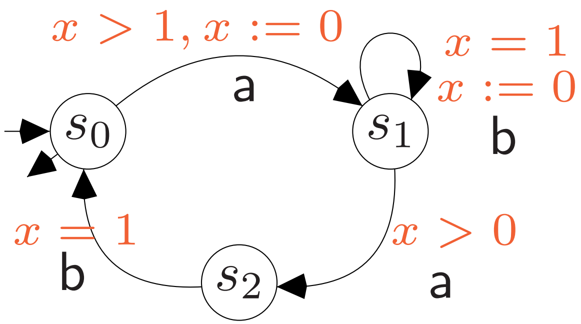
Timed Systems extended with Stacks, Counters and More

Karin Quaas

27th February 2014

Timed Automata [AD90]

- Finite automata extended with a finite set of clocks



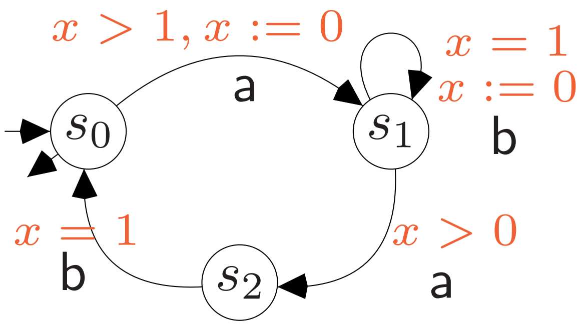
a clock

- ranges over $\mathbb{R}_{\geq 0}$
- grows monotonically while time elapses in a state
- can be compared with constants in $\mathbb{N}$ at the edges
- can be reset to zero at the edges

[AD90] Alur, Dill: A Theory of Timed Automata, 1990.

Timed Automata [AD90]

- Finite automata extended with a finite set of clocks



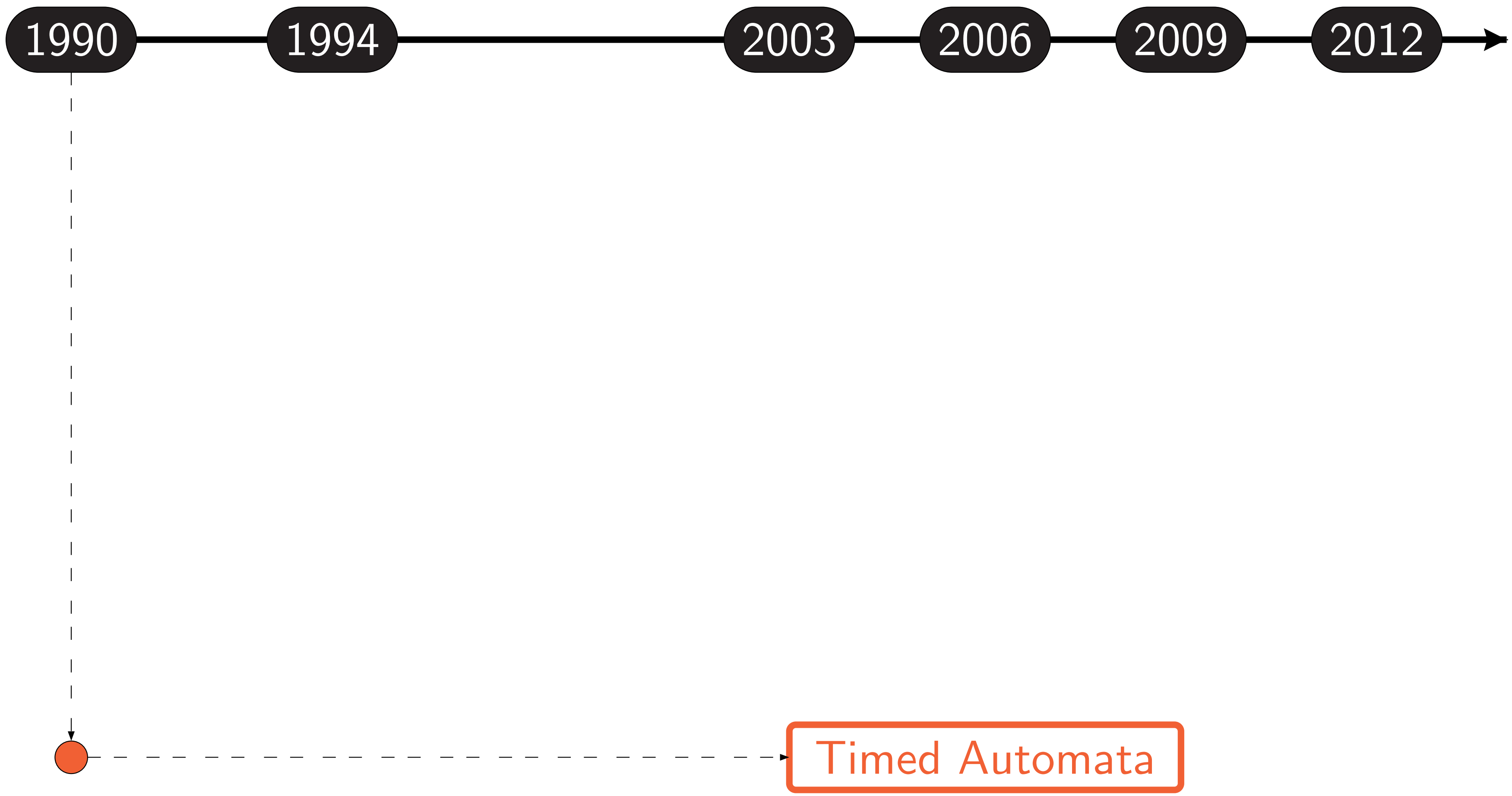
- **Emptiness**: decidable (region graph construction) [AD90].
- **Language inclusion** $(L(\mathcal{A}) \subseteq L(\mathcal{B})?)$: decidable if $\mathcal{B}$ uses ≤ 1 clock [OW04], otherwise undecidable [AD90].
- **Universality**: decidable if ≤ 1 clock is used, otherwise undecidable [OW04].
- **MTL model checking**: decidable [OW05].

[AD90] Alur, Dill: A Theory of Timed Automata, 1990.

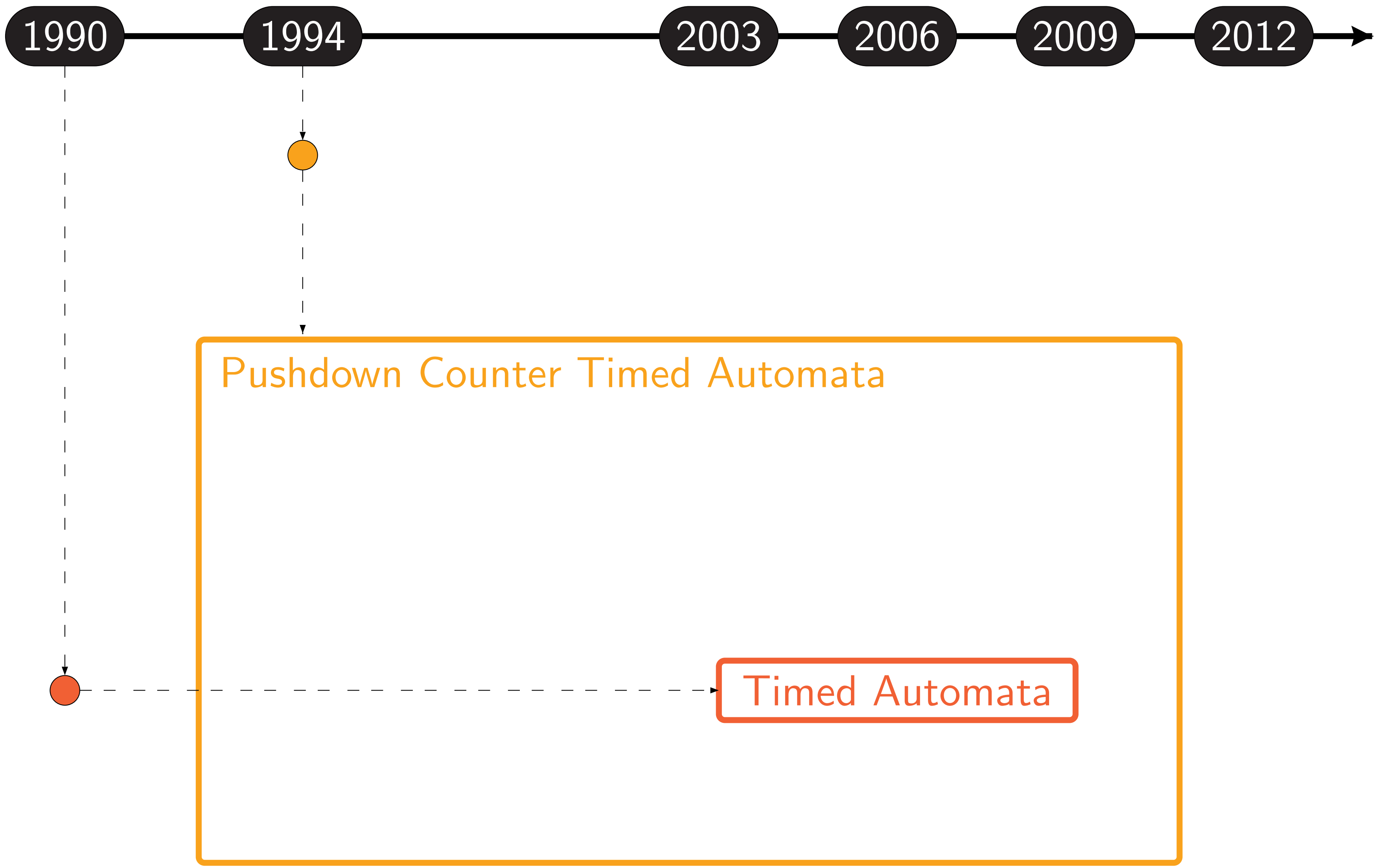
[OW04] Ouaknine, Worrell: On the language inclusion problem for timed automata: Closing a dec.., 2004.

