Layered controller synthesis for dynamic multi-agent systems

Emily Clement[1] Nicolas Perrin-Gilbert[1] Philipp Schlehuber-Caissier[2]

[1]Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France
[2]EPITA Research Laboratory

March 16 2023

Introduction

Dynamic multi-agent system's verification

Our objectives

https://perso.eleves.ens-rennes.fr/people/Emily.Clement/Videos/example_episodes/ex_0.mp4

Issues of different methods

- Timed Automata: issues to $\cdots$
  1) represent speed variation
  2) scales to be executed in real-time

Our objectives

https://perso.eleves.ens-rennes.fr/people/Emily.Clement/Videos/example_episodes/ex_0.mp4

Issues of different methods

- Timed Automata: issues to $\cdots$
  1) represent speed variation
  2) scales to be executed in real-time
- Reinforcement Learning:
  1) combinatorial aspects
  2) continuous aspects

Our objectives

https://perso.eleves.ens-rennes.fr/people/Emily.Clement/Videos/example_episodes/ex_0.mp4

Issues of different methods

- Timed Automata: issues to ···
  1) represent speed variation
  2) scales to be executed in real-time
- Reinforcement Learning:
  1) combinatorial aspects
  2) continuous aspects

Our solution

- Solve a (simplified) model with an efficient Timed Automata reachability algorithm

Our objectives

https://perso.eleves.ens-rennes.fr/people/Emily.Clement/Videos/example_episodes/ex_0.mp4

Issues of different methods

- Timed Automata: issues to ···
  1) represent speed variation
  2) scales to be executed in real-time
- Reinforcement Learning:
  1) combinatorial aspects
  2) continuous aspects

Our solution

- Solve a (simplified) model with an efficient Timed Automata reachability algorithm
- Relax the simplification assumption for the speed changes using an SMT solver

Our objectives

https://perso.eleves.ens-rennes.fr/people/Emily.Clement/Videos/example_episodes/ex_0.mp4

Issues of different methods

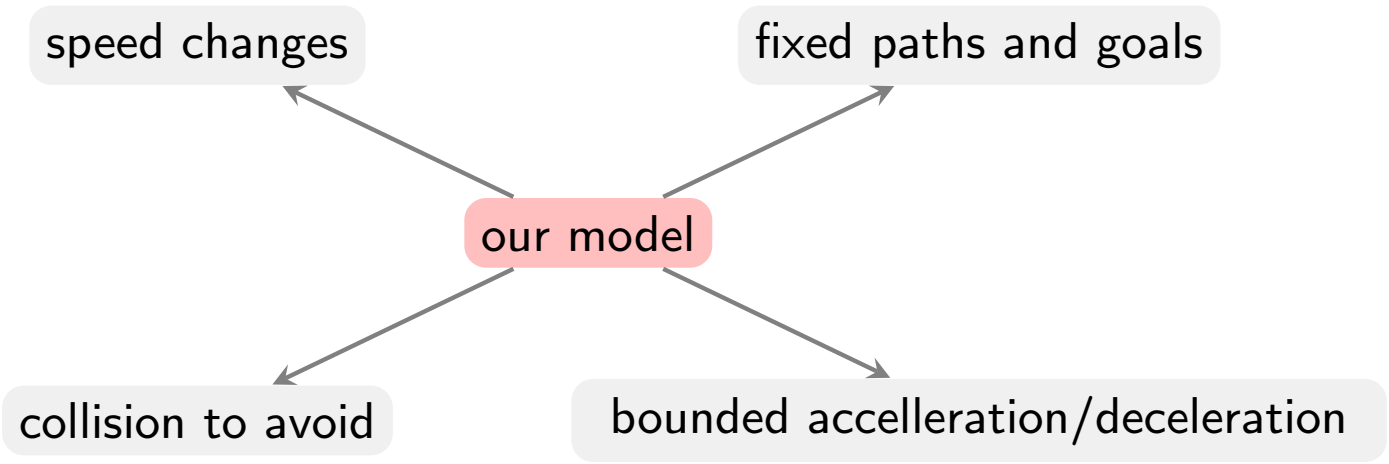
- Timed Automata: issues to ···
  1) represent speed variation
  2) scales to be executed in real-time
- Reinforcement Learning:
  1) combinatorial aspects
  2) continuous aspects

Our solution

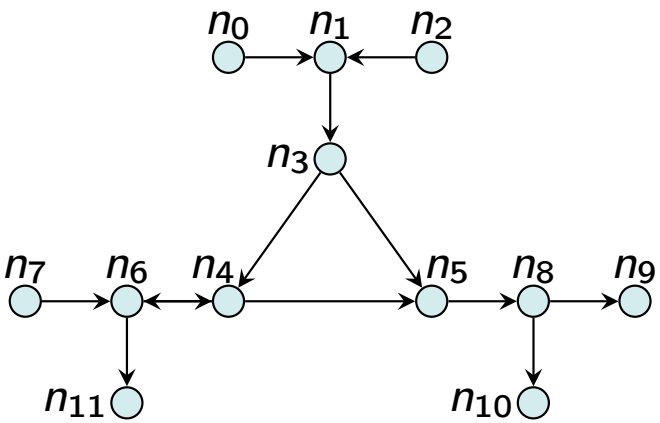
- Solve a (simplified) model with an efficient Timed Automata reachability algorithm
- Relax the simplification assumption for the speed changes using an SMT solver
- Generate an SWA-SMT solver to help RL solving this problem.

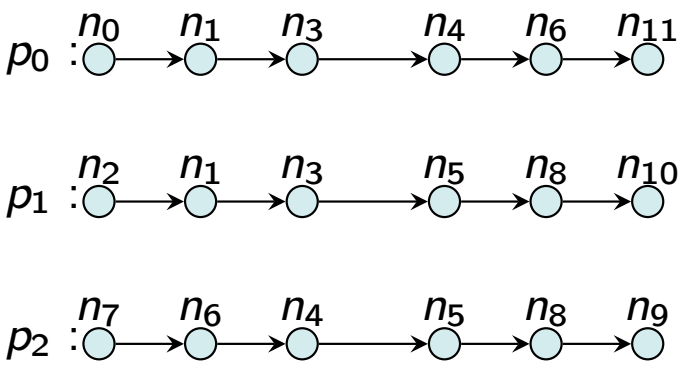
Dynamic multi-agent system

- Our model



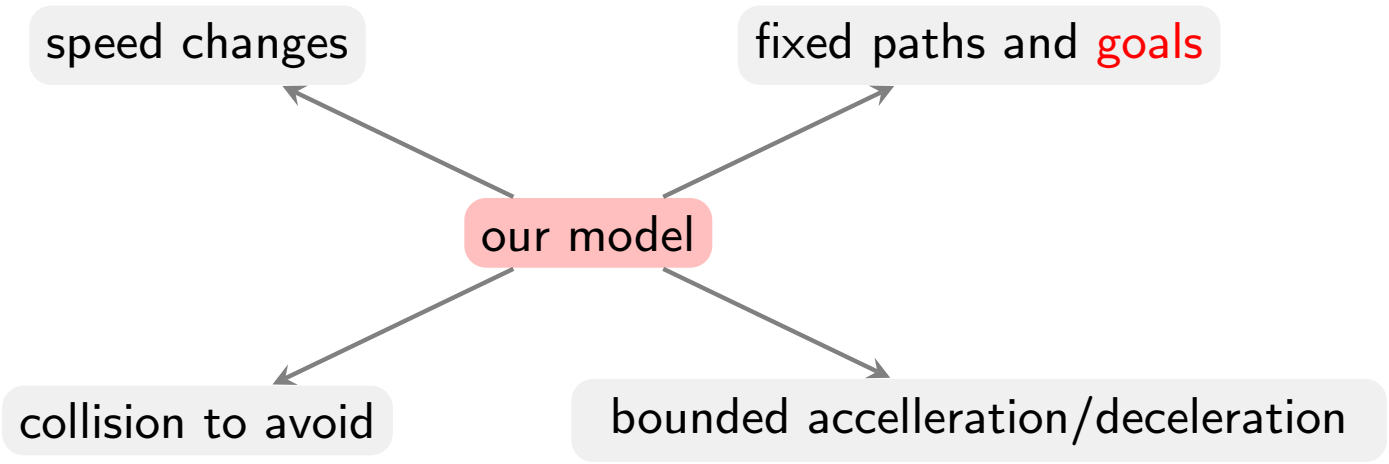
- Example of traffic (left) and associated paths p_0, p_1, p_2 (right)



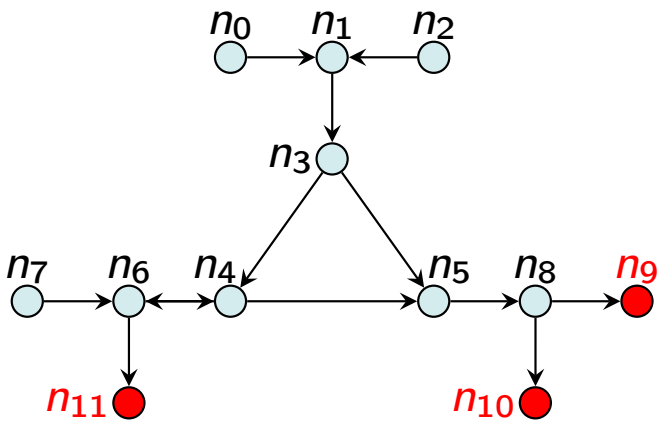


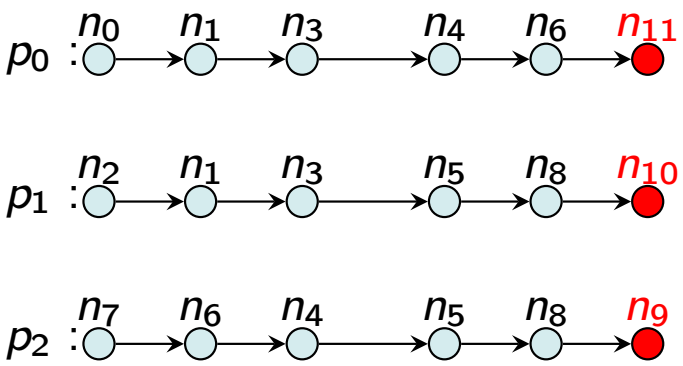
Dynamic multi-agent system

- Our model

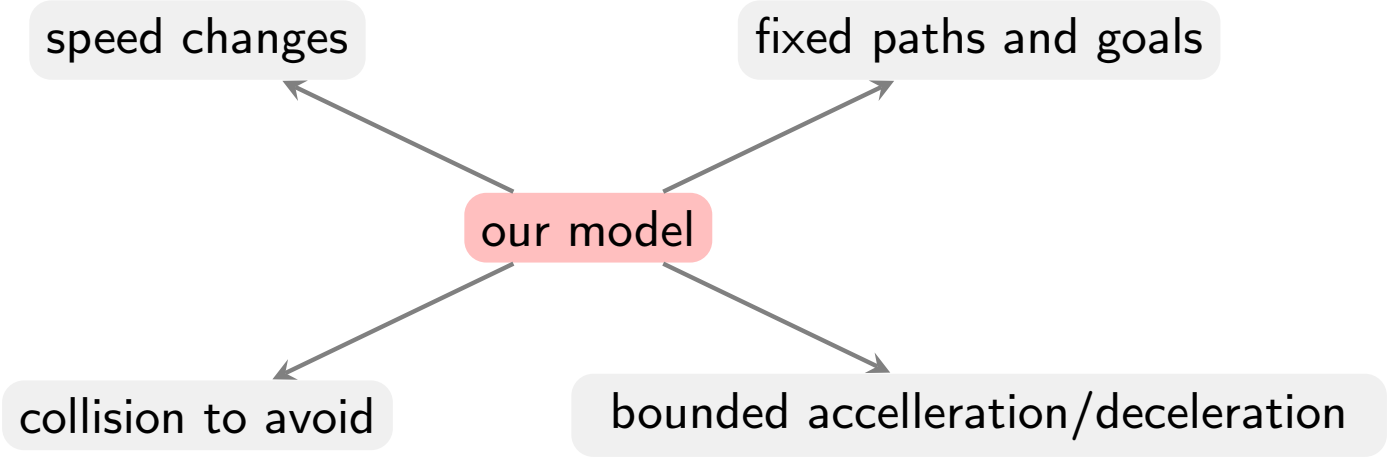


- Example of traffic (left) and associated paths p_0, p_1, p_2 (right)



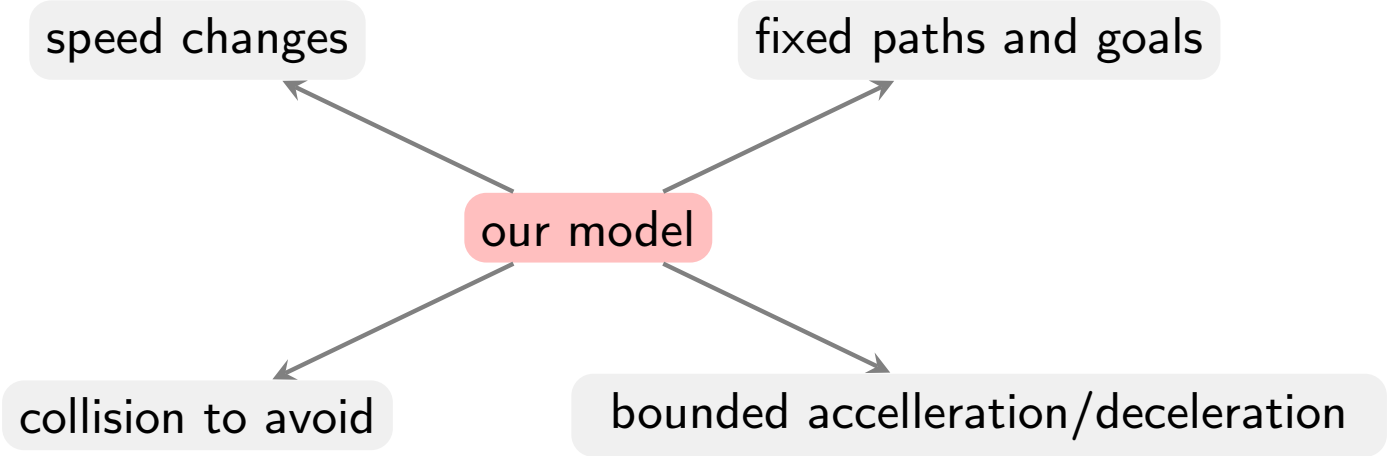


Dynamic multi-agent system

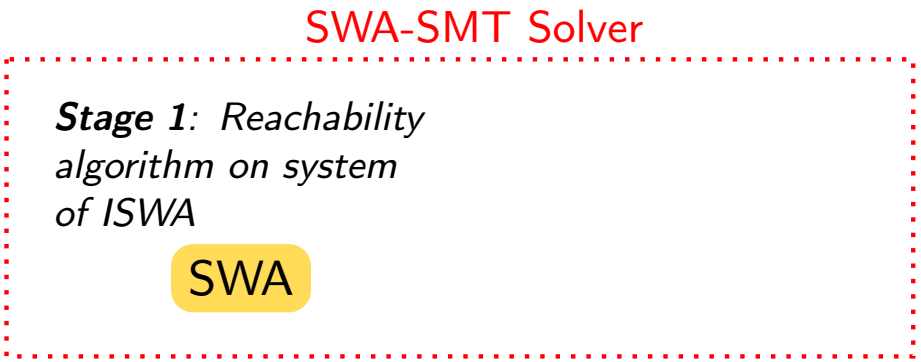


- Our contribution: Controller synthesis
  - ▷ Goal: reach goals while avoiding collisions between agents

