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 $\cdots$
  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

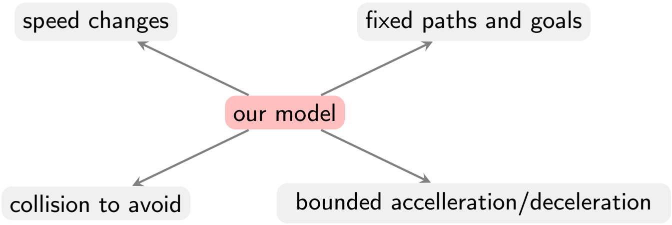
Issues of different methods

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

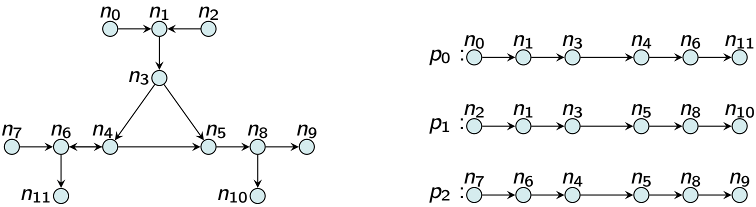
Our solution

- 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.

- Our model

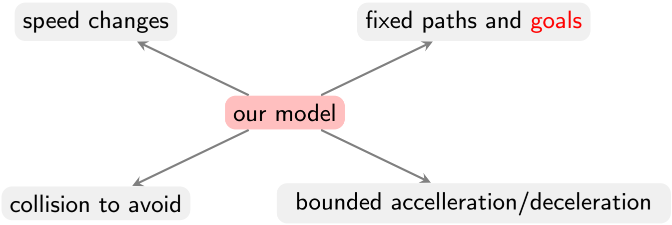


- Example of traffic (left) and associated paths p_0, p_1, p_2 (right)

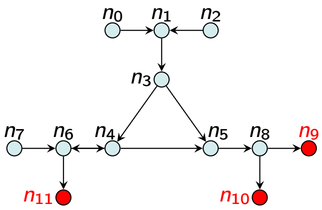


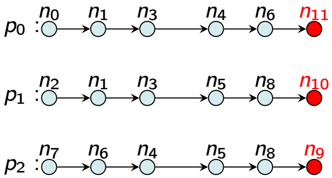
Dynamic multi-agent system

- Our model

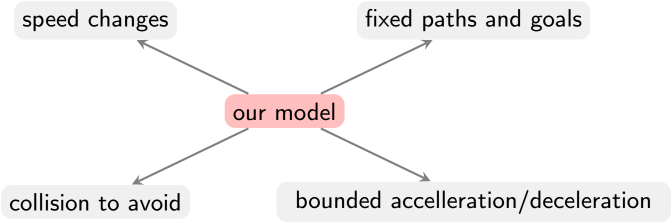


- Example of traffic (left) and associated paths p_0, p_1, p_2 (right)



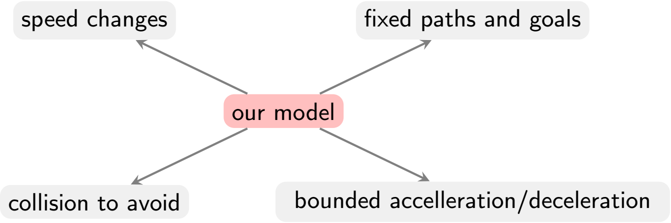


Dynamic multi-agent system

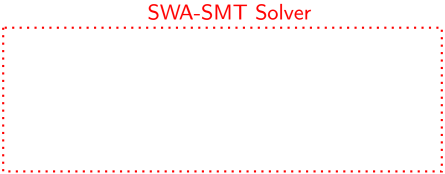


- Our contribution: Controller synthesis
  - Goal: reach goals while avoiding collisions between agents