[OW05] Ouaknine, Worrell: On the decidability of Metric Temporal Logic, 2005.

Extensions of Timed Automata with Stacks and Counters

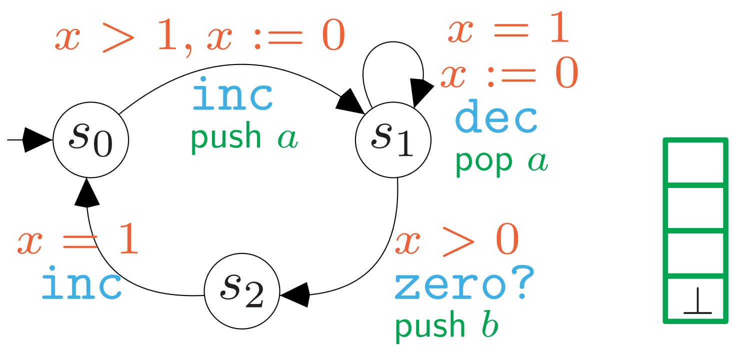


Extensions of Timed Automata with Stacks and Counters



Pushdown Counter Timed Systems [Bou94]

- Finite automata extended with a finite set of clocks, counters and a stack



stack

- takes elements from a finite stack alphabet
- elements can be pushed and popped

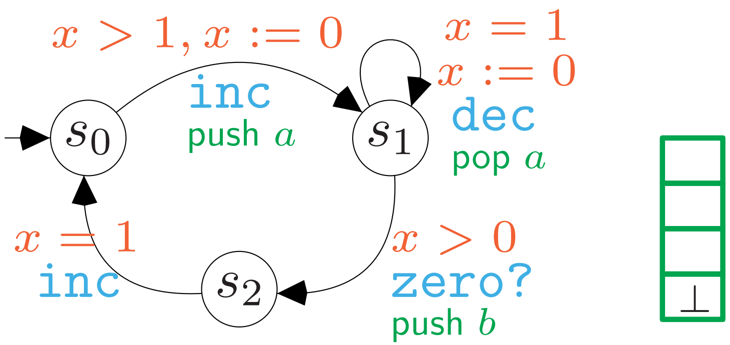
a counter

- ranges over $\mathbb{Z}$
- can be incremented, decremented
- can be compared with constants in $\mathbb{Z}$ at the edges

[Bou94] Bouajjani, Echahed, Robbana: On the automatic verification of systems with ..., 1994.

Pushdown Counter Timed Systems [Bou94]

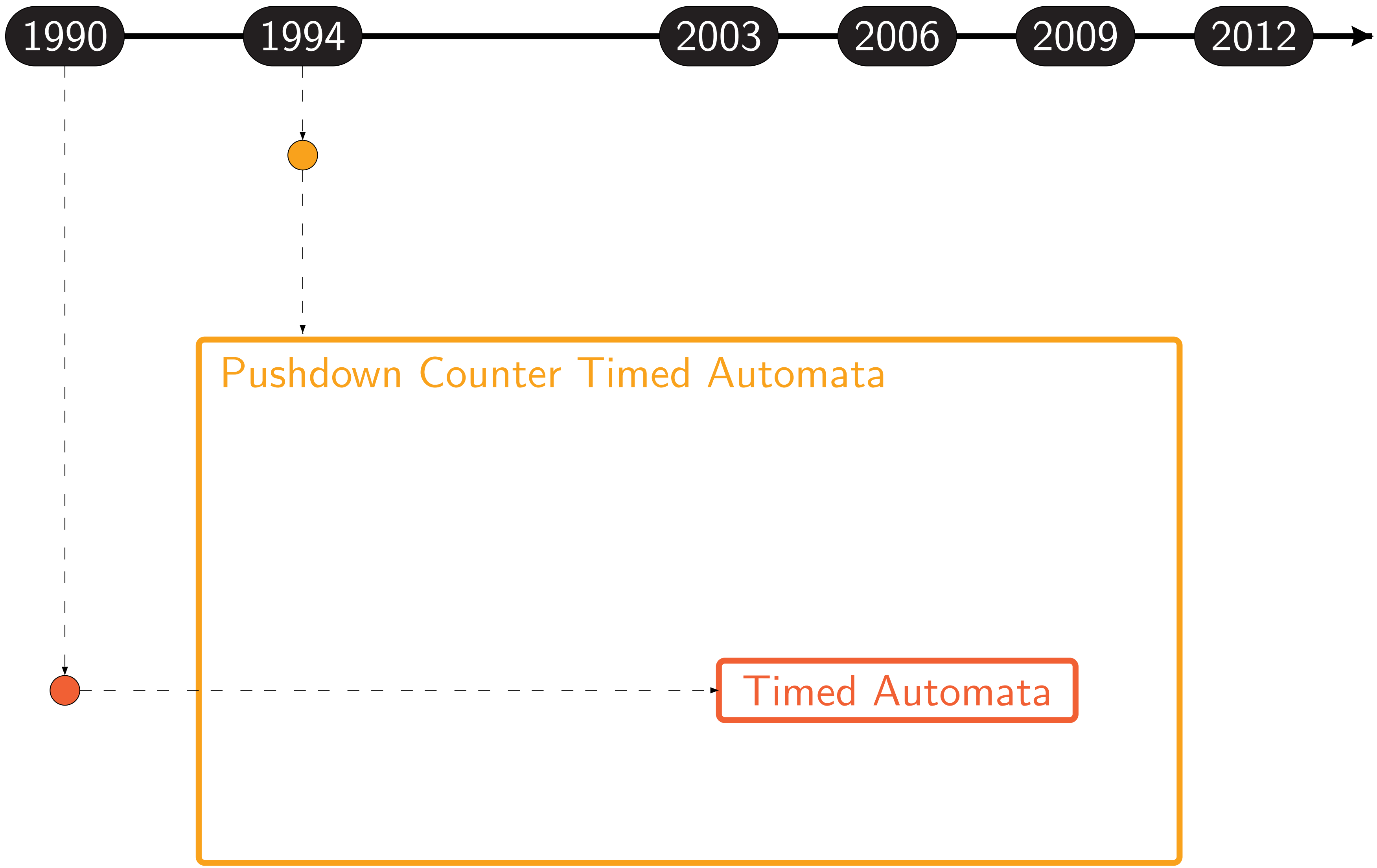
- Finite automata extended with a finite set of clocks, counters and a stack