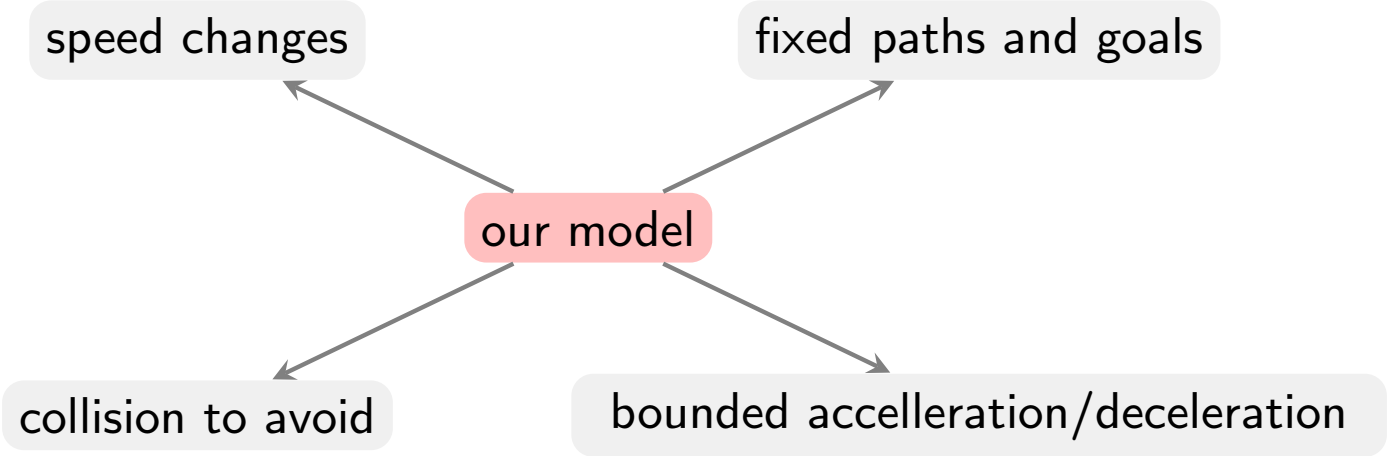
Dynamic multi-agent system



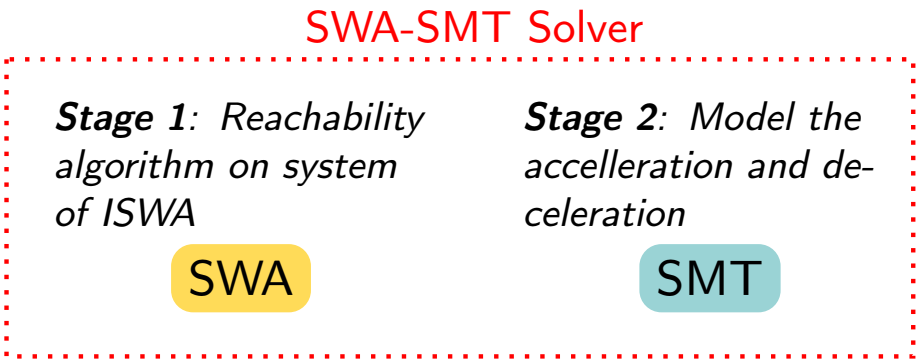
- Our contribution: Controller synthesis
  - ▷ Goal: reach goals while avoiding collisions between agents
  - ▷ **Three-layer** Method: SWA-SMT solver + RL Training:



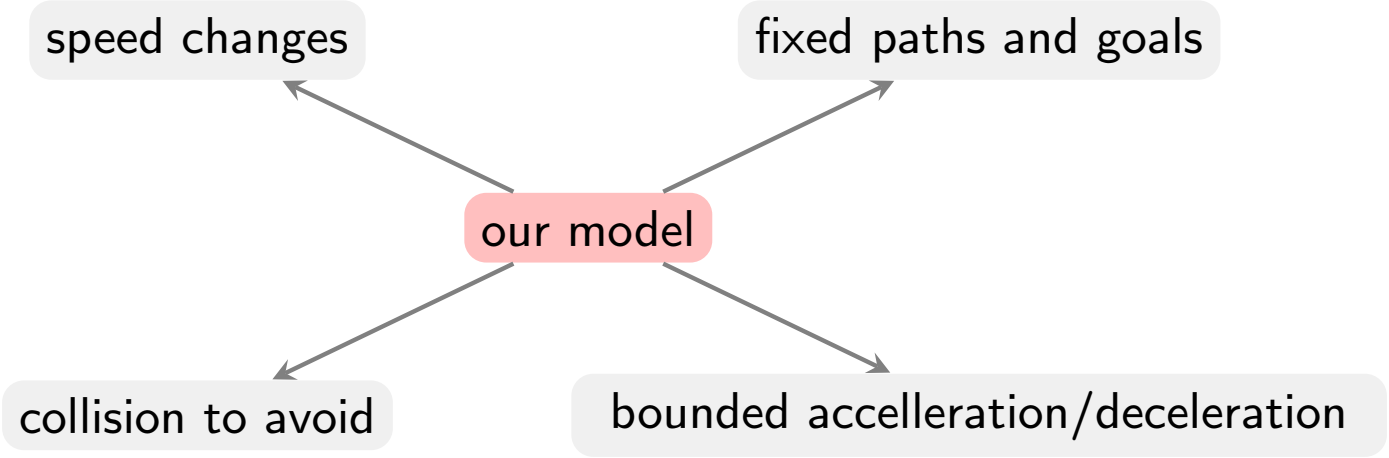
Dynamic multi-agent system



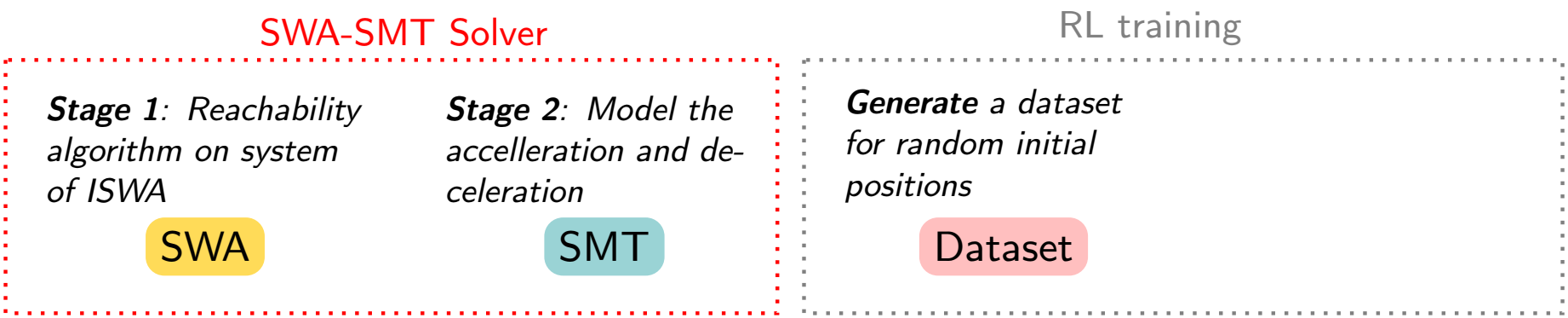
- Our contribution: Controller synthesis
  - ▷ Goal: reach goals while avoiding collisions between agents
  - ▷ **Three-layer** Method: SWA-SMT solver + RL Training:



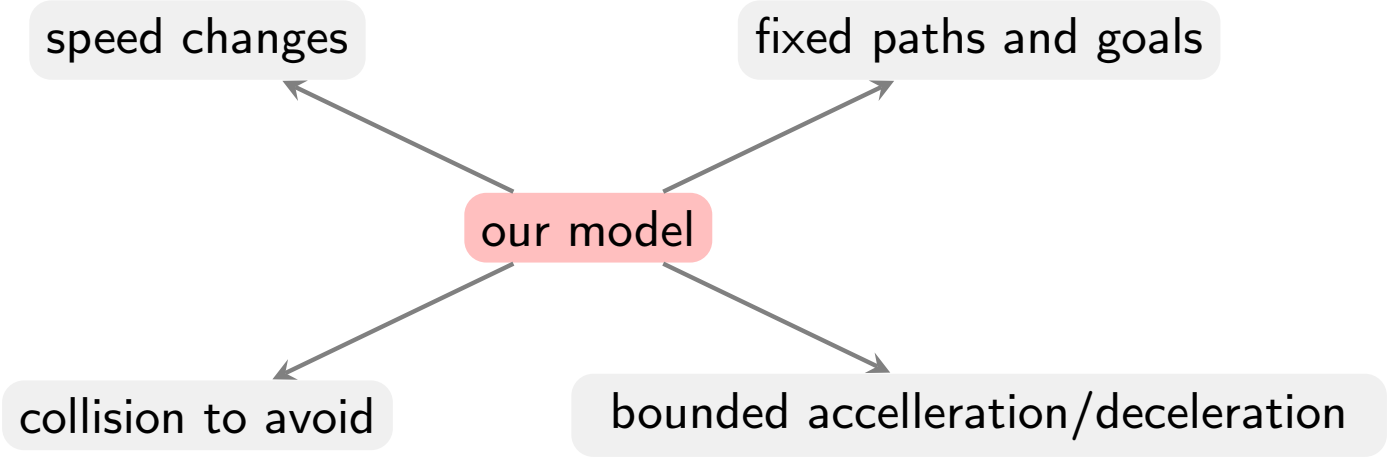
Dynamic multi-agent system



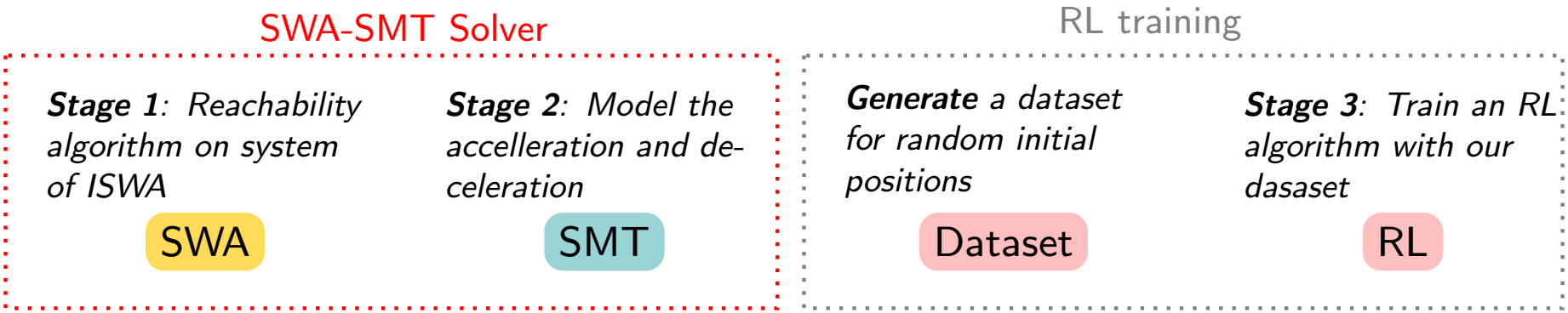
- Our contribution: Controller synthesis
  - ▷ Goal: reach goals while avoiding collisions between agents
  - ▷ **Three-layer** Method: SWA-SMT solver + RL Training:



Dynamic multi-agent system

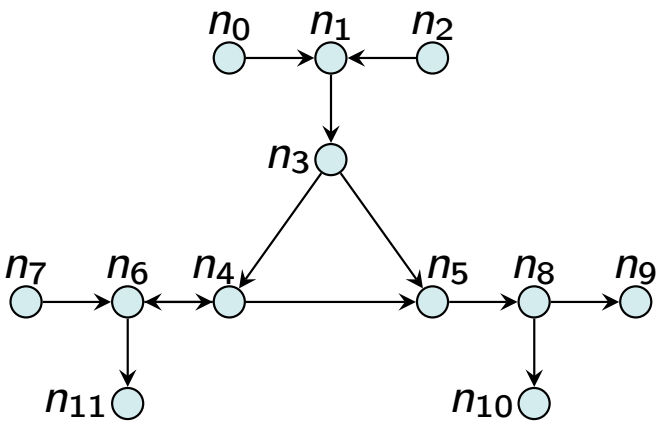


- Our contribution: Controller synthesis
  - Goal: reach goals while avoiding collisions between agents
  - **Three-layer** Method: SWA-SMT solver + RL Training:



How to model a Car Traffic ?

- A point in $\mathbb{R}^2$: a node n_0
- A section $s_{[n_0,n_1],L}$ of the road: $n_0 \xrightarrow{\quad L \quad} n_1$
- A path: $p_0 : n_0 \to n_1 \to n_3 \to n_4 \to n_6 \to n_{11}$
- A car traffic: c_0, c_1, c_2 are each assigned paths p_0, p_1, p_2:



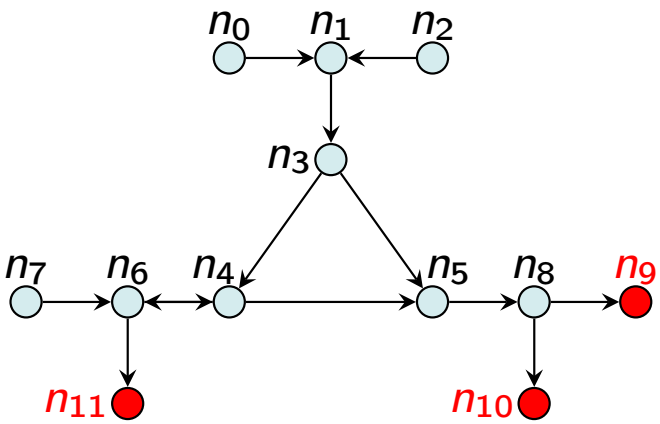
$p_0 : n_0 \to n_1 \to n_3 \to n_4 \to n_6 \to n_{11}$

$p_1 : n_2 \to n_1 \to n_3 \to n_5 \to n_8 \to n_{10}$

$p_2 : n_7 \to n_6 \to n_4 \to n_5 \to n_8 \to n_9$

How to model a Car Traffic ?

- A point in $\mathbb{R}^2$: a node n_0
- A section $s_{[n_0,n_1],L}$ of the road: $n_0 \xrightarrow{\quad L \quad} n_1$
- A path: $p_0 : n_0 \to n_1 \to n_3 \to n_4 \to n_6 \to n_{11}$
- A car traffic: c_0, c_1, c_2 are each assigned paths p_0, p_1, p_2:



$p_0 : n_0 \to n_1 \to n_3 \to n_4 \to n_6 \to n_{11}$

$p_1 : n_2 \to n_1 \to n_3 \to n_5 \to n_8 \to n_{10}$

$p_2 : n_7 \to n_6 \to n_4 \to n_5 \to n_8 \to n_9$

Rules to model collision avoidance

- #1: security distance when driving in the same direction and between neighbouring sections
- #2: cars cannot share a section if driving in **opposite** direction
- #3: No Overtaking between cars

SWA-SMT solver

SWA solver

***Stage 1**: Reachability algorithm on system of ISWA*

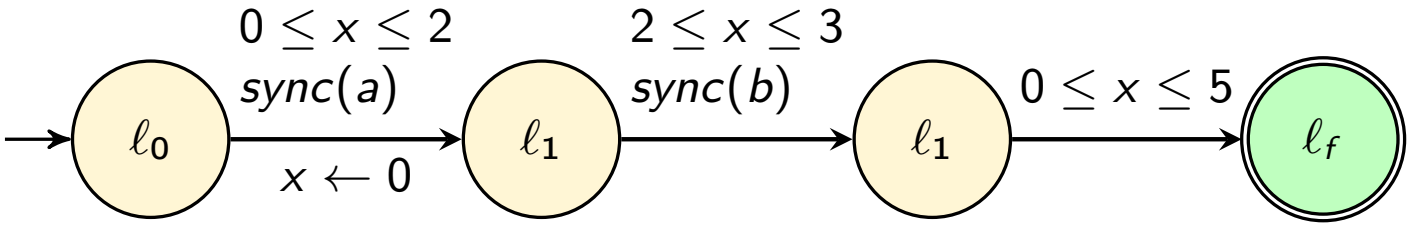
SWA

***Stage 2**: Model the accelleration and deceleration*

SMT

What is a Timed Automaton (and its variants) ?