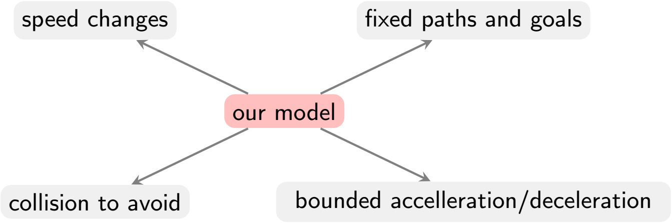
Dynamic multi-agent system



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



Dynamic multi-agent system

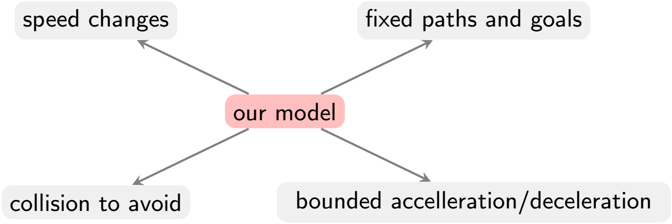


- Our contribution: Controller synthesis
  - Goal: reach goals while avoiding collisions between agents
  - **Three-layer** Method: SWA-SMT solver + RL Training:

SWA-SMT Solver

Stage 1: *Reachability algorithm on system of ISWA*

SWA



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

SWA-SMT Solver

Stage 1: *Reachability algorithm on system of ISWA*

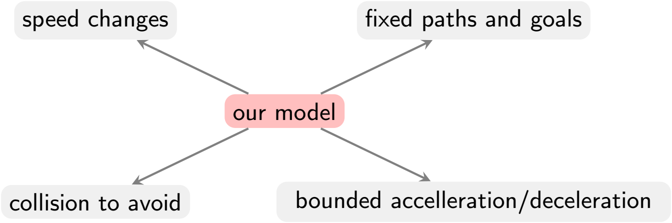
SWA

Stage 2: *Model the accelleration and deceleration*

SMT

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:

SWA-SMT Solver

Stage 1: *Reachability algorithm on system of ISWA*

SWA

Stage 2: *Model the accelleration and deceleration*

SMT

RL training

Generate *a dataset for random initial positions*

Dataset

Dynamic multi-agent system

speed changes

fixed paths and goals

our model

collision to avoid

bounded accelleration/deceleration

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

SWA-SMT Solver

Stage 1: *Reachability algorithm on system of ISWA*

SWA

Stage 2: *Model the accelleration and de-celeration*

SMT

RL training

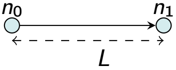
Generate *a dataset for random initial positions*

Dataset

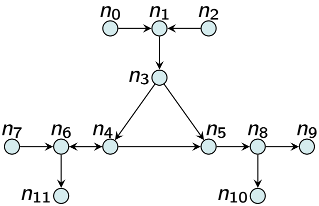
Stage 3: *Train an RL algorithm with our dasaset*

RL

- A point in $\mathbb{R}^2$: a node n_0
- A section $s_{[n_0,n_1],L}$ of the road:



- A path: $p_0 : n_0 \rightarrow n_1 \rightarrow n_3 \rightarrow n_4 \rightarrow n_6 \rightarrow n_{11}$
- A car traffic: c_0, c_1, c_2 are each assigned paths p_0, p_1, p_2:

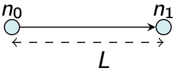


$p_0 : n_0 \rightarrow n_1 \rightarrow n_3 \rightarrow n_4 \rightarrow n_6 \rightarrow n_{11}$

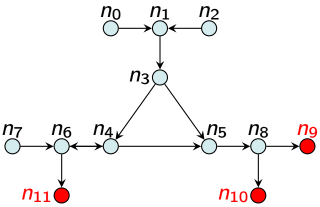
$p_1 : n_2 \rightarrow n_1 \rightarrow n_3 \rightarrow n_5 \rightarrow n_8 \rightarrow n_{10}$

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

- A point in $\mathbb{R}^2$: a node n_0
- A section $s_{[\mathbf{n_0},\mathbf{n_1}],L}$ of the road:



- 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

Main principles

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

Main principles

- Car A progress along its paths
  - x_A: distance travelled along its paths
  - x_A stops (Stopwatches) when the car stops
  - Assumption: cars stops instantly.