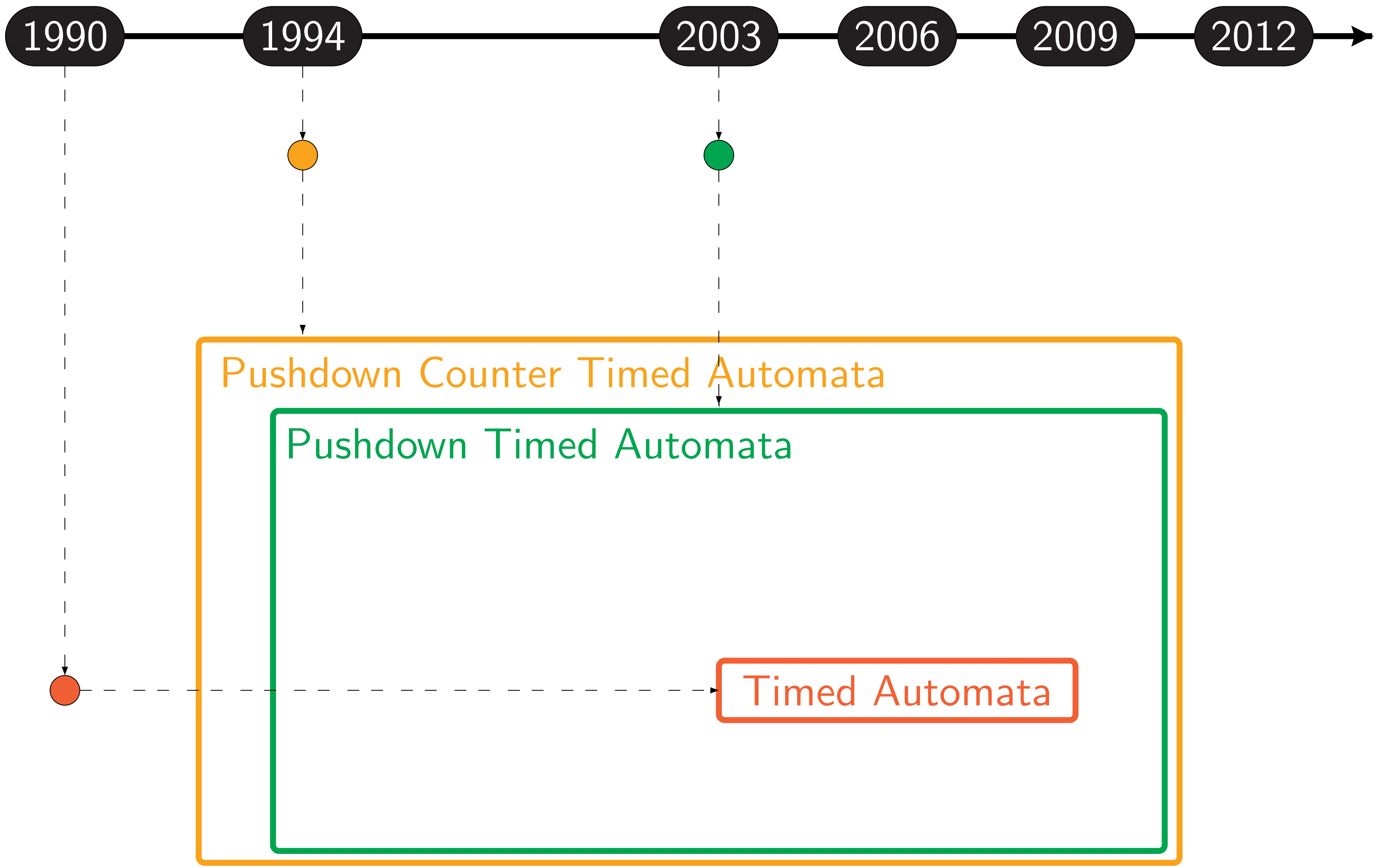
- **Verification of reachability formulas** that constrain locations, clocks, and counter values
- Decidable for pushdown timed systems, pushdown timed systems with monotonic counters, pushdown timed systems with observers
- Reduction to emptiness problem for pushdown automata using extension of region graph

[Bou94] Bouajjani, Echahed, Robbana: On the automatic verification of systems with ..., 1994.

Extensions of Timed Automata with Stacks and Counters

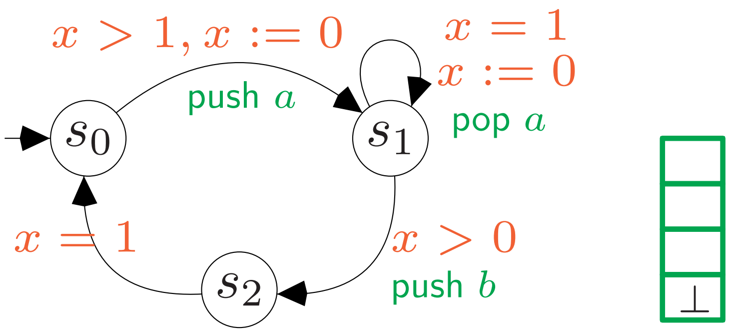


Extensions of Timed Automata with Stacks and Counters



Pushdown Timed Systems [Dang03]

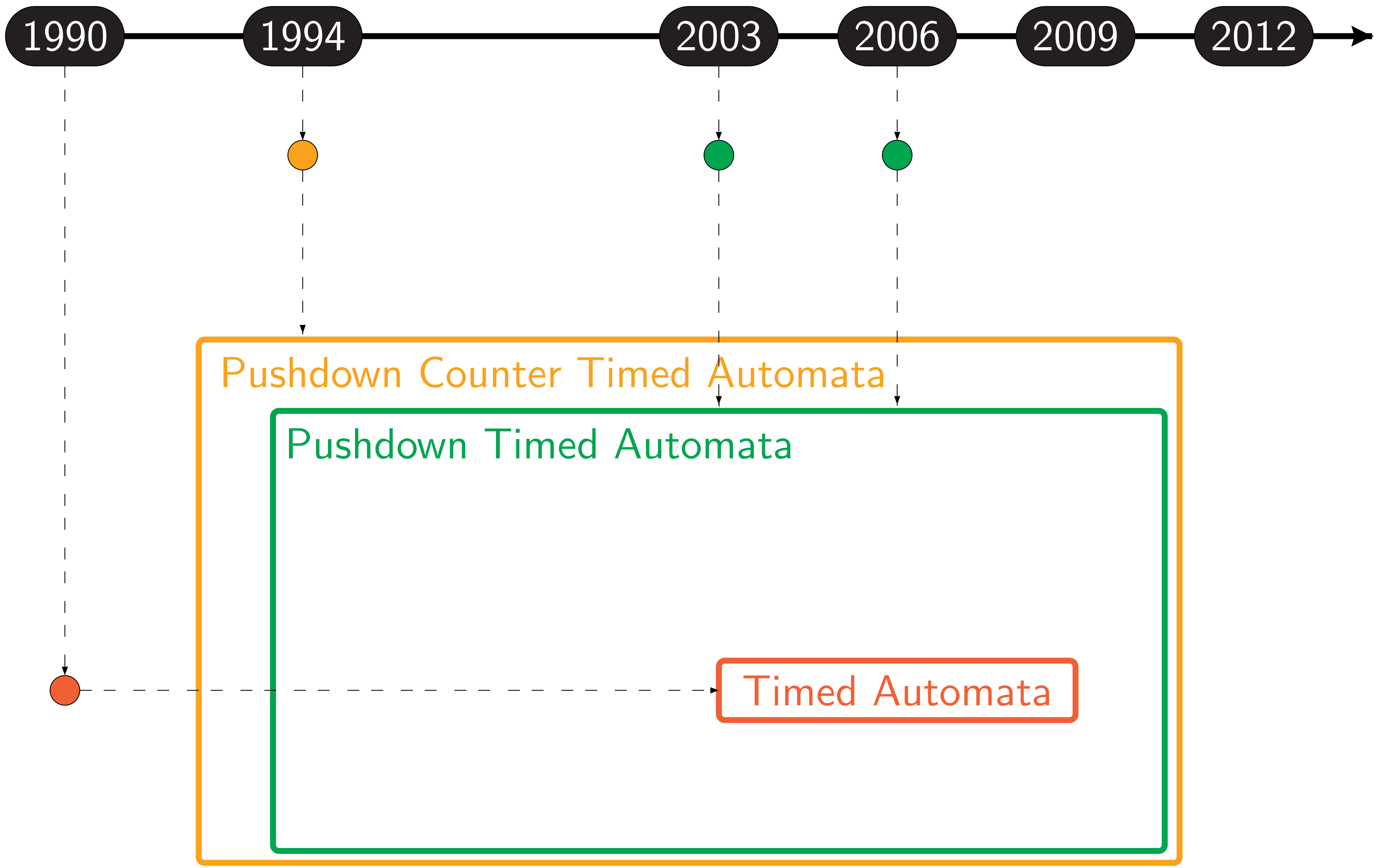
- Finite automata extended with a finite set of clocks and a stack