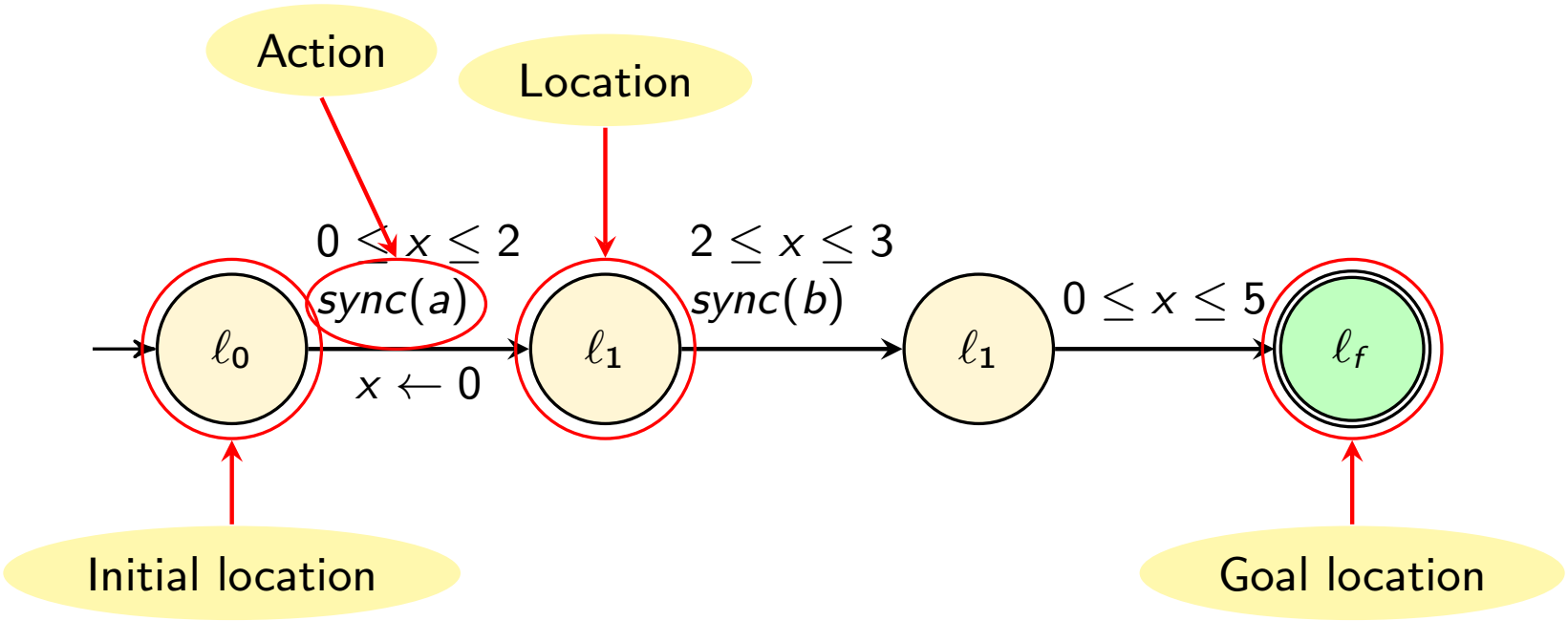
- A classic timed automaton



- Variants

What is a Timed Automaton (and its variants) ?

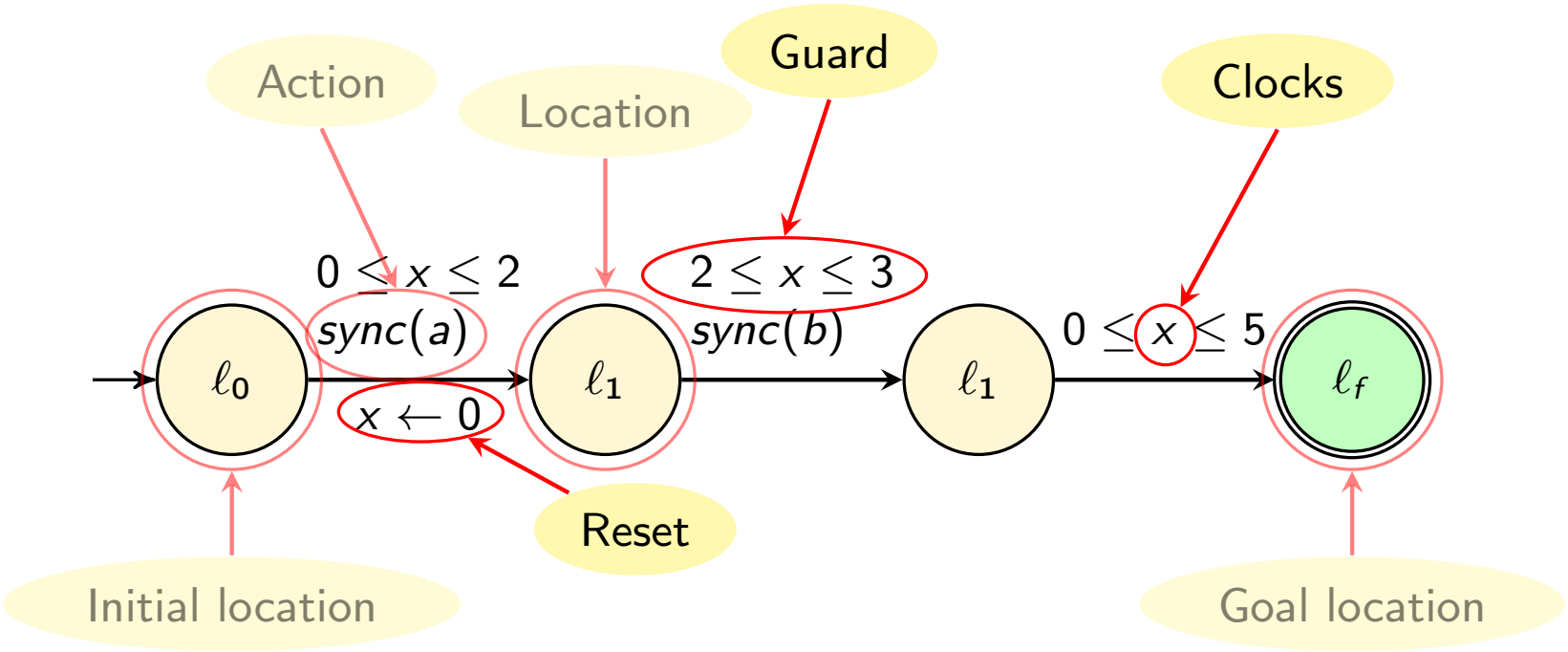
- A classic timed automaton



- Variants

What is a Timed Automaton (and its variants) ?

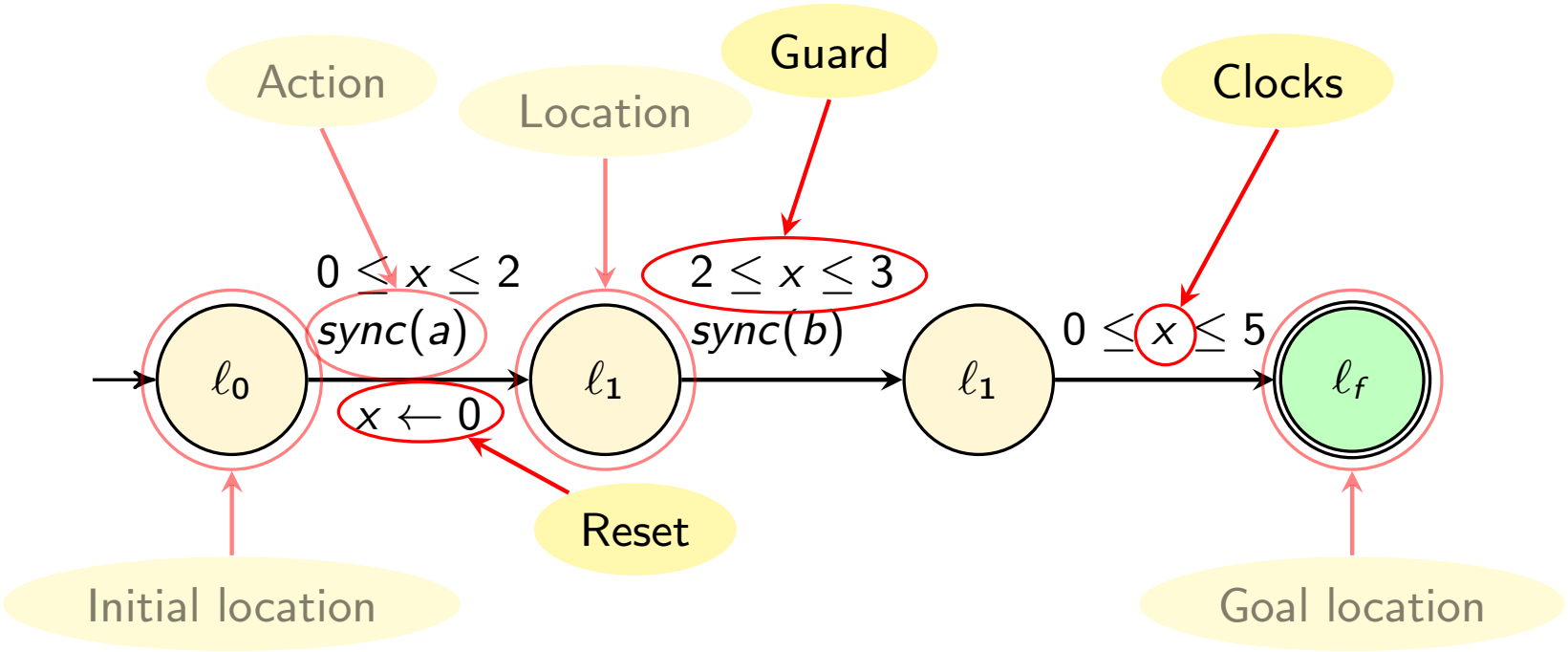
- A classic timed automaton

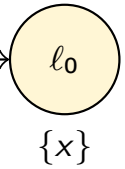


- Variants

What is a Timed Automaton (and its variants) ?

- A classic timed automaton

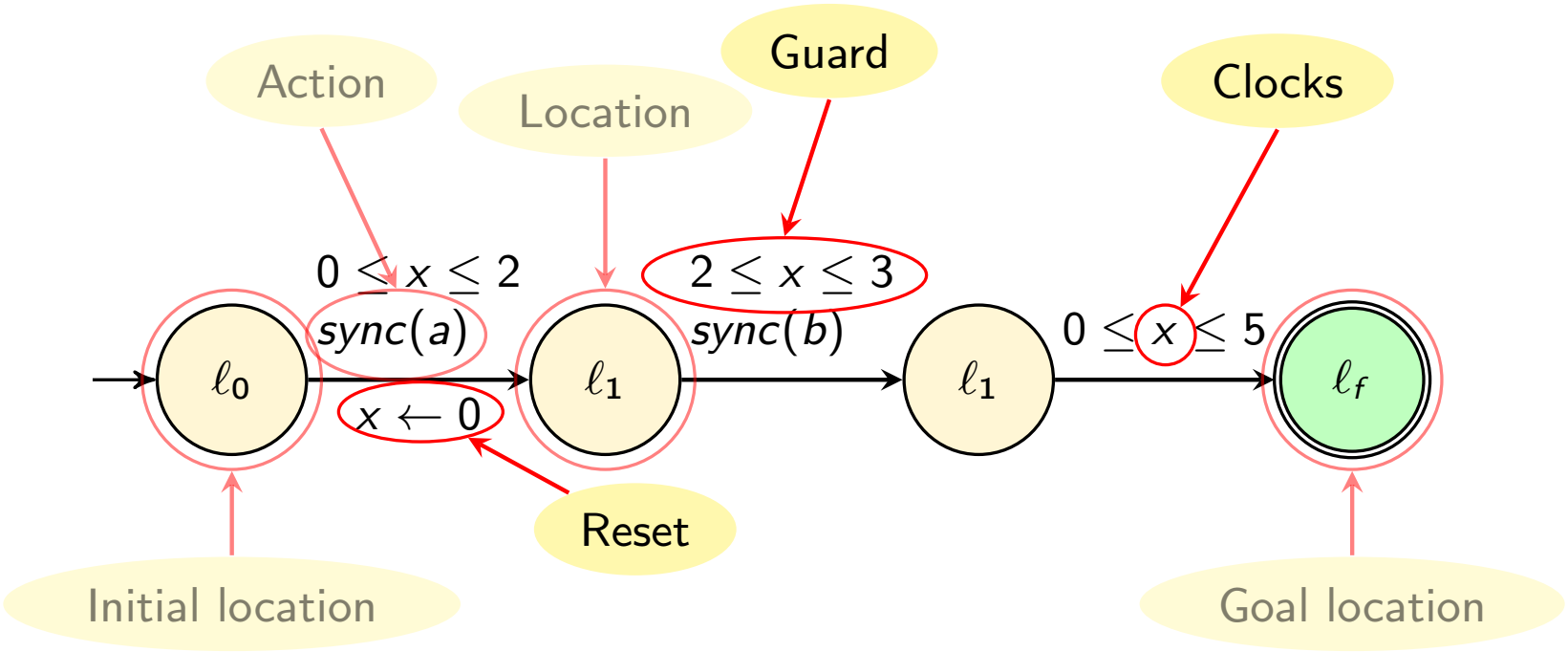


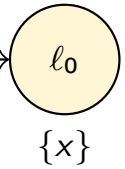
- Variants
  - ▷ Stopwatch 

: clock x is stopped in location ℓ_0.

What is a Timed Automaton (and its variants) ?

- A classic timed automaton

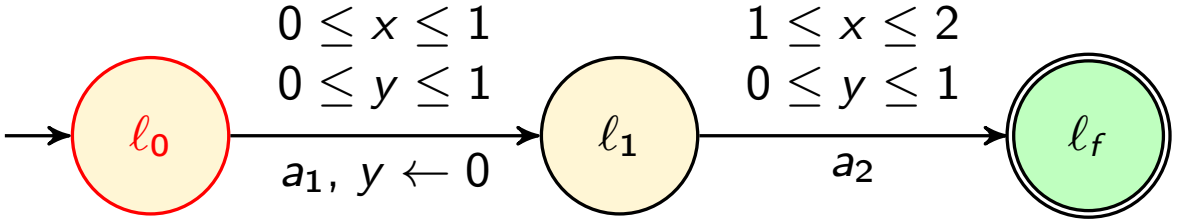


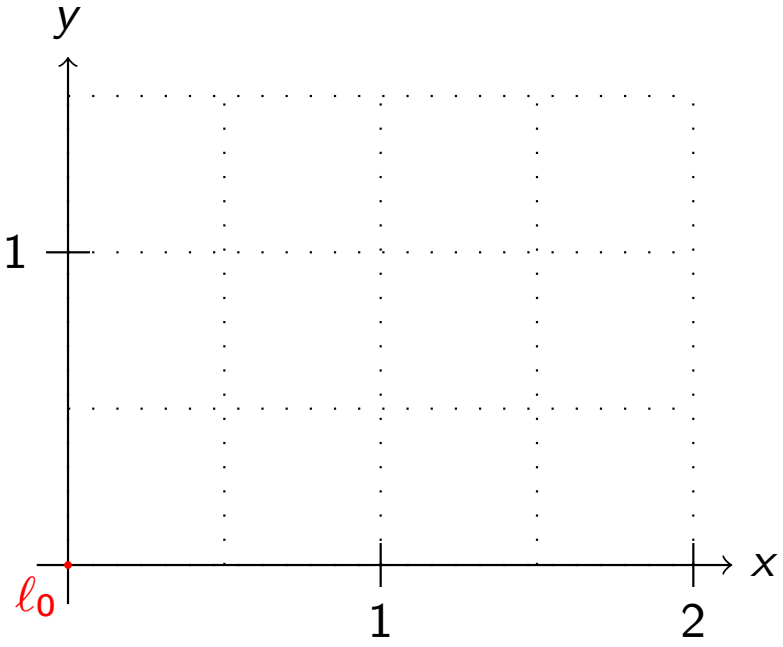
- Variants
  - ▷ Stopwatch 

: clock x is stopped in location ℓ_0.
  - ▷ Channels: FiFo queue of symbols (actions) to be pushed/read