Main principles

- Car A progress along its paths
  - x_A: distance travelled along its paths
  - x_A stops (Stopwatches) when the car stops
  - Assumption: cars stops instantly.
  - No overtaking: order of cars respected in a section s with channels: $\mathfrak{c}_{s'}!x_A/\mathfrak{c}_{s'}?x_A$

Main principles

- Car A progress along its paths
  - x_A: distance travelled along its paths
  - x_A stops (Stopwatches) when the car stops
  - Assumption: cars stops instantly.
  - No overtaking: order of cars respected in a section s with channels: $\mathfrak{c}_{s'}!x_A / \mathfrak{c}_{s'}?x_A$
  - Intersection: use classical synchronized action to active *intersection automata*

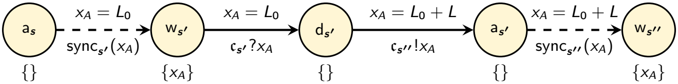
Main principles

- Car A progress along its paths
  - x_A: distance travelled along its paths
  - x_A stops (Stopwatches) when the car stops
  - Assumption: cars stops instantly.
  - No overtaking: order of cars respected in a section s with channels: $\mathfrak{c}_{s'}!x_A/\mathfrak{c}_{s'}?x_A$
  - Intersection: use classical synchronized action to active *intersection automata*
- Intersection automata
  - Each intersection **tracks** the progress of the last car to enter

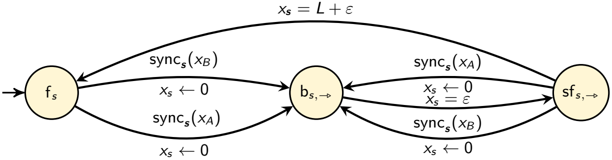
- Car A progress along its paths
  - x_A: distance travelled along its paths
  - x_A stops (Stopwatches) when the car stops
  - Assumption: cars stops instantly.
  - No overtaking: order of cars respected in a section s with channels: $\mathfrak{c}_{s'}!x_A / \mathfrak{c}_{s'}?x_A$
  - Intersection: use classical synchronized action to active *intersection automata*
- Intersection automata
  - Each intersection **tracks** the progress of the last car to enter
  - Goal: ensure that cars maintain a safe distance from each other.
  - Goal: forbid cars to drive in both direction at the same time.

Our model representation

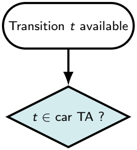
- The car timed automaton of car c_A
  - ▷ represents the progress of the car along its path
  - ▷ is synchronized with intersection automaton



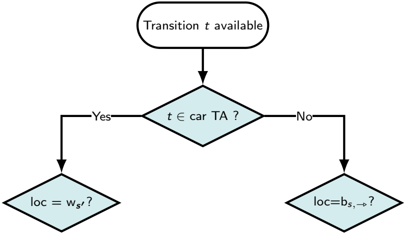
- Timed Automaton of an intersection s
  - ▷ x_s: the progress of the last entered car along s
  - ▷ guards: constraints distance between cars
  - ▷ For A to enter in s: need to active $\text{sync}_s(x_A)$



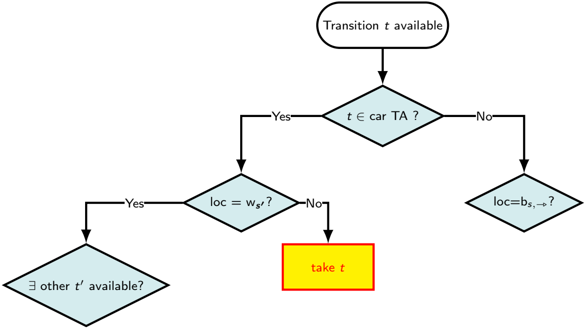
Our Algorithm: a DFS with an optimised succ function