- The set $\{(\gamma, \gamma') \mid \gamma \text{ reaches } \gamma' \text{ in } \mathcal{A}\}$ has a decidable characterization
- Refinement of the region equivalence

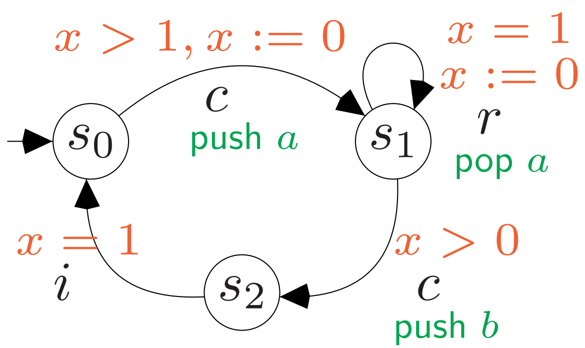
[Dang03] Dang: Pushdown timed automata: a binary reachability characterization and safety verification, 2003.

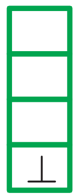
Extensions of Timed Automata with Stacks and Counters



Timed (Visibly) Pushdown Automata [Emmi06]

- Finite automata extended with a finite set of clocks and a stack





Visibly pushdown stack

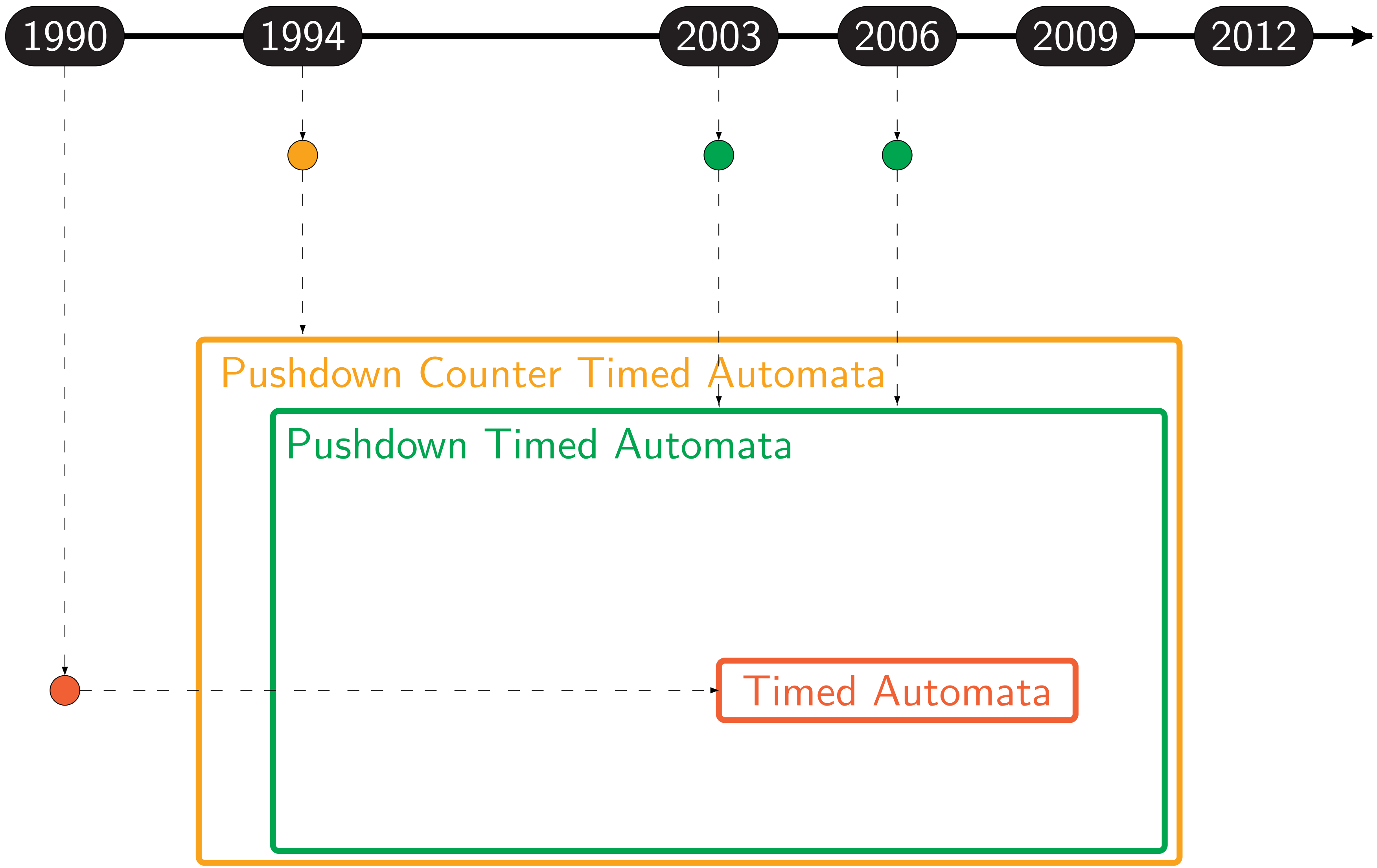
input alphabet partitioned into

- *call* symbols to push
- *return* symbols to pop
- *internal* symbols

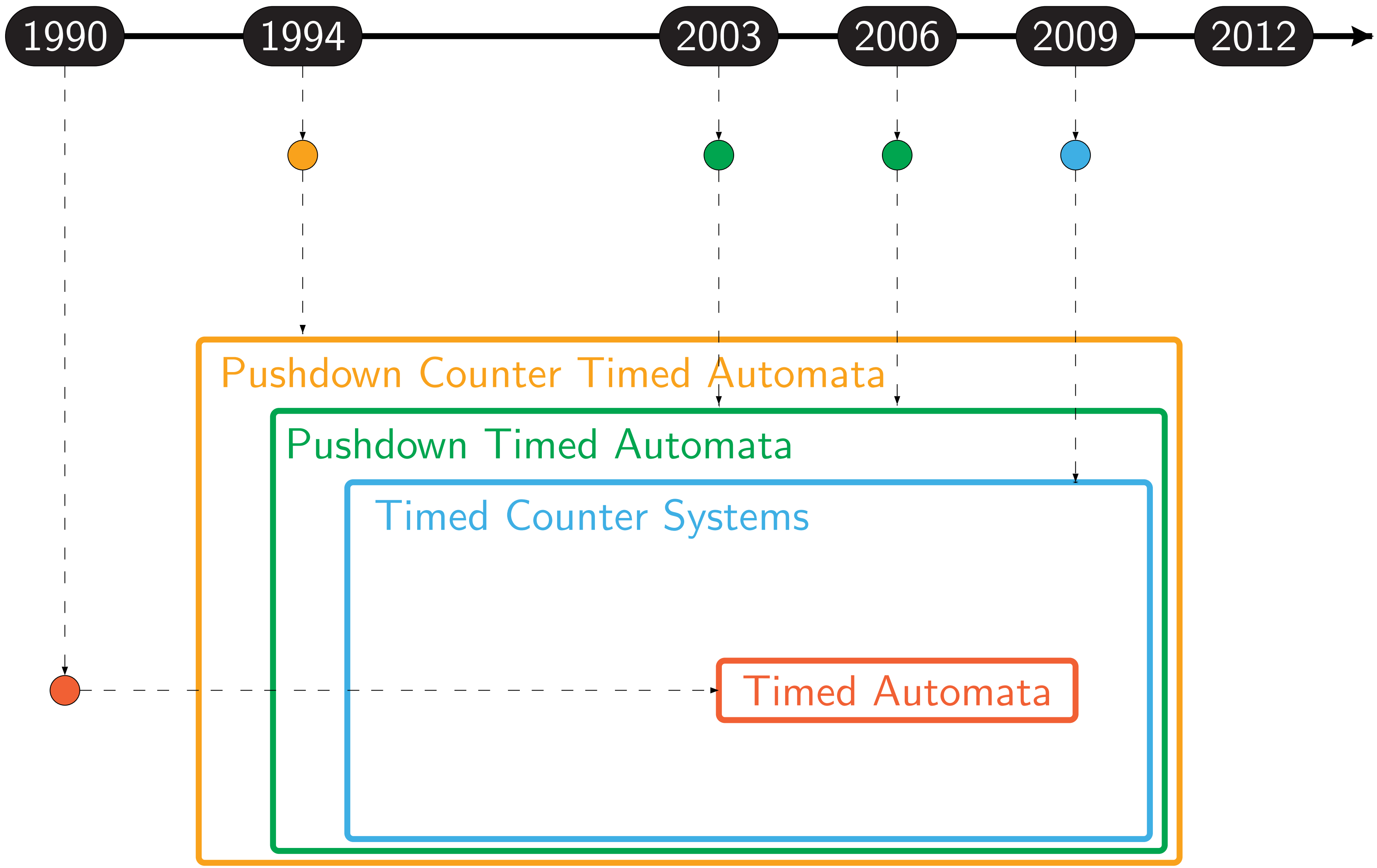
