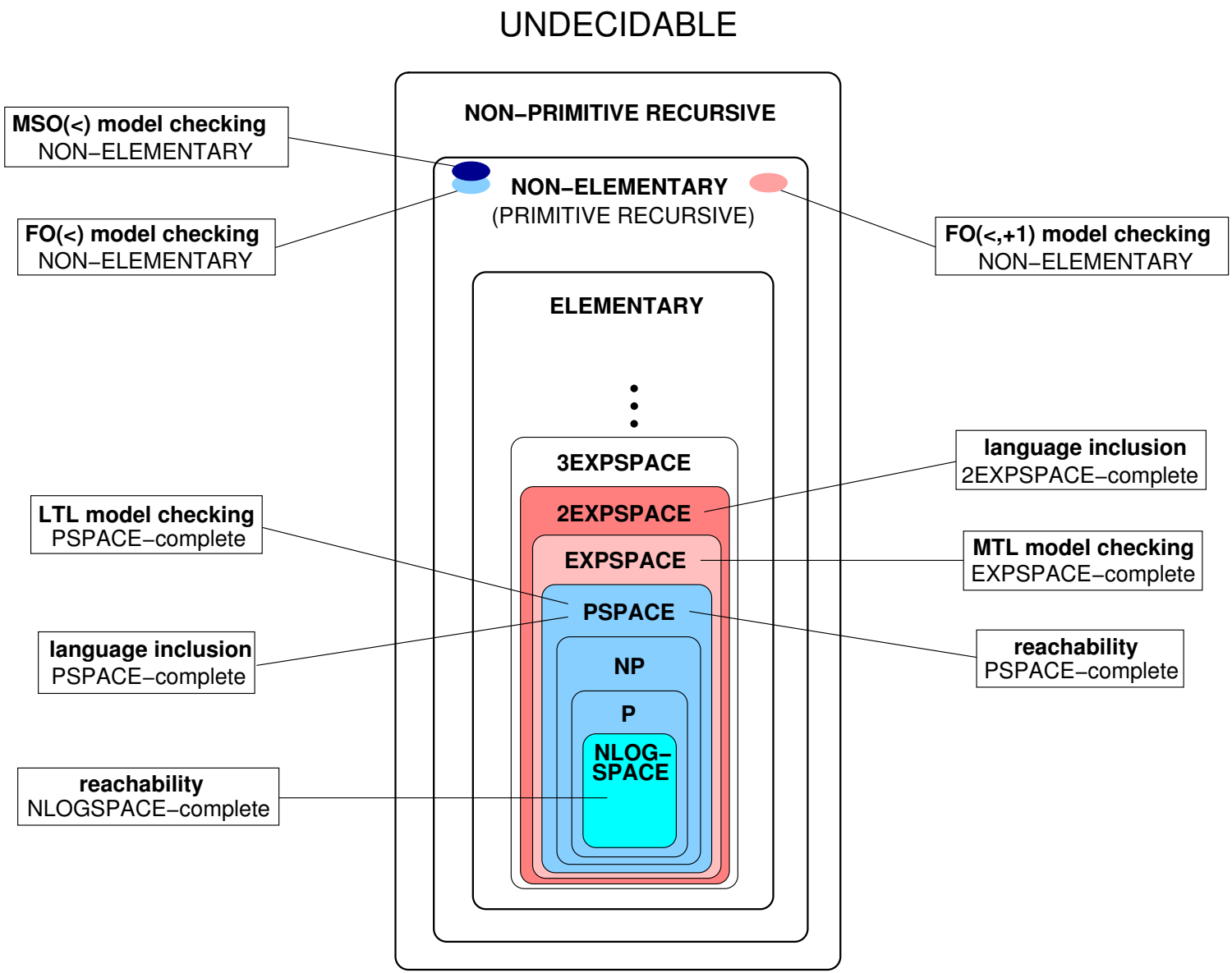
The Time-Bounded Theory: Complexity

Classical Theory **Time–Bounded Theory**



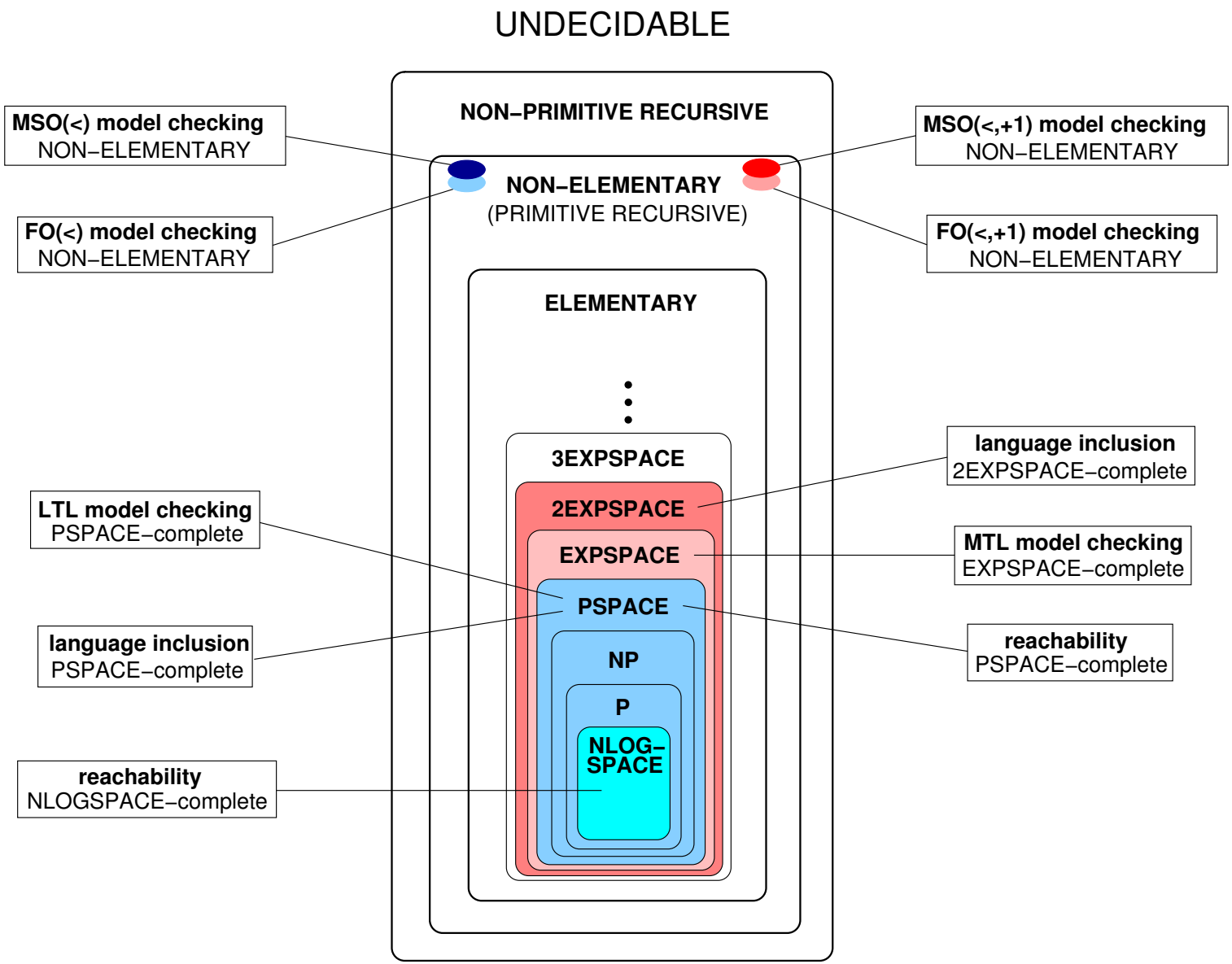
The Time-Bounded Theory: Complexity

Classical Theory **Time–Bounded Theory**



The Time-Bounded Theory: Complexity

Classical Theory **Time–Bounded Theory**



Conclusion and Future Work

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

Conclusion and Future Work

- For specifying and verifying real-time systems, the time-bounded theory is much better behaved than the real-time theory.
- Original motivation for this work was the time-bounded language inclusion problem for timed automata.
  We used logic as a tool to solve this problem.

Conclusion and Future Work

- For specifying and verifying real-time systems, the time-bounded theory is much better behaved than the real-time theory.
- Original motivation for this work was the time-bounded language inclusion problem for timed automata.
  We used logic as a tool to solve this problem.

Future work:

- Timed alternating automata
- Data complexity
- Time-bounded reachability for hybrid systems
- Expressiveness issues
- Implementation