A run of a timed automaton

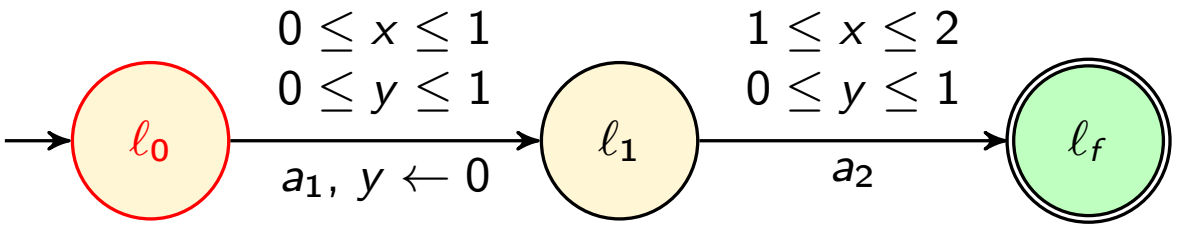
- A run of a two-clock Timed Automaton

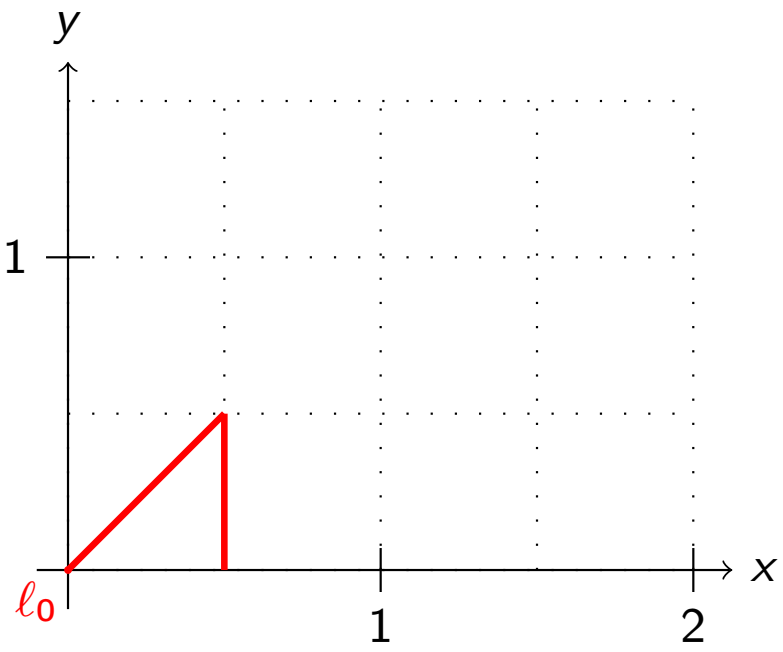




A run of a timed automaton

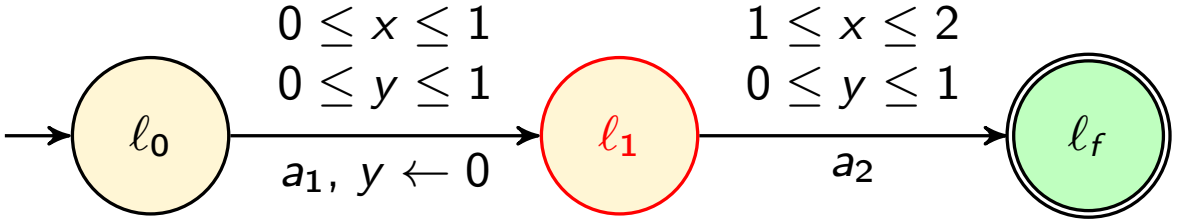
- A run of a two-clock Timed Automaton

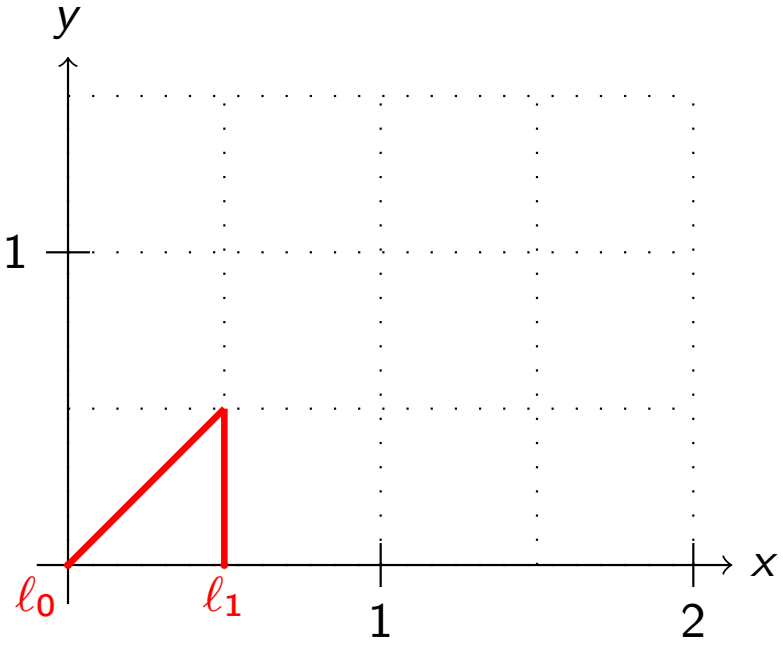




A run of a timed automaton

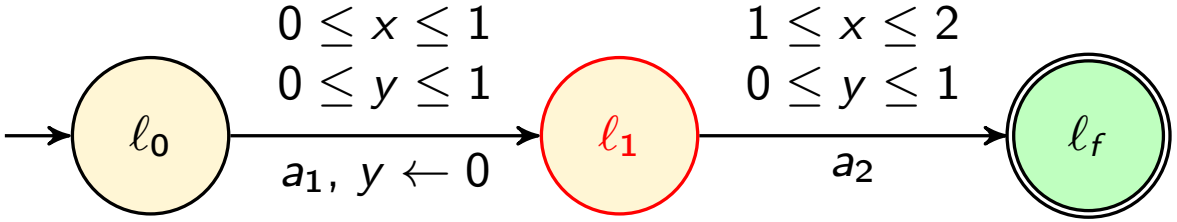
- A run of a two-clock Timed Automaton

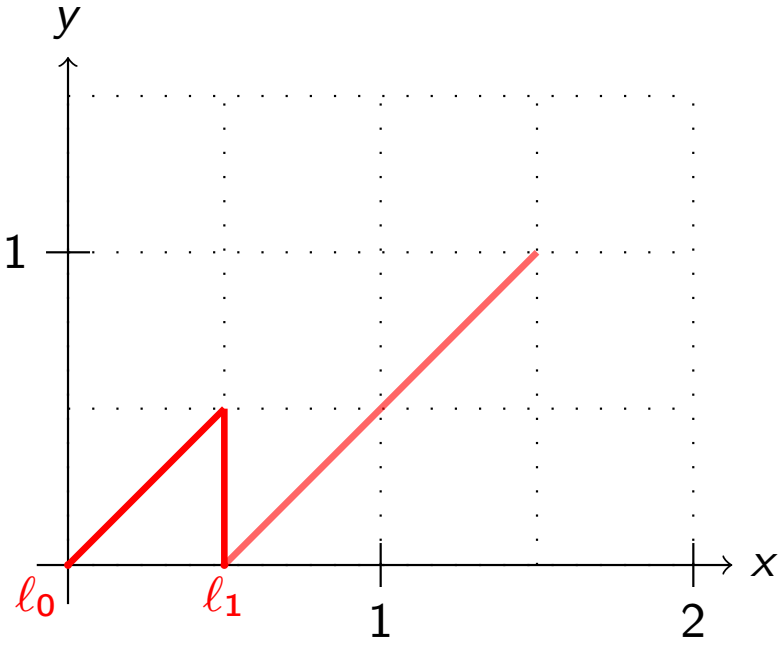




A run of a timed automaton

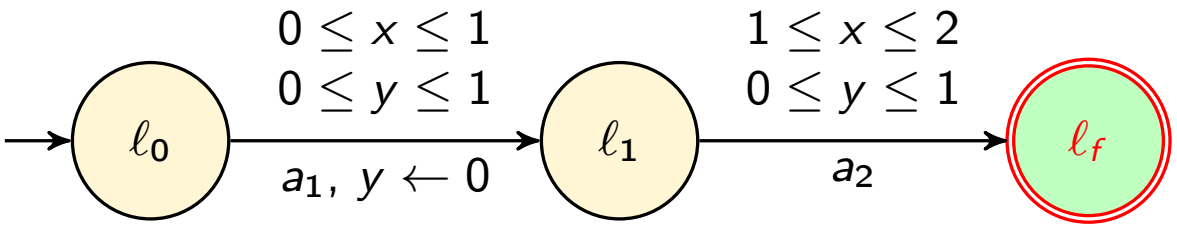
- A run of a two-clock Timed Automaton

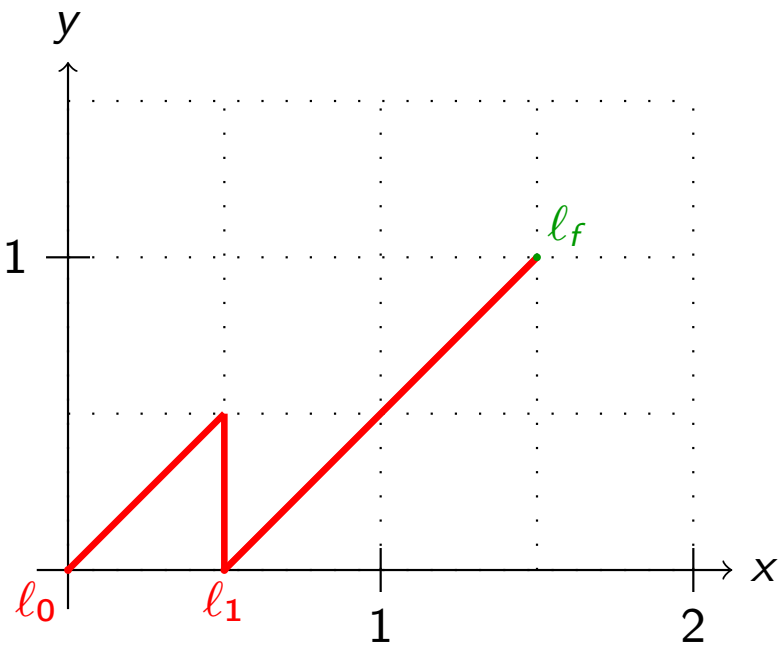




A run of a timed automaton

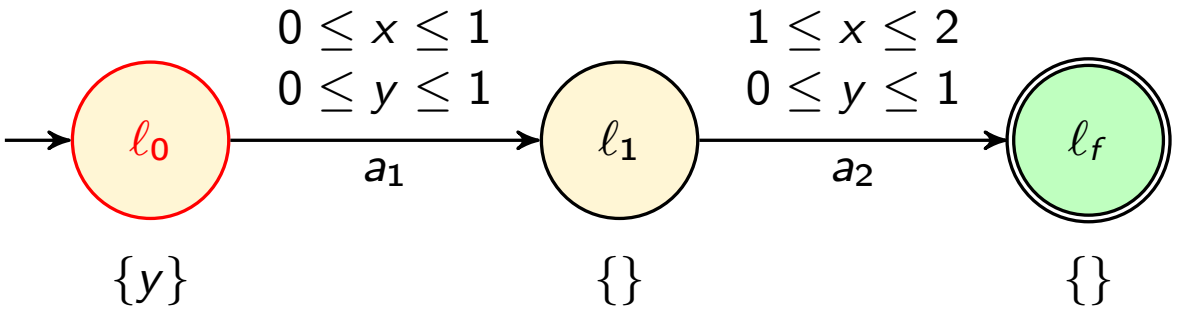
- A run of a two-clock Timed Automaton

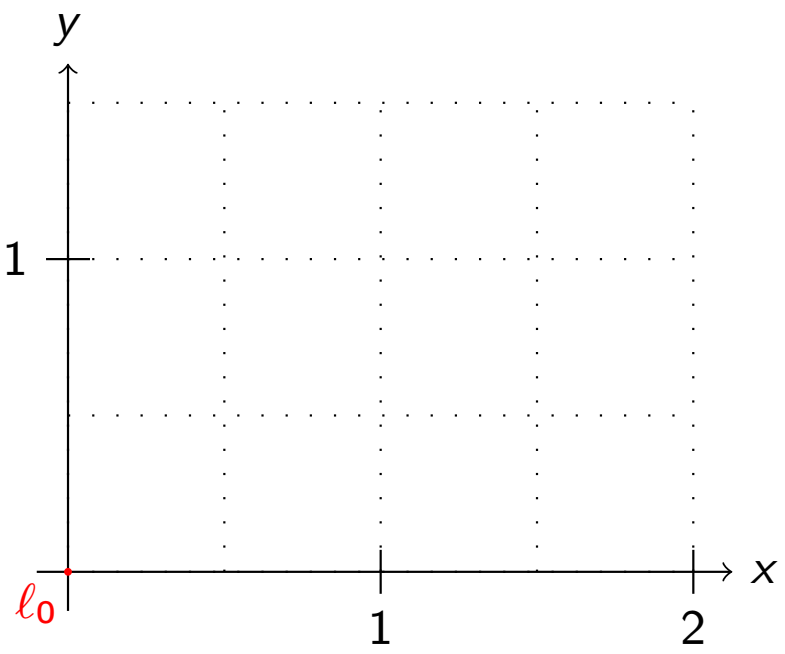




A run of a timed automaton

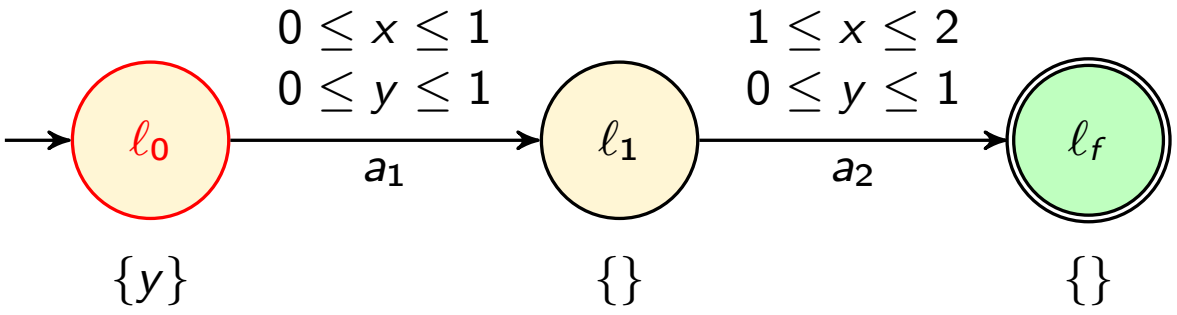
- A run with a (two-clock) stopwatch timed automaton (ISWA)

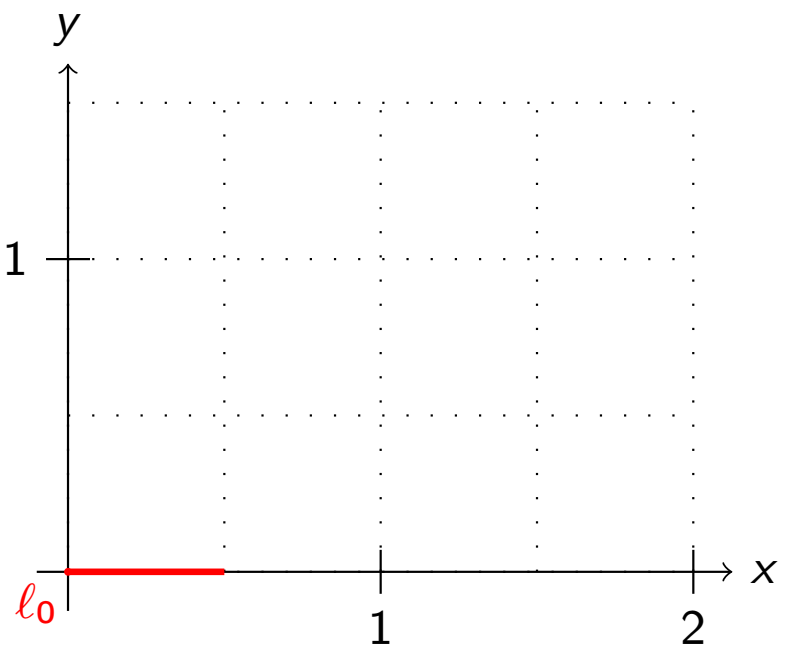




A run of a timed automaton

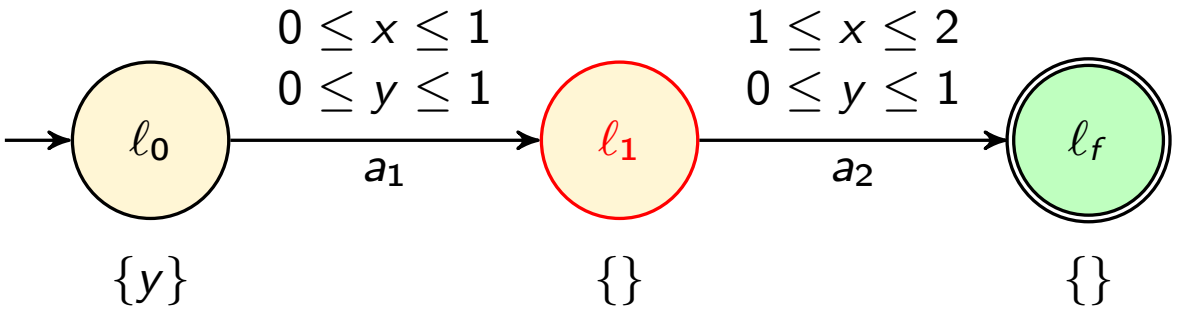
- A run with a (two-clock) stopwatch timed automaton (ISWA)