- **Language Inclusion** ($L(\mathcal{A}) \subseteq L(\mathcal{B})$?): decidable if $\mathcal{A}$ is a timed pushdown automaton, $\mathcal{B}$ is a timed automaton with ≤ 1 clock (Proof not correct!)
- **Universality** for timed visibly pushdown automata with one clock is undecidable (Gaps in the proof!)

[Emmi06] Emmi, Majumdar: Decision Problems for the Verification of Real-Time Software, 2006.

Extensions of Timed Automata with Stacks and Counters

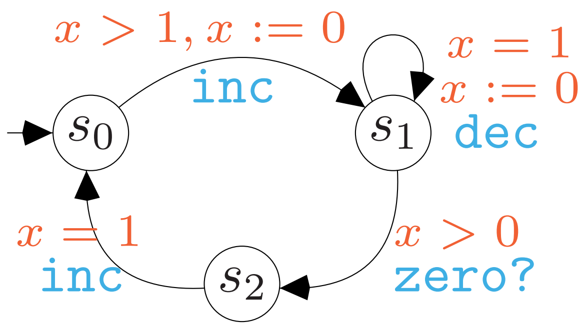


Extensions of Timed Automata with Stacks and Counters



Timed Counter Systems [BFS09]

- Finite automata extended with a finite set of clocks and counters



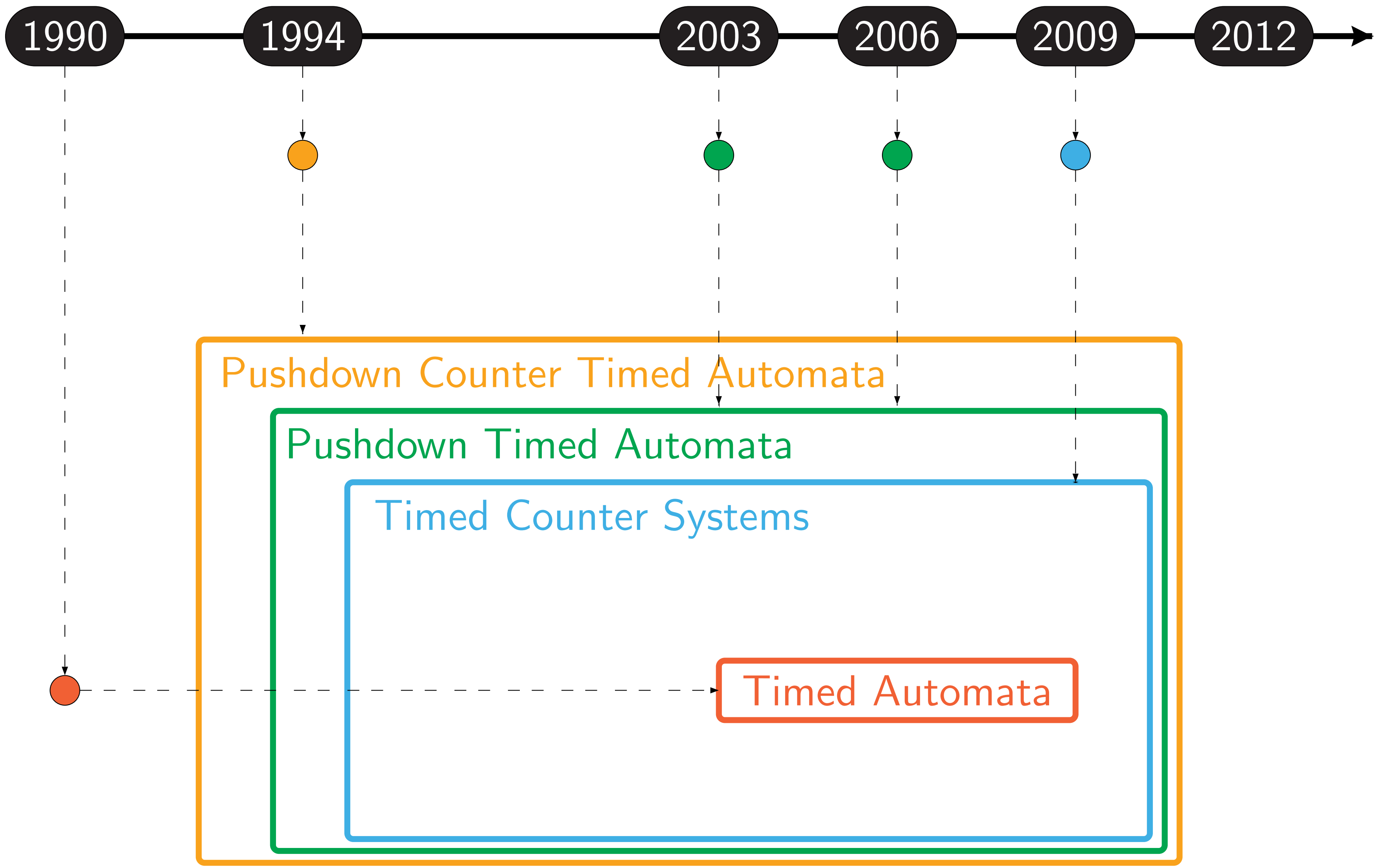
a counter

- ranges over $\mathbb{N}$
- can be incremented and decremented
- can be compared with zero

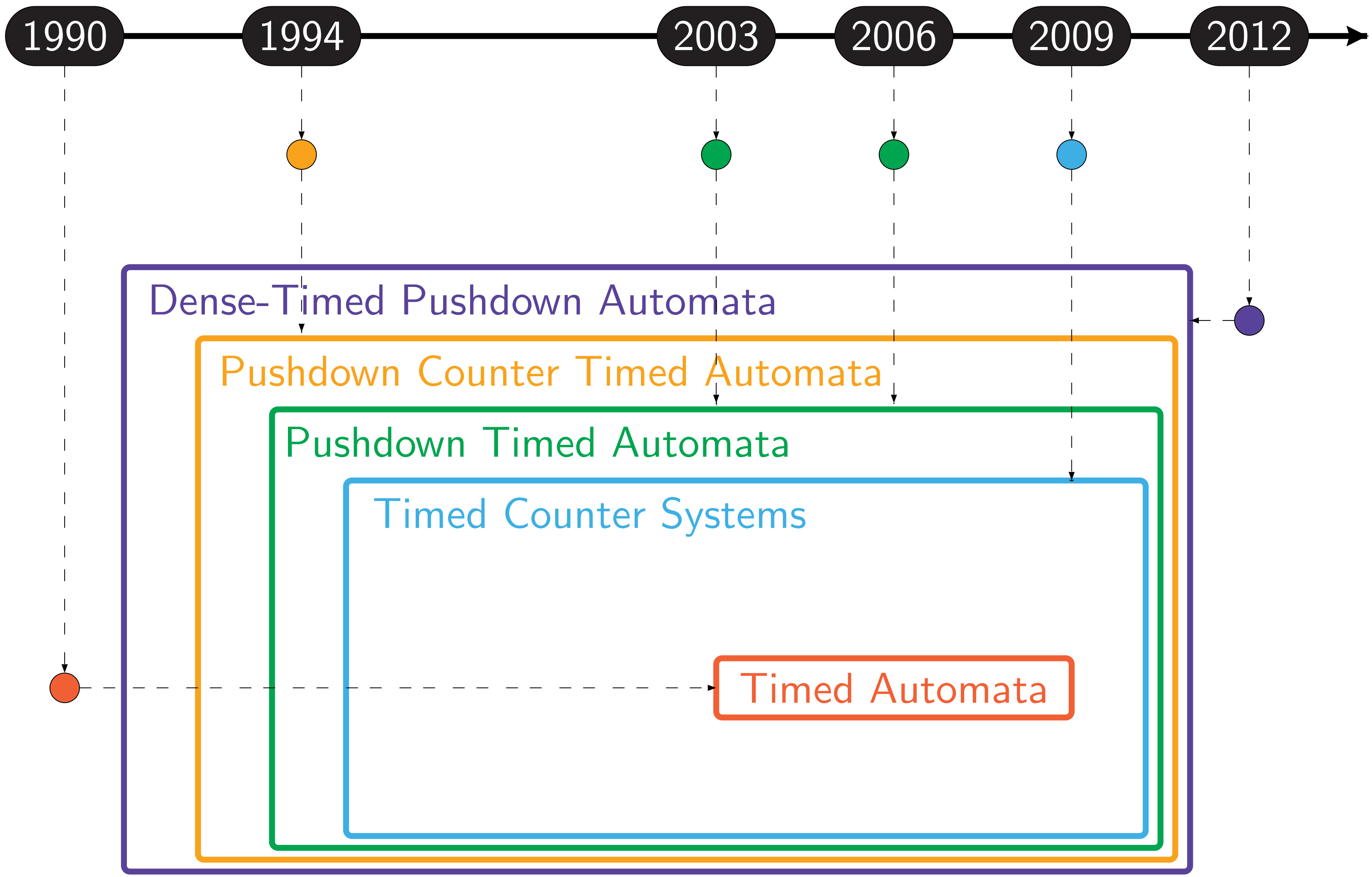
- **Emptiness**: decidable for all subclasses of counter systems for which emptiness is decidable, e.g. VASS, reversal-bounded counter machines, etc.
- Reduction to emptiness of the corresponding counter system by extending the region graph

[BFS09] Bouchy, Finkel, Sangnier: Reachability in Timed Counter Systems, 2009.

Extensions of Timed Automata with Stacks and Counters

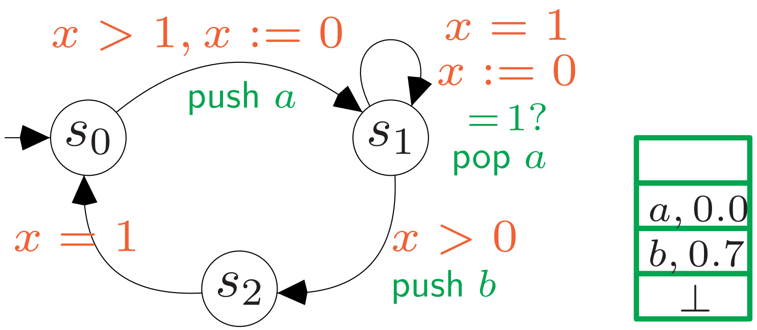


Extensions of Timed Automata with Stacks and Counters



Dense-Timed Pushdown Automata [AAS12]

- Finite automata extended with a finite set of clocks and stack



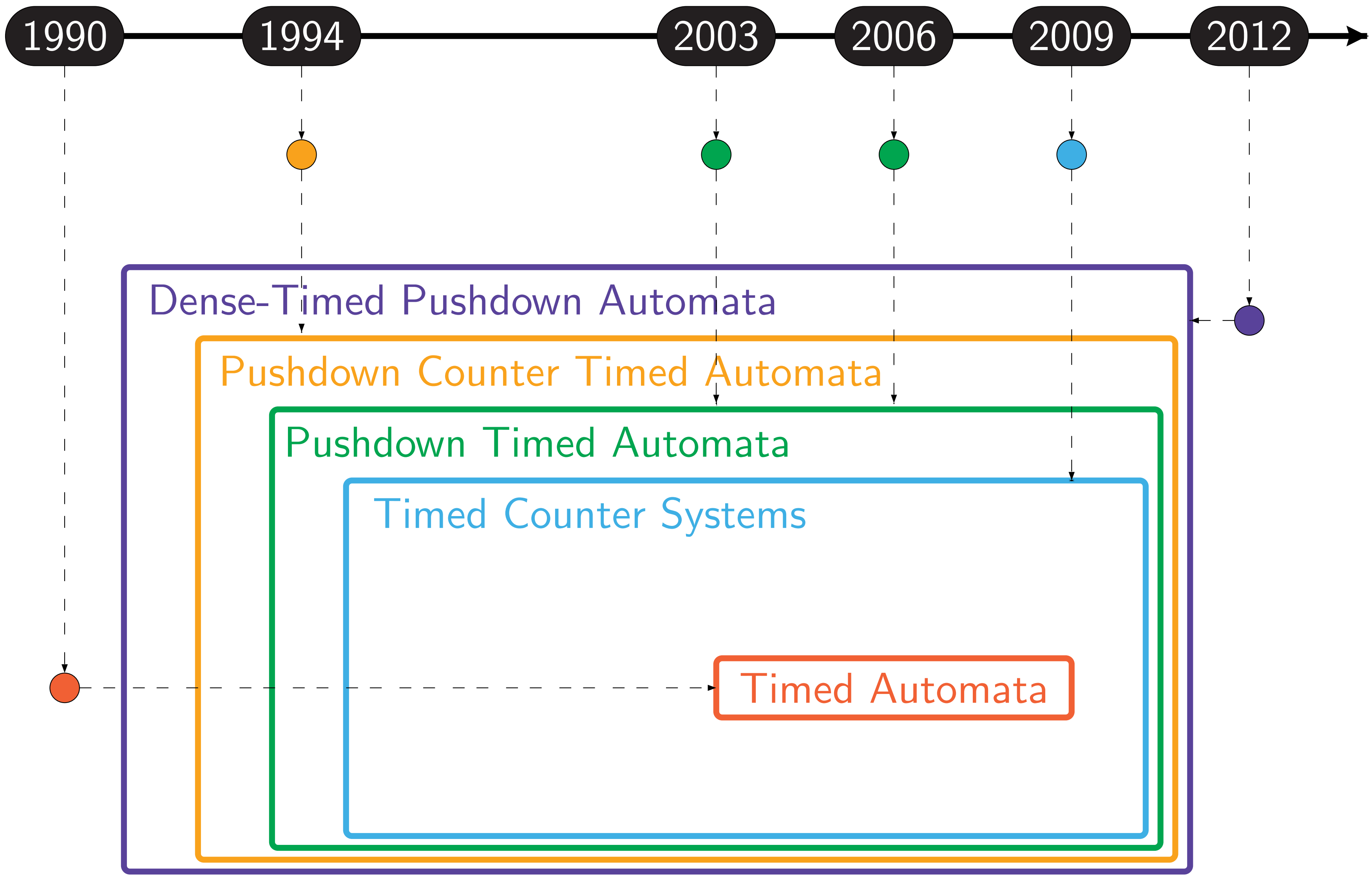
Stack

- takes elements from infinite alphabet
- each element has an age
- initial age when pushed is 0
- element is popped if guard is satisfied