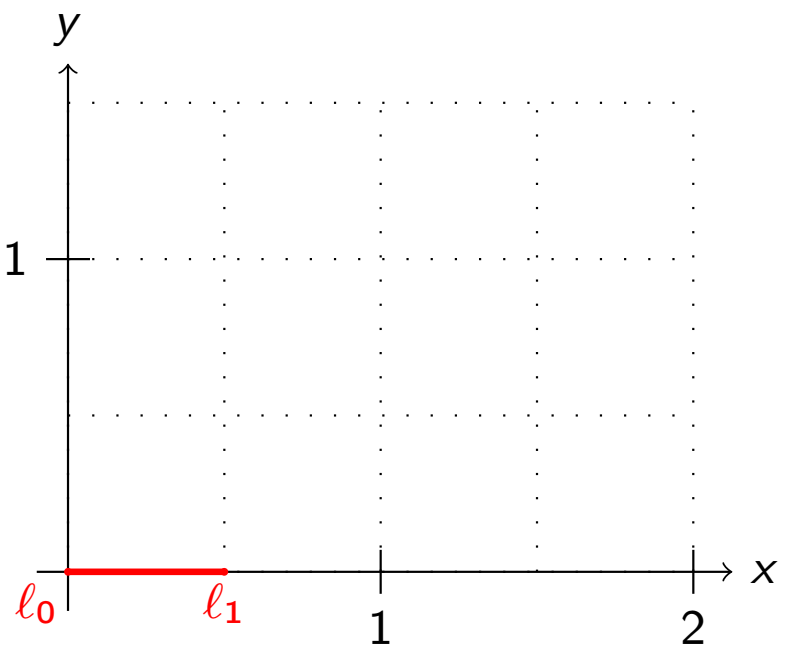




A run of a timed automaton

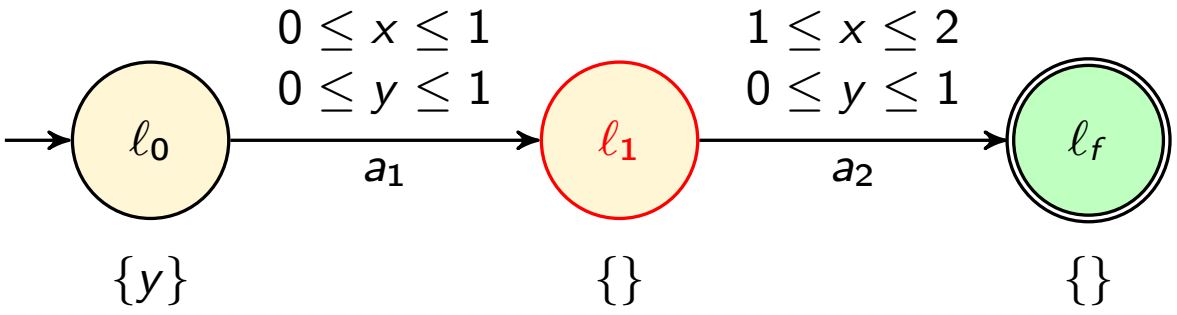
- A run with a (two-clock) stopwatch timed automaton (ISWA)

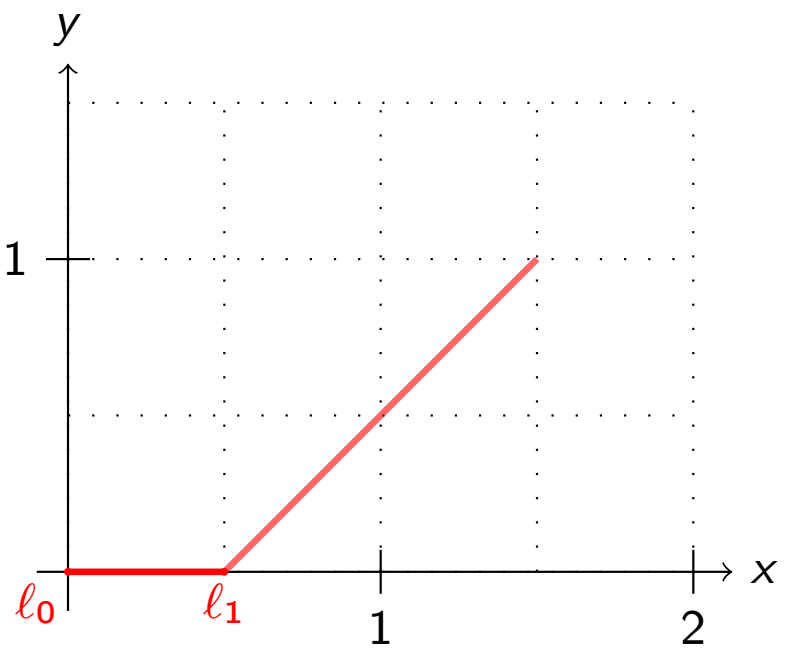




A run of a timed automaton

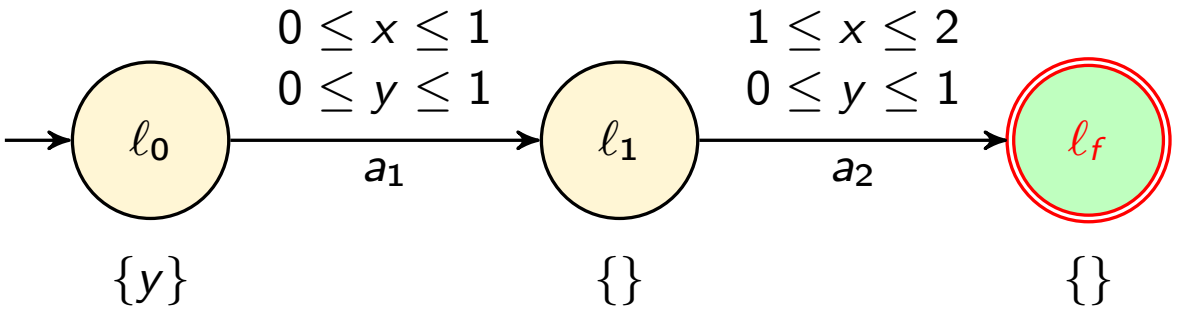
- A run with a (two-clock) stopwatch timed automaton (ISWA)

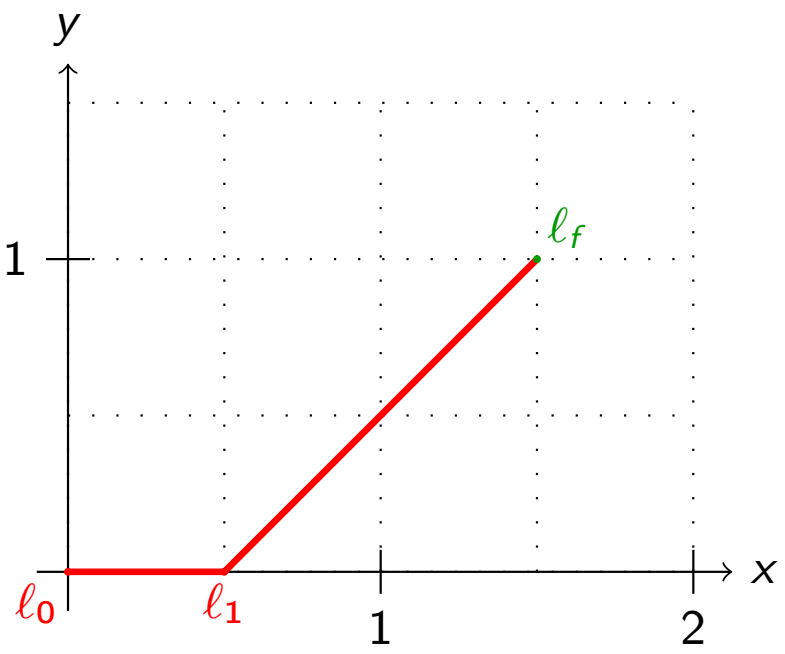




A run of a timed automaton

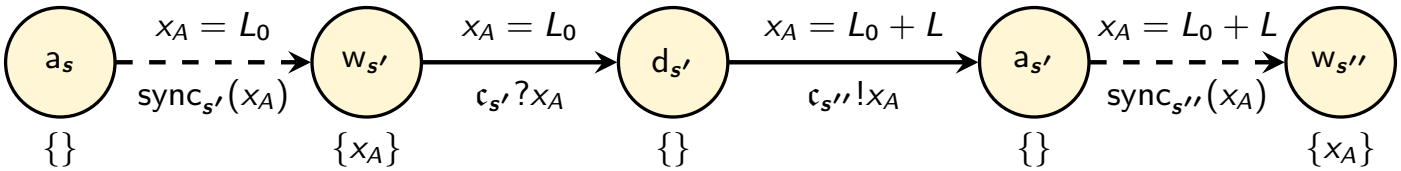
- A run with a (two-clock) stopwatch timed automaton (ISWA)





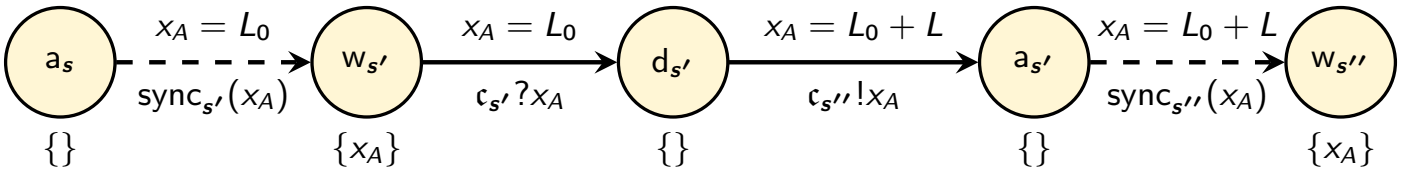
Automata-based Modelisation: car progress

- Car A progress along its paths
  - x_A: distance travelled along its paths



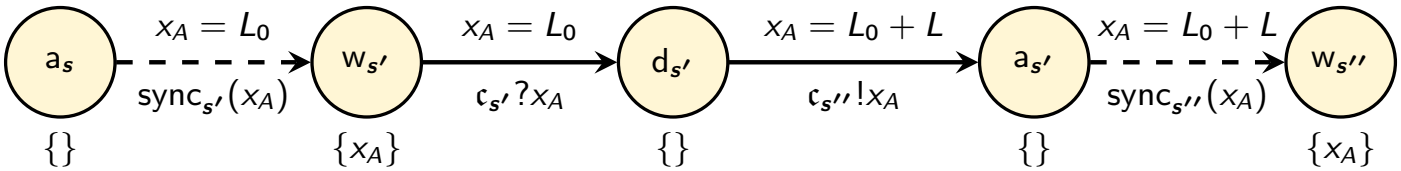
Automata-based Modelisation: car progress

- Car A progress along its paths
  - x_A: distance travelled along its paths
  - **Stopwatches** $\{x_A\}$: x_A stops/the car stops $\Rightarrow$: car stop instantly.

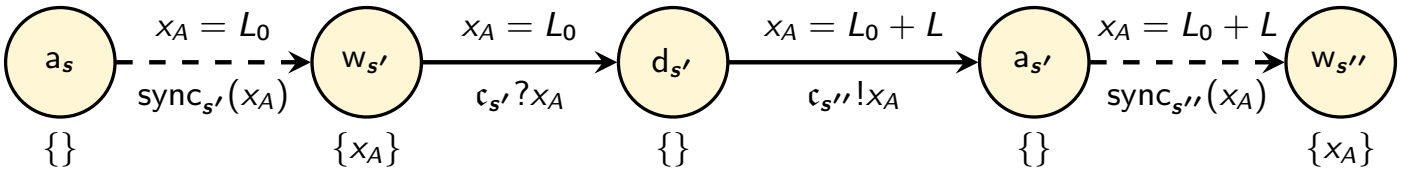


Automata-based Modelisation: car progress

- Car A progress along its paths
  - x_A: distance travelled along its paths
  - **Stopwatches** $\{x_A\}$: x_A stops/the car stops $\Rightarrow$: car stop instantly.
  - **Channels** $\mathfrak{c}_{s'}!x_A/\mathfrak{c}_{s'}?x_A$: respect of order of cars in a section $s \Rightarrow$ no overtaking.

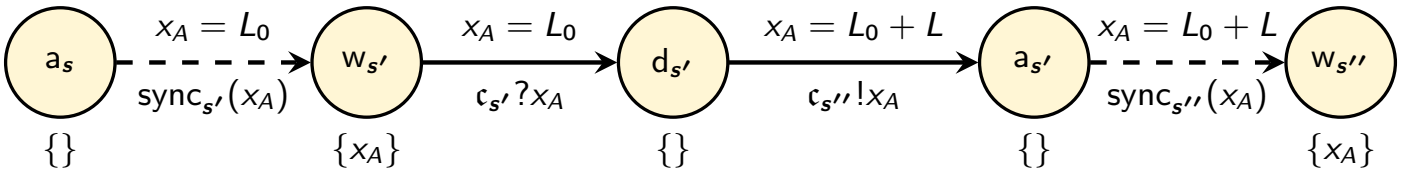


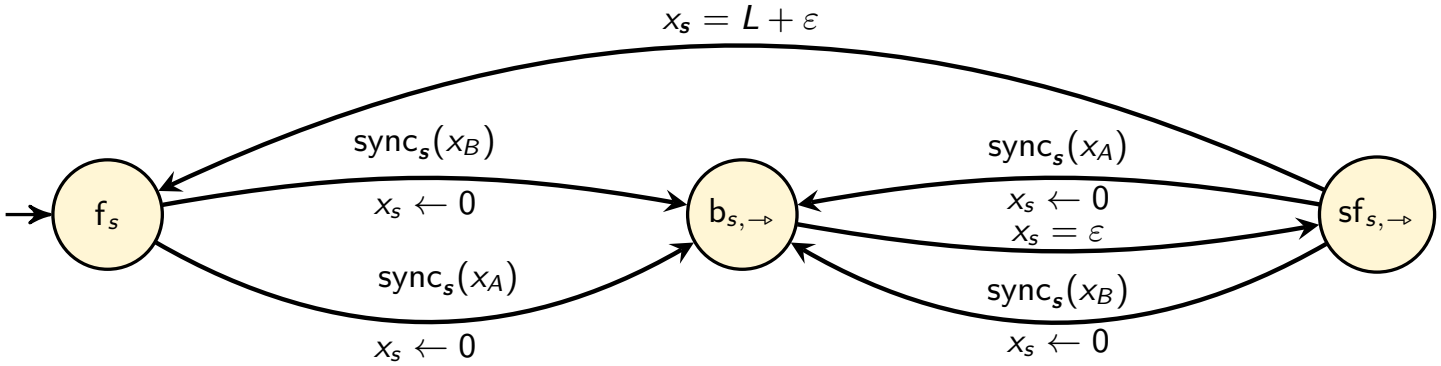
- Car A progress along its paths
  - x_A: distance travelled along its paths
  - **Stopwatches** $\{x_A\}$: x_A stops/the car stops $\Rightarrow$: car stop instantly.
  - **Channels** $\mathfrak{c}_{s'}!x_A/\mathfrak{c}_{s'}?x_A$: respect of order of cars in a section $s \Rightarrow$ no overtaking.
  - **Intersection**: use classical synchronized action to active *intersection automata*



Automata-based Model: intersection

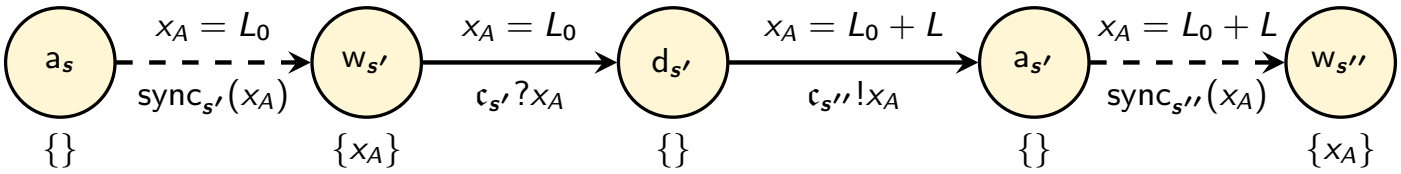
- Reminder: car automaton



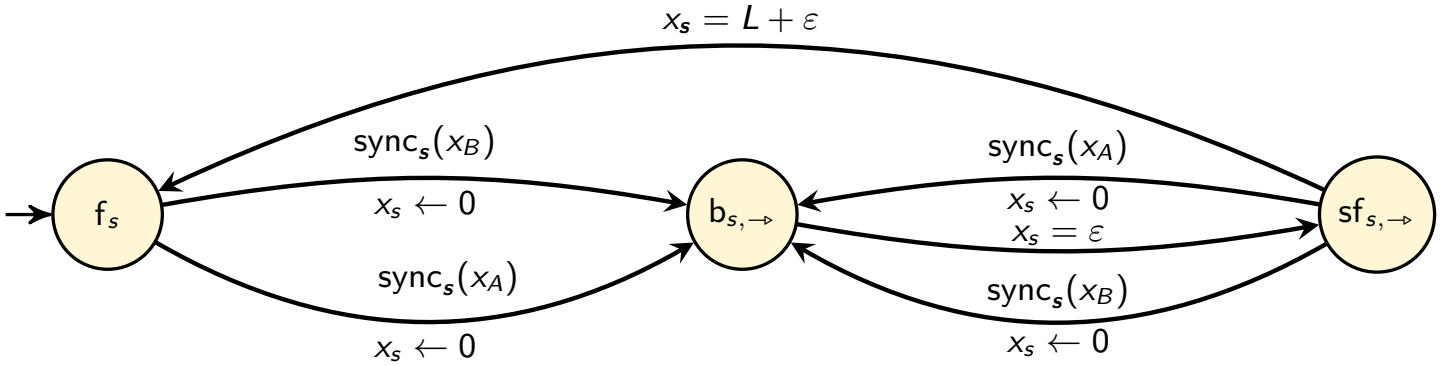


Automata-based Model: intersection

- Reminder: car automaton

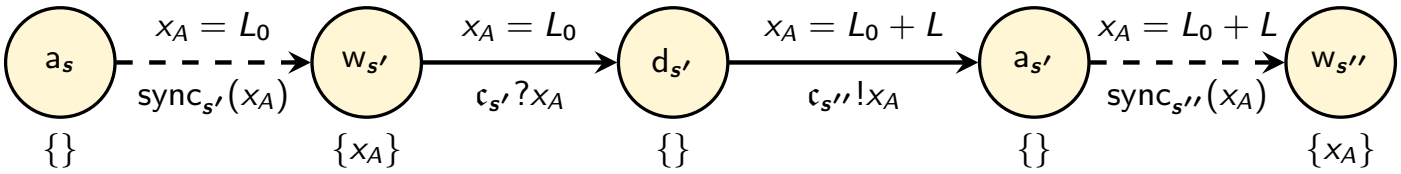


- Intersection automata
  - ▷ Each intersection **<u>tracks</u>** the progress of the last entered car along s