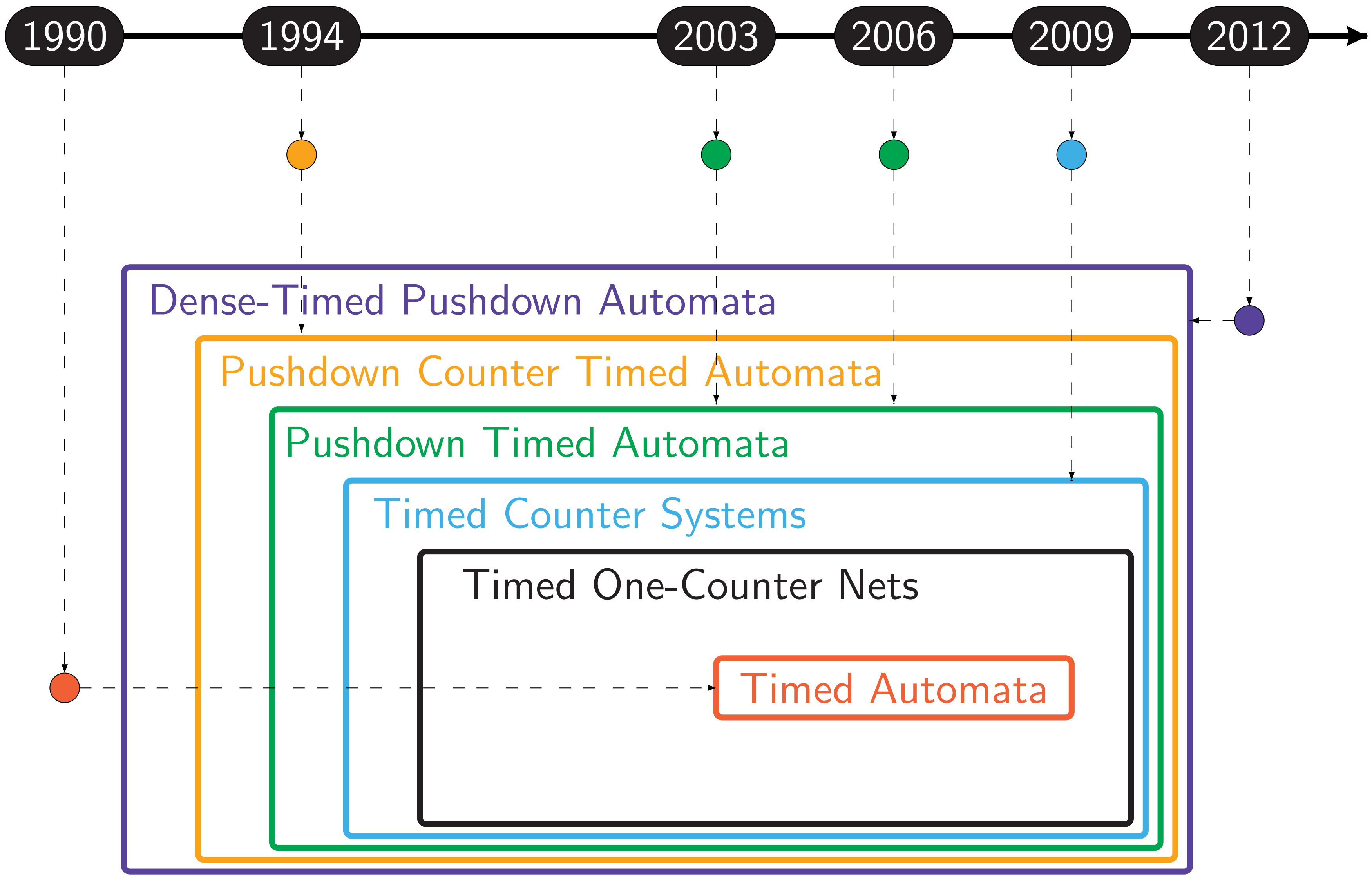
- **Emptiness**: decidable
- Reduction to emptiness for pushdown automata by an intricate region graph construction.

[AAS12] Abdulla, Atig, Stenman: Dense-timed Pushdown Automata, 2012.

Extensions of Timed Automata with Stacks and Counters

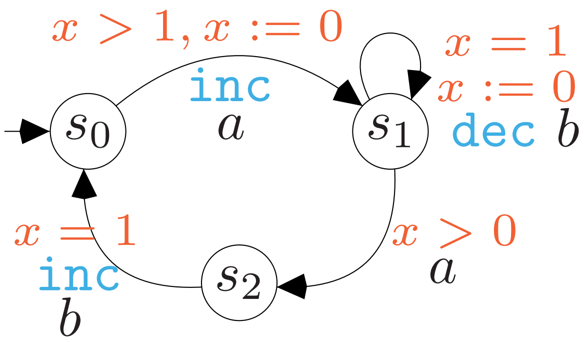


Extensions of Timed Automata with Stacks and Counters



Timed One-Counter Nets

- Finite automata extended with a finite set of clocks and one counter



a clock

- ranges over $\mathbb{R}_{\geq 0}$
- grows monotonically while time elapses in a state
- can be compared with constants in $\mathbb{N}$ at the edges
- can be reset to zero at the edges

a counter

- ranges over $\mathbb{N}$
- can be incremented, decremented
- no zero test
- cannot become negative: edges are blocked

Language Inclusion Problem for Timed One-Counter Nets

Instance: Two timed one-counter nets $\mathcal{A}$ and $\mathcal{B}$.
Question: Does $L(\mathcal{A}) \subseteq L(\mathcal{B})$ hold?

Model   Specification

Theorem.

1. The language inclusion problem is undecidable, even if $\mathcal{A}$ is deterministic and uses no clocks, and $\mathcal{B}$ is a timed automaton with at most one clock.

2. The language inclusion problem is decidable if $\mathcal{A}$ is a timed automaton, and $\mathcal{B}$ is a timed one-counter net with at most one clock.

$\Rightarrow$ Use timed one-counter nets as specification!

Corollary.

The universality problem for timed one-counter nets with at most one clock variable is decidable.

Proof Idea of the Decidability Result

Theorem.
2. The language inclusion problem is decidable if $\mathcal{A}$ is a timed automaton, and $\mathcal{B}$ is a timed one-counter net with at most one clock.

Proof. (Sketch)

- Generalize the corresponding proof for $\mathcal{B}$ a timed automaton with at most one clock [OW04]
- Construct a downward compatible well-structured state-transition system
- The nodes are *joint configurations* of $\mathcal{A}$ and $\mathcal{B}$
- Solve a reachability problem on the state-transition system

[OW04] Ouaknine, Worrell: On the language inclusion problem for timed automata: Closing a dec.., 2004.

Proof Idea of the Undecidability Result

Theorem.

1. The language inclusion problem is undecidable, even if $\mathcal{A}$ is deterministic and uses no clocks, and $\mathcal{B}$ is a timed automaton with at most one clock.

Proof. (Sketch)

- Reduction of the (undecidable) reachability problem for channel machines
- Given a channel machine $\mathcal{C}$ and a state q, we can define a timed language $L(\mathcal{C}, q)$ that encodes computations of channel machines with insertion errors [OW06]
- Construct a timed one-counter net $\mathcal{A}$ to exclude insertion errors:

$$\mathcal{C} \text{ does not reach } q \Leftrightarrow L(\mathcal{A}) \cap L(\mathcal{C}, q) = \emptyset$$

- Construct timed automaton $\mathcal{B}$ with one clock that recognizes the complement of $L(\mathcal{C}, q)$:

$$L(\mathcal{A}) \cap L(\mathcal{C}, q) = \emptyset \Leftrightarrow L(\mathcal{A}) \subseteq \overline{L(\mathcal{C}, q)} \Leftrightarrow L(\mathcal{A}) \subseteq L(\mathcal{B})$$

[OW06] Ouaknine, Worrell: On Metric Temporal Logic and Faulty Turing Machines, 2006.