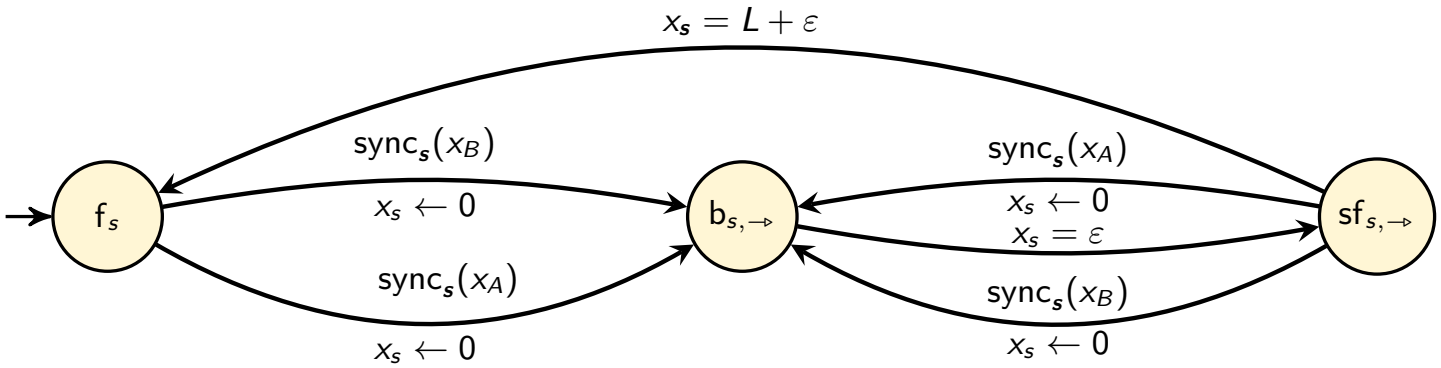


Automata-based Model: intersection

- Reminder: car automaton

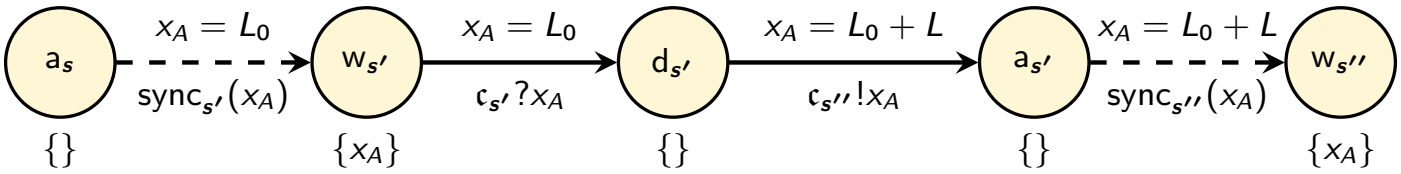


- Intersection automata
  - ▷ Each intersection **tracks** the progress of the last entered car along s
  - ▷ Goal: ensure that cars maintain a safe distance from each other (with **guards**)

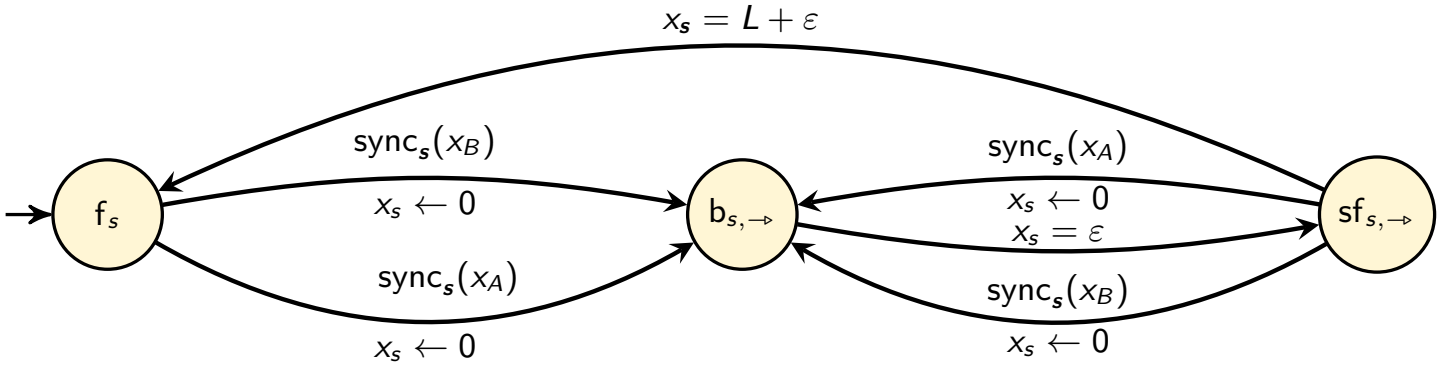


Automata-based Model: intersection

- Reminder: car automaton

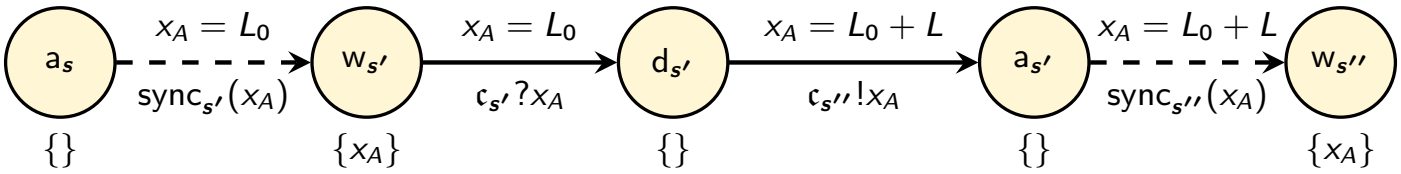


- Intersection automata
  - ▷ Each intersection **tracks** the progress of the last entered car along s
  - ▷ Goal: ensure that cars maintain a safe distance from each other (with **guards**)
  - ▷ For A to enter in s: need to active $\text{sync}_s(x_A)$

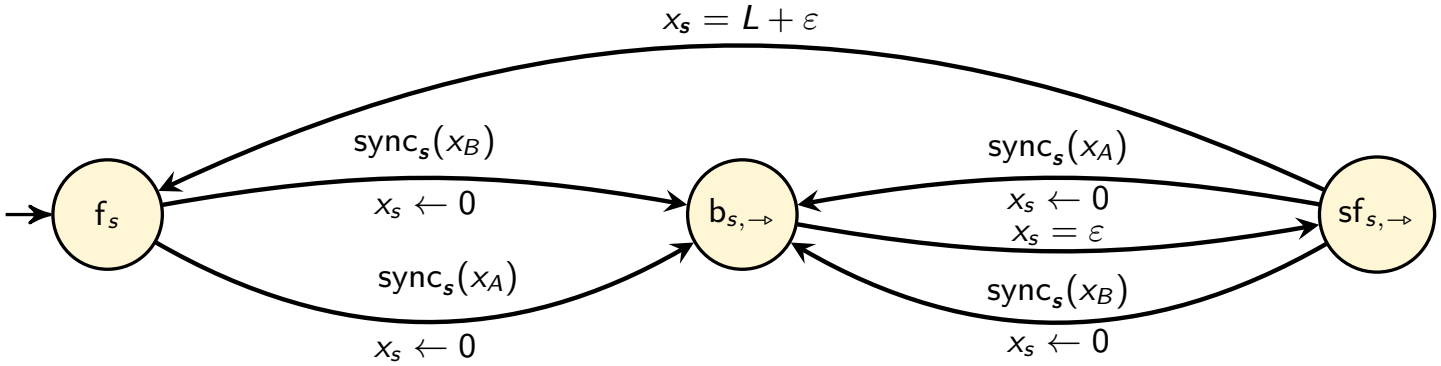


Automata-based Model: intersection

- Reminder: car automaton



- Intersection automata
  - ▷ Each intersection **tracks** the progress of the last entered car along s
  - ▷ Goal: ensure that cars maintain a safe distance from each other (with **guards**)
  - ▷ For A to enter in s: need to active $\text{sync}_s(x_A)$
  - ▷ Forbid cars to drive in both direction at the same time.



Goal: reachability algorithm

- Reachability for Timed Automata
  - ▷ **PSPACE**: equivalence with a untimed automaton

Goal: reachability algorithm

- Reachability for Timed Automata
  - ▷ **PSPACE**: equivalence with a untimed automaton
- Reachability for Stopwatch Timed Automata
  - ▷ **Undecidable** in general cases

Goal: reachability algorithm

- Reachability for Timed Automata
  - **PSPACE**: equivalence with a untimed automaton
- Reachability for Stopwatch Timed Automata
  - **Undecidable** in general cases
  - **Decidable** for Initialized Stopwatch Timed Automata (subclass)

Goal: reachability algorithm

- Reachability for Timed Automata
  - **PSPACE**: equivalence with a untimed automaton
- Reachability for Stopwatch Timed Automata
  - **Undecidable** in general cases
  - **Decidable** for Initialized Stopwatch Timed Automata (subclass)
- Reachability with channels...
  - **Undecidable** in general cases

Goal: reachability algorithm

- Reachability for Timed Automata
  - **PSPACE**: equivalence with a untimed automaton
- Reachability for Stopwatch Timed Automata
  - **Undecidable** in general cases
  - **Decidable** for Initialized Stopwatch Timed Automata (subclass)
- Reachability with channels...
  - **Undecidable** in general cases
  - **Decidable** for bounded channels (bounded number of symbols)

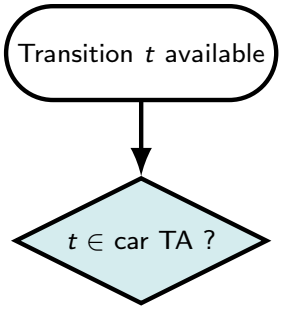
Goal: reachability algorithm

- Reachability for Timed Automata
  - **PSPACE**: equivalence with a untimed automaton
- Reachability for Stopwatch Timed Automata
  - **Undecidable** in general cases
  - **Decidable** for Initialized Stopwatch Timed Automata (subclass)
- Reachability with channels...
  - **Undecidable** in general cases
  - **Decidable** for bounded channels (bounded number of symbols)
- Our model and algorithm
  - Our Model: systems of **Initialized** SWA with *bounded channels*
  - They have a **specific form** (few cycles, stopwatches are always in the same locations...)

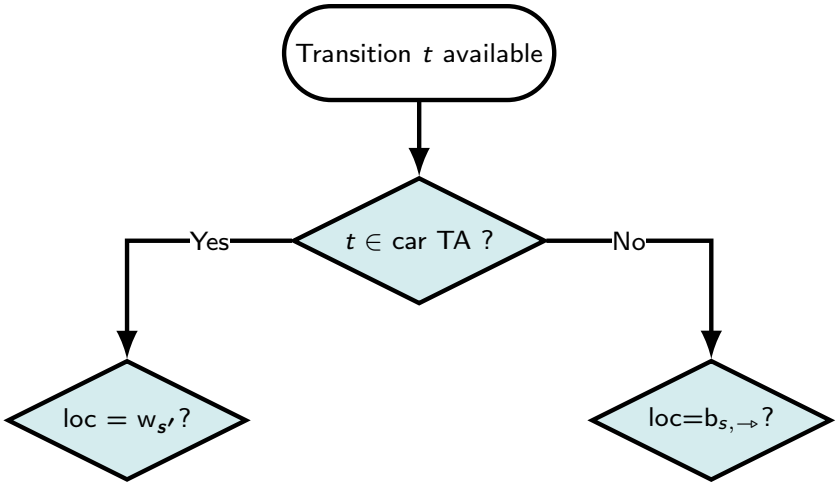
Goal: reachability algorithm

- Reachability for Timed Automata
  - **PSPACE**: equivalence with a untimed automaton
- Reachability for Stopwatch Timed Automata
  - **Undecidable** in general cases
  - **Decidable** for Initialized Stopwatch Timed Automata (subclass)
- Reachability with channels...
  - **Undecidable** in general cases
  - **Decidable** for bounded channels (bounded number of symbols)
- Our model and algorithm
  - Our Model: systems of **Initialized** SWA with *bounded channels*
  - They have a **specific form** (few cycles, stopwatches are always in the same locations...)
  - Our contribution: a specified reachability algorithm: a **DFS** with an optimised successor function.

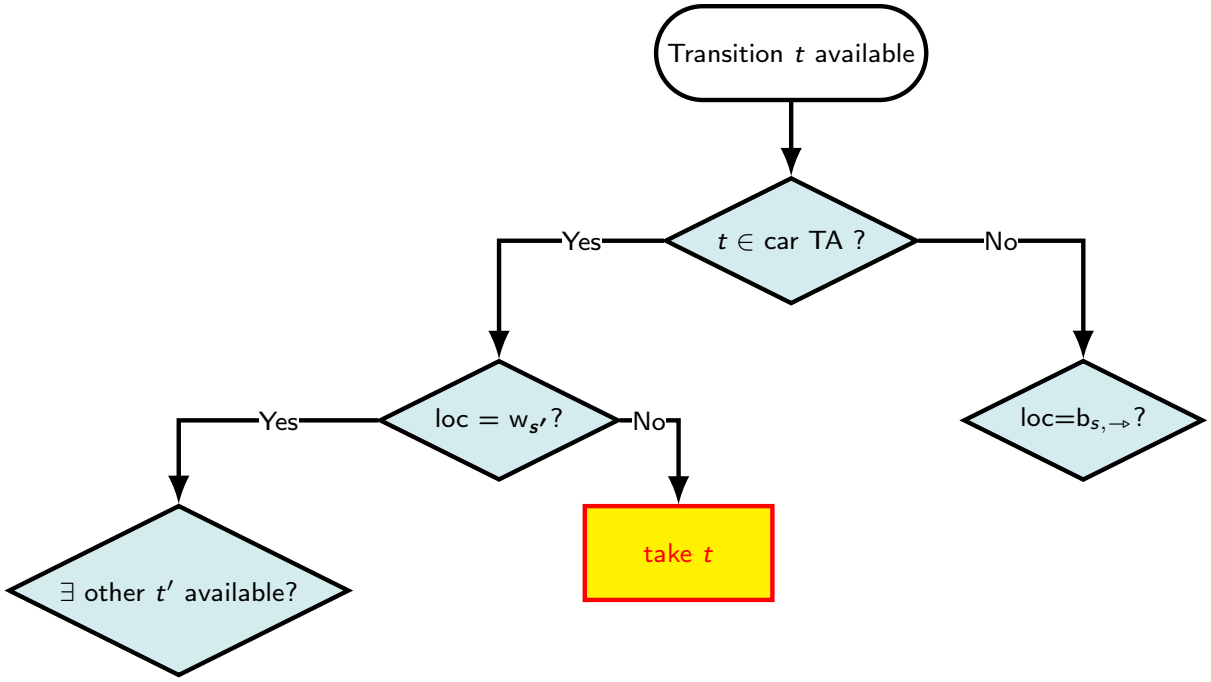
Our Algorithm: a DFS with an optimised succ function



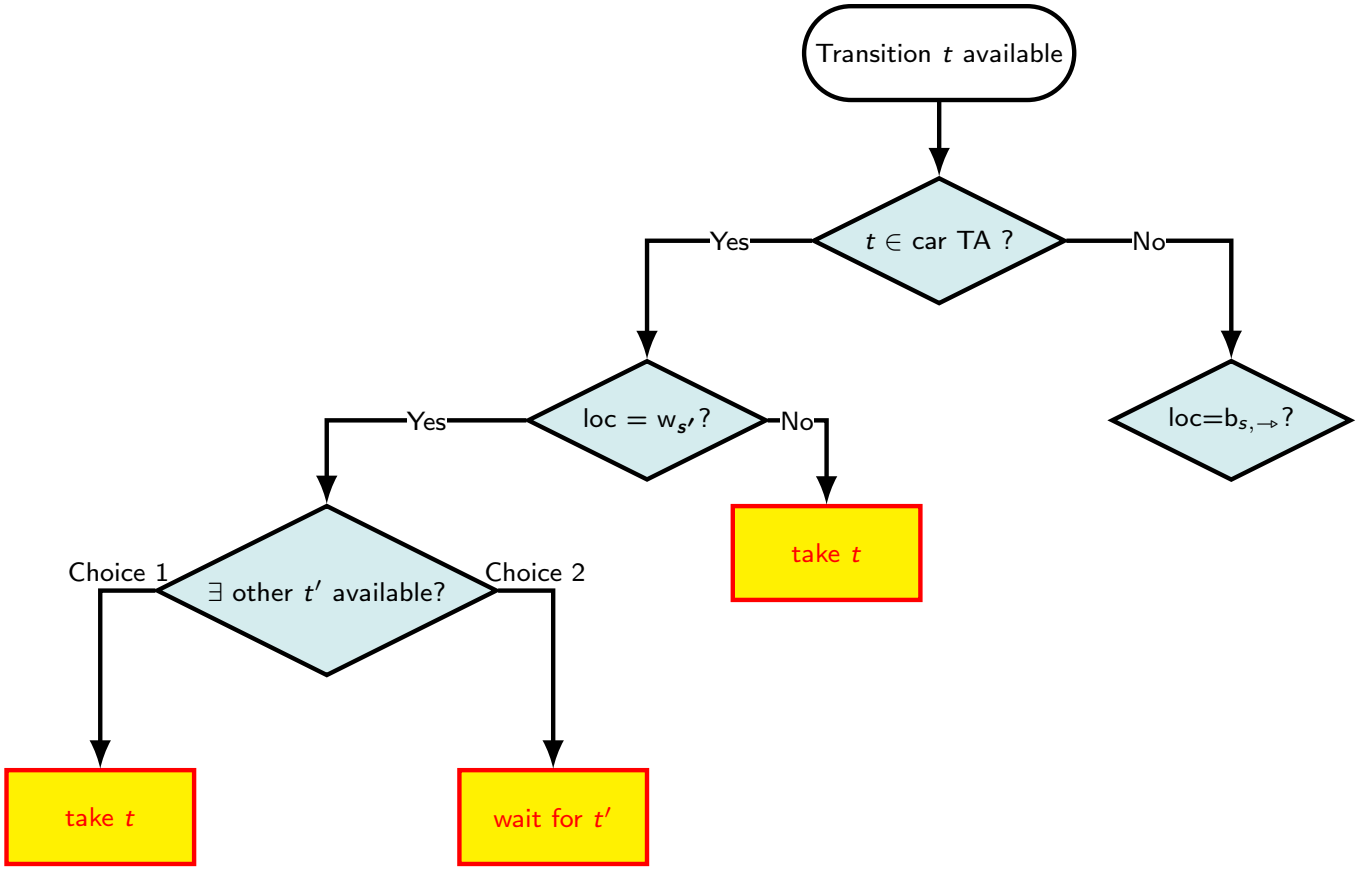
Our Algorithm: a DFS with an optimised succ function



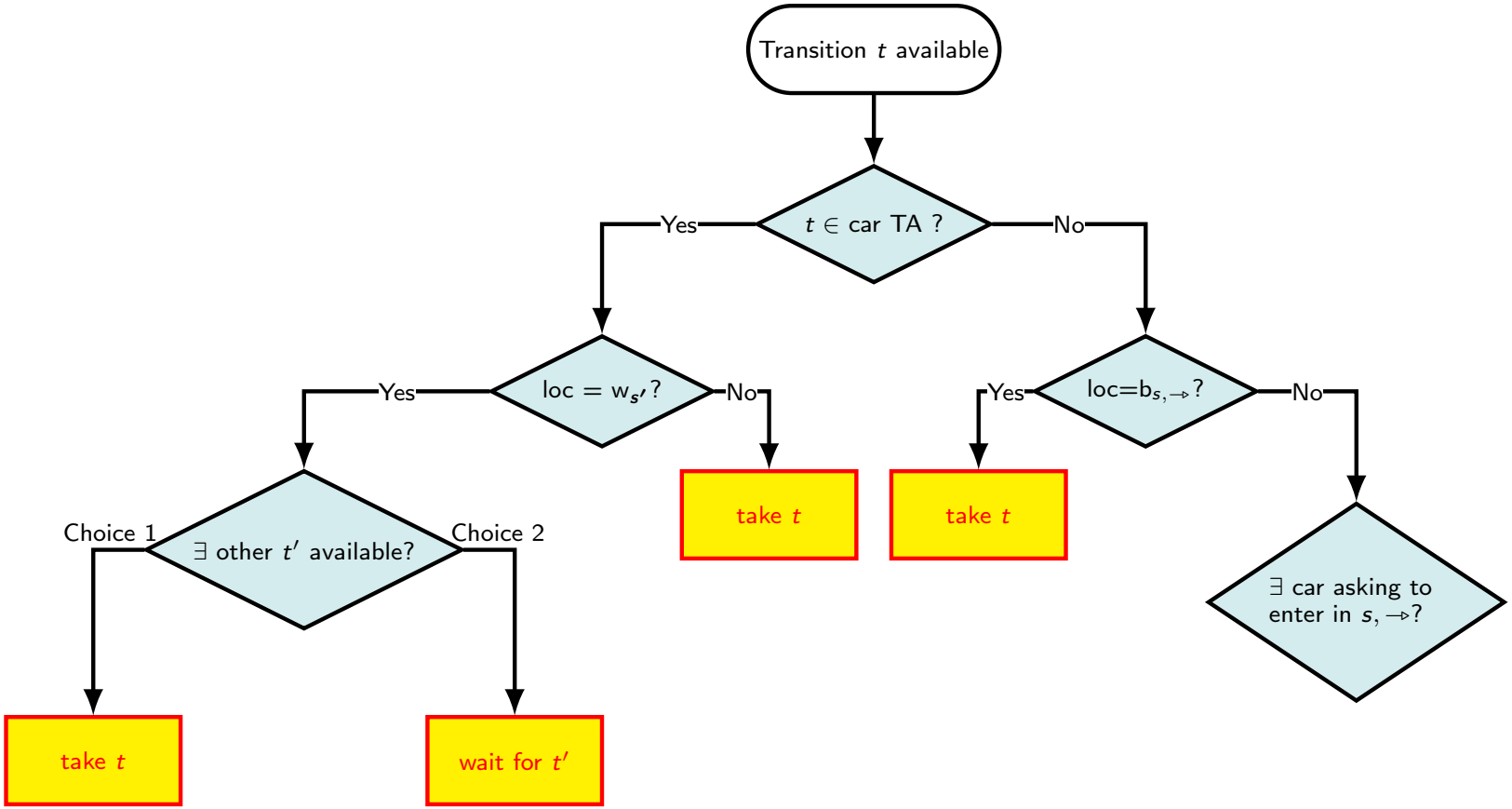
Our Algorithm: a DFS with an optimised succ function



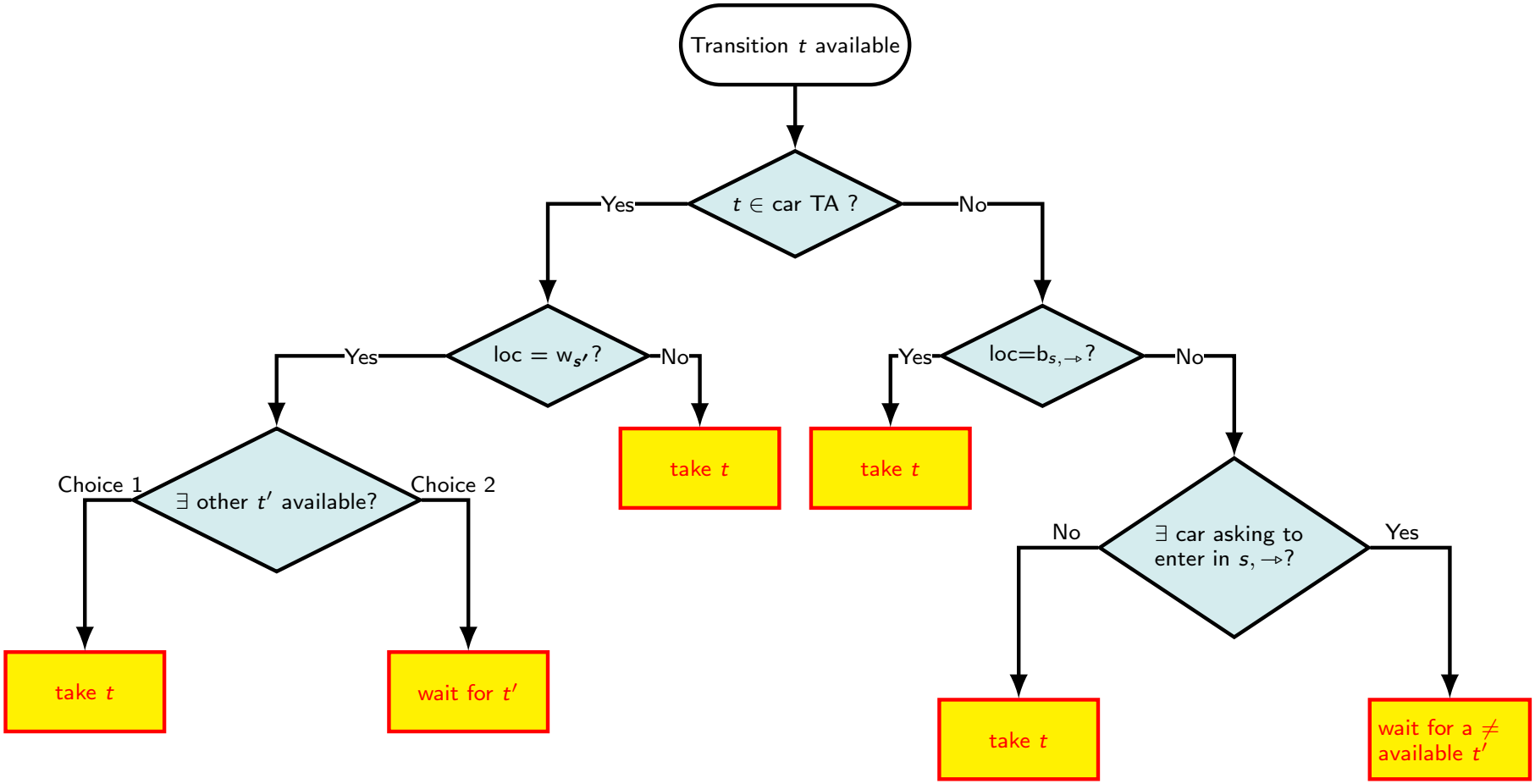
Our Algorithm: a DFS with an optimised succ function



Our Algorithm: a DFS with an optimised succ function

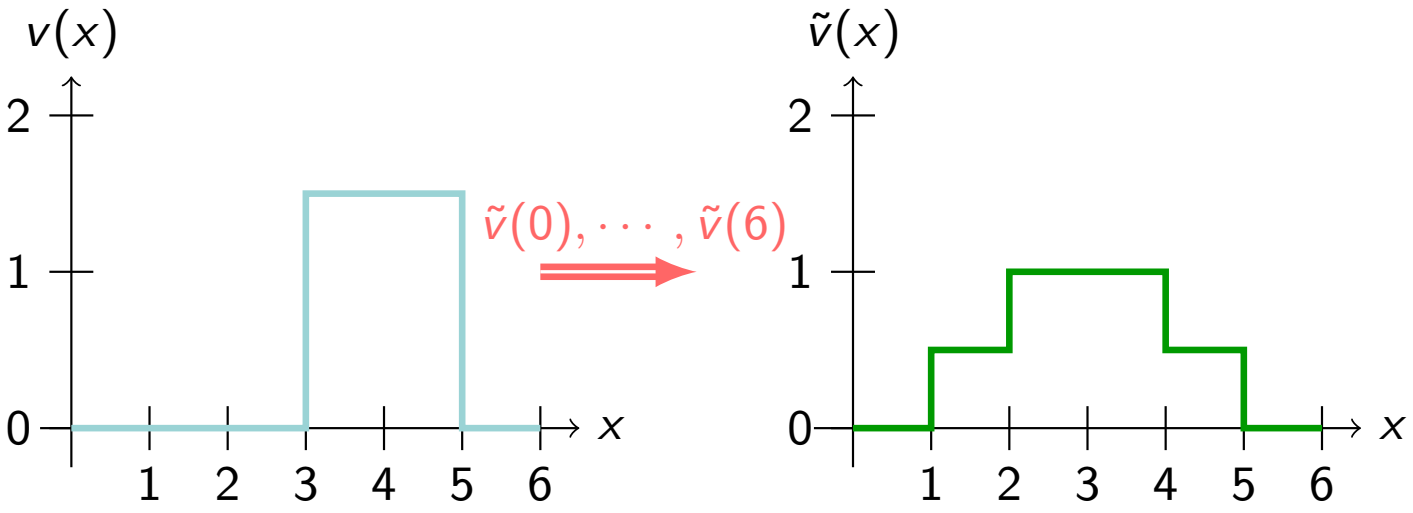


Our Algorithm: a DFS with an optimised succ function



SWA-SMT solver

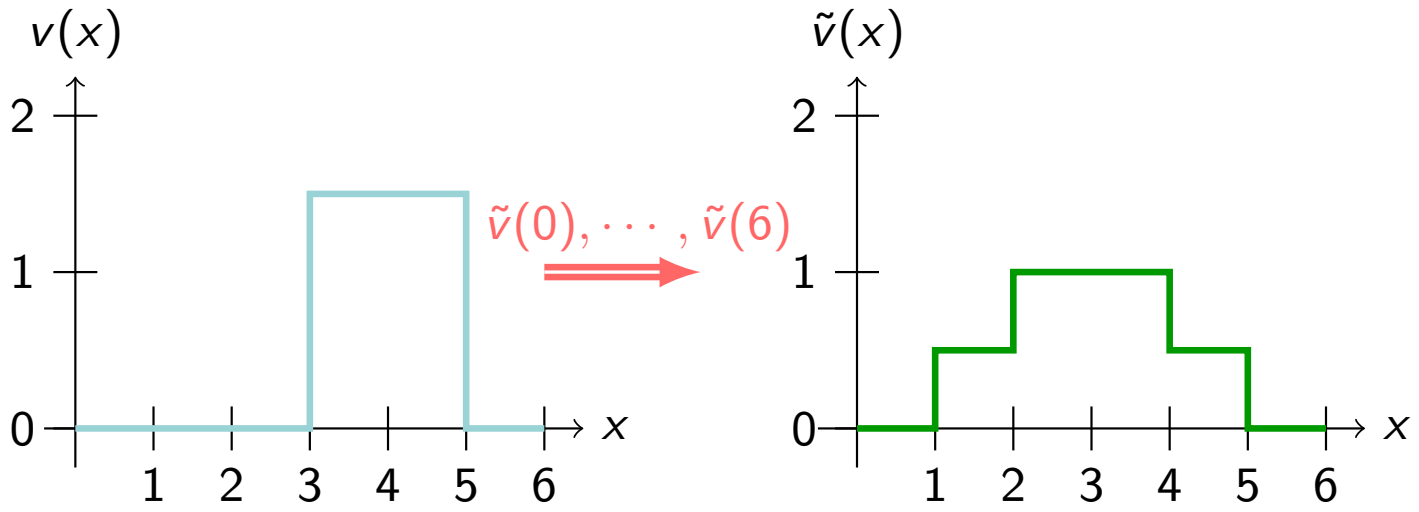
SMT solver



New model for speed graph

- Model speed as a constant piecewise affine function
  - A more realistic model that takes into account the **dynamic of the system**
  - **Different** car speed
  - **Bounds** on deceleration and acceleration

$$v_i \Rightarrow \tilde{v}_i(0), \cdots, \tilde{v}_i(k-1)$$



New positions : $\tilde{x}_i(k) = \sum_{l=0}^{k-1} \tilde{v}_i(l)$

How to still ensure security distance ?

$$\tilde{x}_i(k) = \sum_{l=0}^{k-1} \tilde{v}_i(l)$$

- SMT solver
  - Variables: $(\tilde{v}_i(l))_l$
  - Constraints: linear inequalities w.t.r $\tilde{v}_i$ and some constants (max acceleration, deceleration...)