Details of the Undecidability Proof

$\mathcal{B}$: Channel machine $M = (\{p, q, r\}, p, \{e, t, x\}, \Delta)$, $(p, !t, q), (q, ?e, q), \ldots \in \Delta$

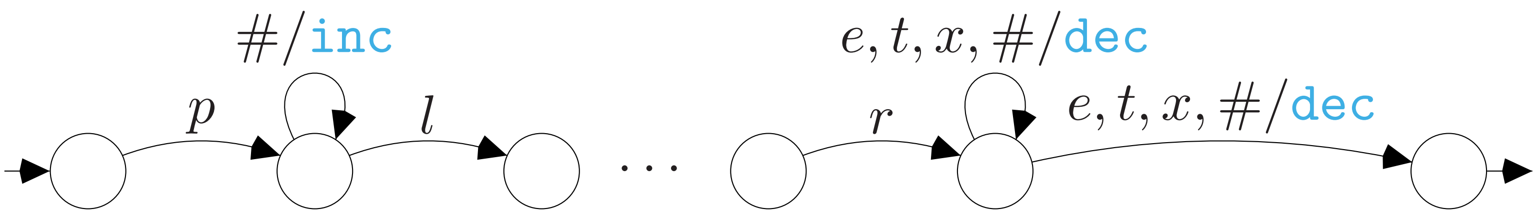
Initial configuration (p, ε) is encoded by $(p, 0)(\#, \delta_1) \ldots (\#, \delta_n)$, where $0 < \delta_1 < \cdots < \delta_n < 1$ for some $n \in \mathbb{N}$.

The transition $\langle (p, tex), !t, (q, text) \rangle$ may be encoded by

$$(p, 6)(t, 6.1)(e, 6.15)(x, 6.5)(\#, 6.73)(!t, 7)(q, 8)(t, 8.1)(e, 8.15)(x, 8.5)(t, 8.73) \ldots$$

$= 1$ $= 2$ $= 2$

$\mathcal{A}$:



Consequences of the Undecidability Result (1)

Theorem.

1. The language inclusion problem is undecidable, even if $\mathcal{A}$ is deterministic and uses no clocks, and $\mathcal{B}$ is a timed automaton with at most one clock.

Recall [Emmi06]:
"$L(\mathcal{A}) \subseteq L(\mathcal{B})$ is decidable if $\mathcal{A}$ is a timed pushdown automaton, and $\mathcal{B}$ is a timed automaton with at most one clock." (Proof not correct!)

Corollary.

The language inclusion problem for pushdown timed automata is undecidable, even if $\mathcal{B}$ is a timed automaton with at most one clock.

[Emmi06] Emmi, Majumdar: Decision Problems for the Verification of Real-Time Software, 2006.

Consequences of the Undecidability Result (2)

Theorem.

1. The language inclusion problem is undecidable, even if $\mathcal{A}$ is deterministic and uses no clocks, and $\mathcal{B}$ is a timed automaton with at most one clock.

- Recall the last step of the proof sketch:

"Construct timed automaton $\mathcal{B}$ with one clock that recognizes the complement of $L(\mathcal{C}, q)$:

$$L(\mathcal{A}) \cap L(\mathcal{C}, q) = \emptyset \Leftrightarrow L(\mathcal{A}) \subseteq \overline{L(\mathcal{C}, q)} \Leftrightarrow L(\mathcal{A}) \subseteq L(\mathcal{B})\text{"}$$

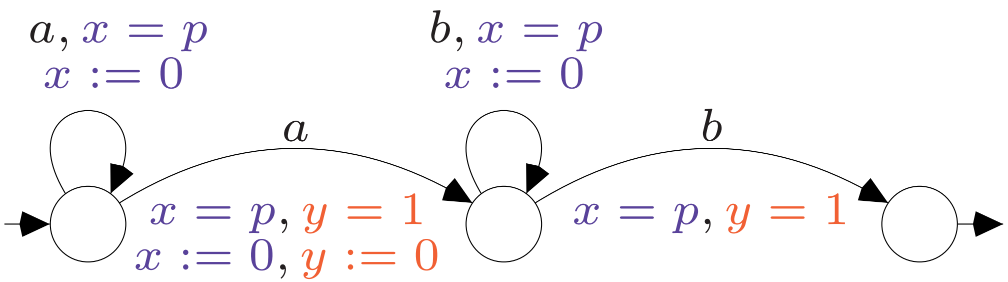
- We can construct an MTL formula φ such that $L(\mathcal{C}, q) = L(\varphi)$.

Theorem.

The MTL model checking problem for timed one-counter nets is undecidable, even if the net is deterministic and uses no clock.

- c.f. decidability of the MTL model checking problem for timed automata

Parametric Timed Automata [AHV93]



a parametric clock

- is a special clock
- can be compared with parameters
- a parameter valuation determines the behaviour of the automaton

- **Emptiness**: decidable if $\mathcal{A}$ uses ≤ 1 parametric clock, undecidable if $\mathcal{A}$ uses ≥ 3 parametric clocks.

[AHV93] Alur, Henzinger, Vardi: Parametric real-time reasoning, 1993.

MTL Model Checking of Parametric Timed Automata

Theorem.

1. The MTL model checking problem for parametric timed automata is undecidable, even if $\mathcal{A}$ is deterministic and uses one parametric clock.

Proof. (Sketch)

- Reduction of the (undecidable) reachability problem for channel machines
- Given a channel machine $\mathcal{C}$ and a state q, we can define a timed language $L(\mathcal{C}, q)$ that encodes computations of channel machines with insertion errors [OW06]
- Construct a parametric timed automaton $\mathcal{A}$ to exclude insertion errors:

$$\mathcal{C} \text{ does not reach } q \Leftrightarrow L(\mathcal{A}) \cap L(\mathcal{C}, q) = \emptyset$$

- Construct MTL formula φ such that $L(\varphi) = L(\mathcal{C}, q)$:

$$L(\mathcal{A}) \cap L(\varphi) = \emptyset \Leftrightarrow L(\mathcal{A}) \subseteq \overline{L(\varphi)} \Leftrightarrow L(\mathcal{A}) \subseteq L(\neg\varphi)$$

[OW06] Ouaknine, Worrell: On Metric Temporal Logic and Faulty Turing Machines, 2006.

Details of the Undecidability Proof

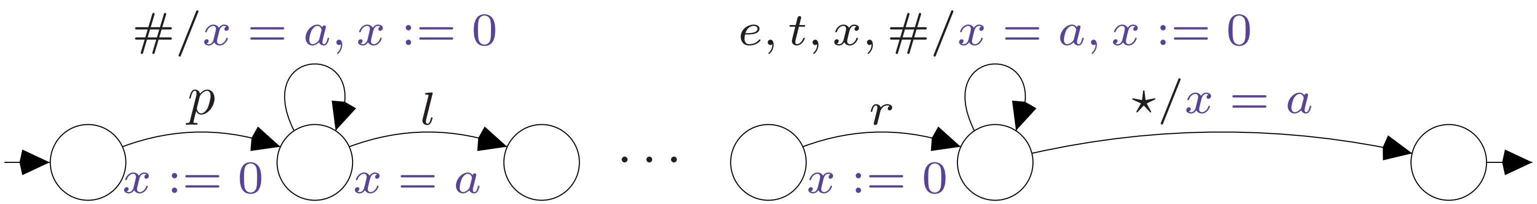
φ : Channel machine $M = (\{p,q,r\}, p, \{e,t,x\}, \Delta)$, $(p, !t, q), (q, ?e, q), ... \in \Delta$

Initial configuration (p, ε) is encoded by $(p,0)(\#,\delta_1)\ldots(\#,\delta_n)$, where $0 < \delta_1 < \cdots < \delta_n < 1$ for some $n \in \mathbb{N}$.

The transition $\langle(p, tex), !t, (q, text)\rangle$ may be encoded by

$(p,6)(t,6.1)(e,6.15)(x,6.5)(\#,6.73)(!t,7)(q,8)(t,8.1)(e,8.15)(x,8.5)(t,8.73)\ldots$

$\mathcal{A}$:



Open Problems

- Parametric Timed Automata:
  - What if the parameters may only take values in the non-negative integers?
  - MTL model checking for L/U-automata [BIT09]

- Is universality for timed visibly pushdown automata [Emmi06] really undecidable?

[BIT09] Bozzelli, La Torre: Decision Problems for lower/upper bound parametric timed automata, 2009.

[Emmi06] Emmi, Majumdar: Decision Problems for the Verification of Real-Time Software, 2006.