How to still ensure security distance ?

$$\tilde{x}_i(k) = \sum_{l=0}^{k-1} \tilde{v}_i(l)$$

- SMT solver
  - Variables: $(\tilde{v}_i(l))_l$
  - Constraints: linear inequalities w.t.r $\tilde{v}_i$ and some constants (max acceleration, deceleration...)

- Example of constraints given to the SMT solver for each step k :

(1) $\tilde{v}_i(k) - d_{\max} \leq \tilde{v}_i(k+1) \leq \tilde{v}_i(k) + a_{\max}$
(2) $0 \leq \tilde{v}_i(k) \leq v_{\max}$

RL training

Generate *a dataset for random initial positions*

Dataset

Stage 3*: Train an RL algorithm with our dasaset*

RL

Model

Trajectories

$$
\begin{tikzcd}
s_0,\ Obs_0,\ act_0 \arrow[d, "rwd_0"] \\
s_1,\ Obs_1,\ act_1 \arrow[d, "rwd_1"] \\
s_2,\ Obs_2,\ act_2 \arrow[d, "rwd_2"] \\
\cdots
\end{tikzcd}
$$

- Discounted returns:

$$\mathrm{rwd}_0 + \mathrm{discount} \cdot \mathrm{rwd}_1 + \mathrm{discount}^2 \cdot \mathrm{rwd}_2 + \cdots$$

Why SWA-SMT before RL training ?

- Possible issues when RL training:
  - Local optima (gradient descent)

Why SWA-SMT before RL training ?

- Possible issues when RL training:
  - Local optima (gradient descent)
  - Little changes (action) can have huge consequence of rewards

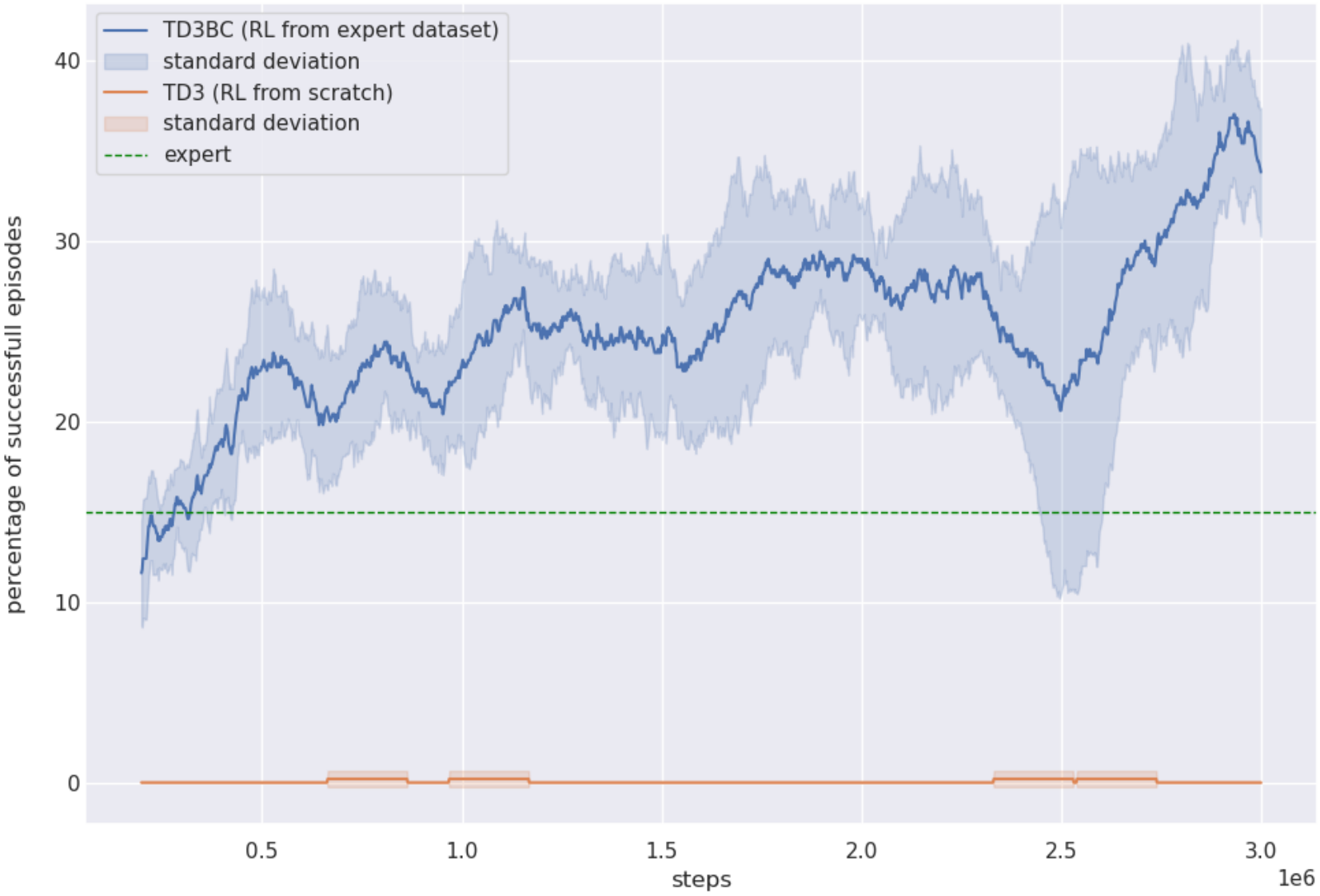
Why SWA-SMT before RL training ?

- Possible issues when RL training:
  - Local optima (gradient descent)
  - Little changes (action) can have huge consequence of rewards
  - Sparce reward signal is a hard exploration problem

Why SWA-SMT before RL training ?

- Possible issues when RL training:
  - Local optima (gradient descent)
  - Little changes (action) can have huge consequence of rewards
  - Sparce reward signal is a hard exploration problem
- Our problem is not easy to solve with single RL training:
  - Combinatorial aspects
  - Discrete and continuous decision together

Results with SWA-SMT solver, post SWA-SMT solver RL and single RL training



Conclusion

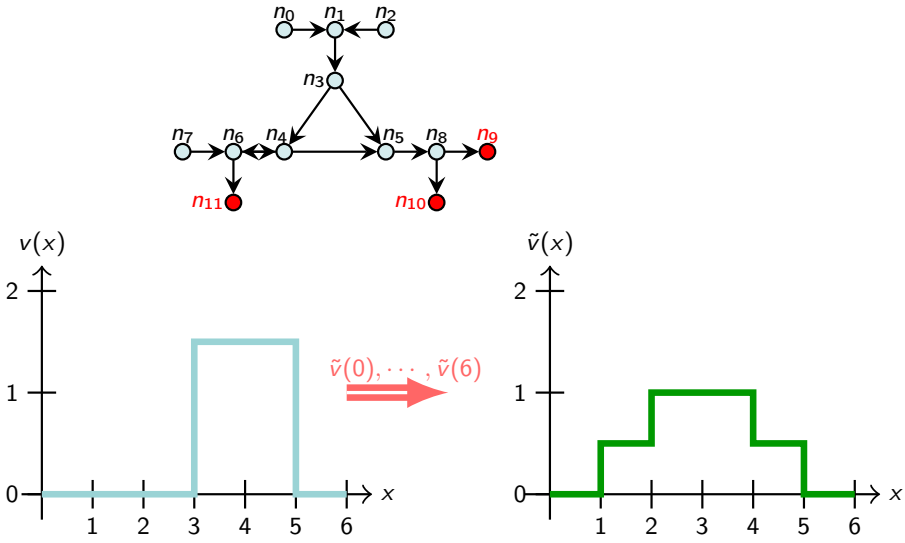
- SWA-SMT Solver

Automata-based model

Efficient algorithm
Abstract model with unrealistic speed model

Piecewise-affine speed graph

Bounded accelleration and deceleration
Different speed
SMT solver to model and solve the distance constraintes



Conclusion

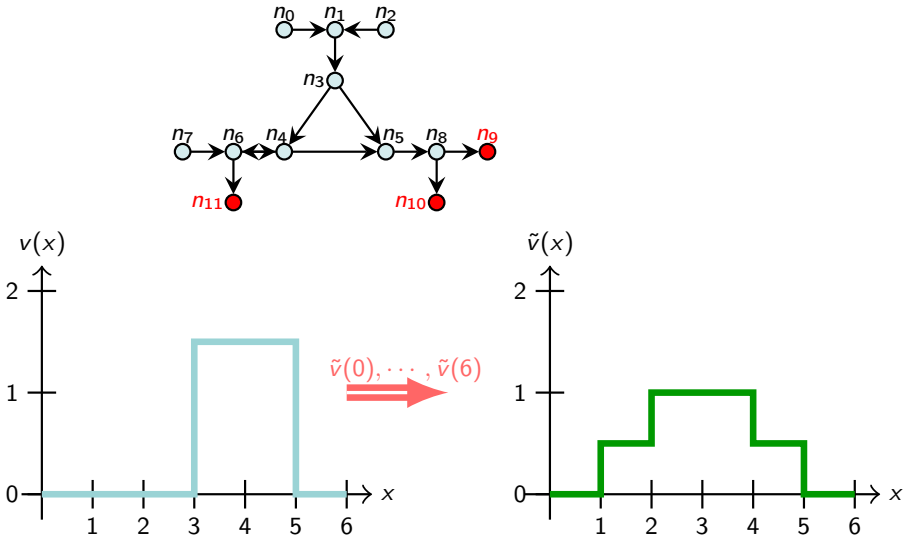
- SWA-SMT Solver

Automata-based model

Efficient algorithm
Abstract model with unrealistic speed model

Piecewise-affine speed graph

Bounded accelleration and deceleration
Different speed
SMT solver to model and solve the distance constraintes



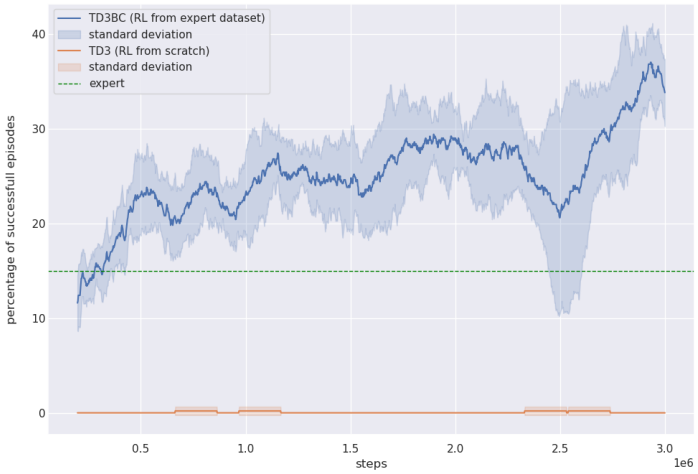
- RL training

Dataset

Trace generated with SWA-SMT solver
Random positions

Performance of RL (helped with SWA-SMT solver)

Better than single RL
Better than SWA-SMT solver
Runtime: $\sim$ 2 days



Conclusion

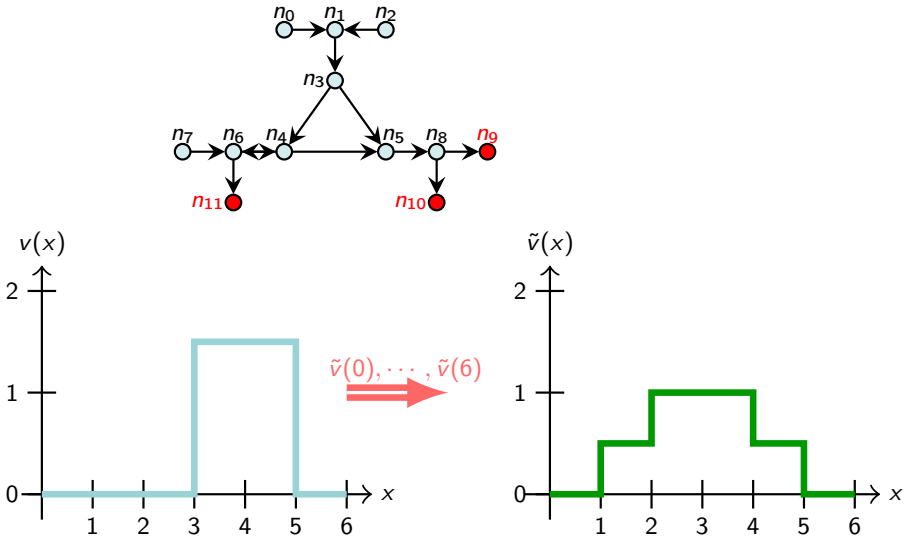
- SWA-SMT Solver

Automata-based model

Efficient algorithm
Abstract model with unrealistic speed model

Piecewise-affine speed graph

Bounded accelleration and deceleration
Different speed
SMT solver to model and solve the distance constraintes



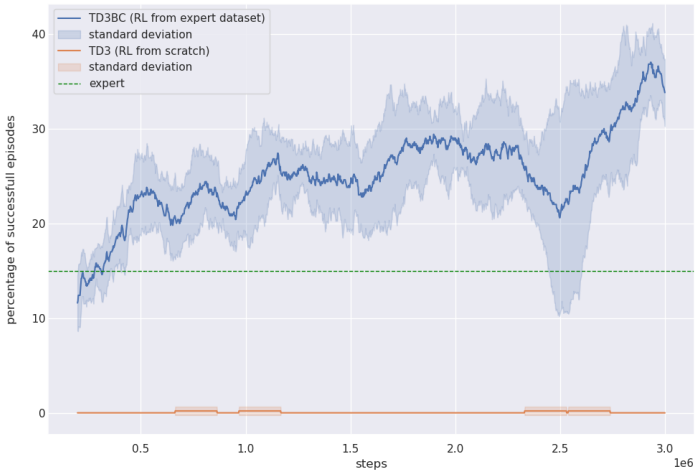
- RL training

Dataset

Trace generated with SWA-SMT solver
Random positions

Performance of RL (helped with SWA-SMT solver)

Better than single RL
Better than SWA-SMT solver
Runtime: $\sim$ 2 days



- Future work: Decentralized multi-agent systems

Appendix

Appendix

Formal translation of these rules

Formal translation of these rules

- **Same directed section**: for all cars $c_i, c_j \in \mathcal{C}$, $c_i \neq c_j$, for all $t \geq 0$, if $\text{sect}_d(c, t) = \text{sect}_d(c', t) = \boldsymbol{s}$ then:

$$|p_{\boldsymbol{s}}(c_i, t) - p_{\boldsymbol{s}}(c_j, t)| \geq \varepsilon$$

- **Neighbouring sections**: for all cars $c_i = (i, [\cdots, (s', \boldsymbol{d}'_i), \cdots]) \in \mathcal{C}$, if there exists a car $c_j = (j, [\cdots, (s, \boldsymbol{d}_j), (s', \boldsymbol{d}'_j), (s'', \boldsymbol{d}''_j), \cdots]) \in \mathcal{C}$ we have two cases:
  If $\boldsymbol{d}'_i = \boldsymbol{d}'_j$: then for all t s.t. $\text{sect}_d(c_i, t) = (s', \boldsymbol{d}'_i)$, $\text{sect}_d(c_j, t) = (s'', \boldsymbol{d}''_j)$ we have

$$L' - p_{(s', \boldsymbol{d}'_i)}(c_i, t) + p_{(s'', \boldsymbol{d}''_j)}(c_j, t) \geq \varepsilon.$$

  If $\boldsymbol{d}'_i \neq \boldsymbol{d}'_j$: then for all t s.t. $\text{sect}_d(c_i, t) = (s', \boldsymbol{d}'_i)$, $\text{sect}_d(c_j, t) = (s, \boldsymbol{d}_j)$ we have

$$L' - p_{(s', \boldsymbol{d}'_i)}(c_i, t) + L - p_{(s, \boldsymbol{d}_i)}(c_j, t) \geq \varepsilon$$

- **Same section, opposite direction**: for all section $s \in \mathcal{S}$, for all $t \geq 0$ and for each pair of cars $c_i, c_j \in \mathcal{C}$:

$$\neg(\text{sect}_d(c_i, t) = (s, \rightarrowtriangle) \wedge \text{sect}_d(c_j, t) = (s, \leftarrowtriangle))$$

- **No overtaking**: we use channels to respect the order of cars.