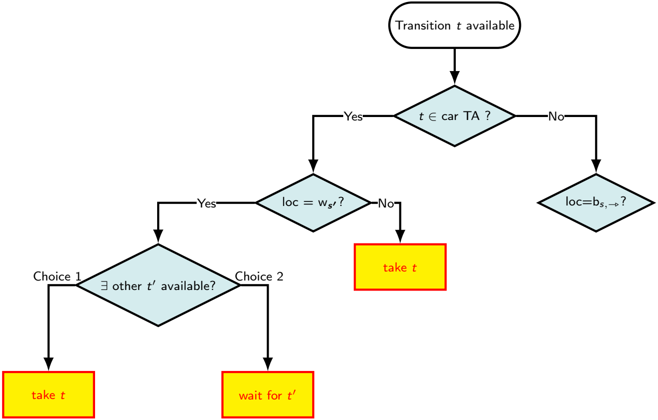
Our Algorithm: a DFS with an optimised succ function



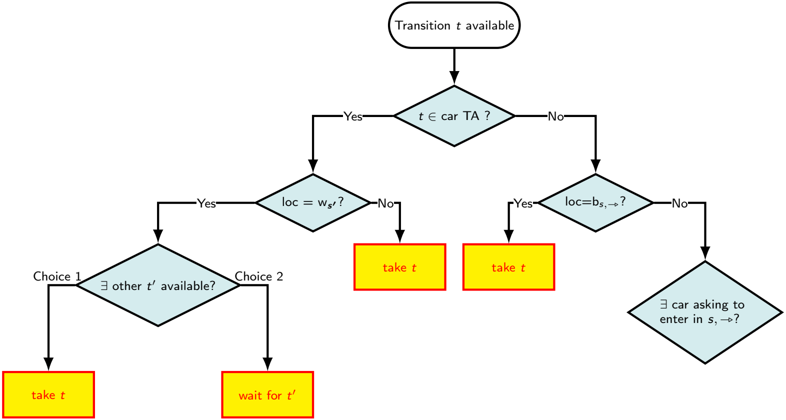
Our Algorithm: a DFS with an optimised succ function



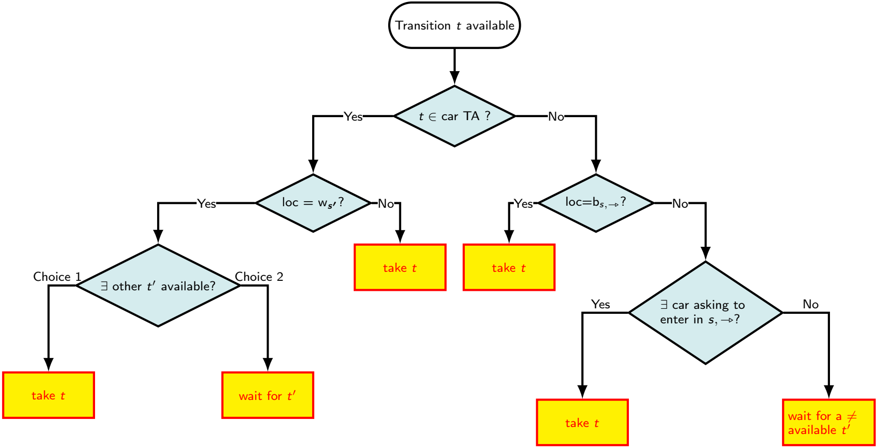
Our Algorithm: a DFS with an optimised succ function



Our Algorithm: a DFS with an optimised succ function



Our Algorithm: a DFS with an optimised succ function



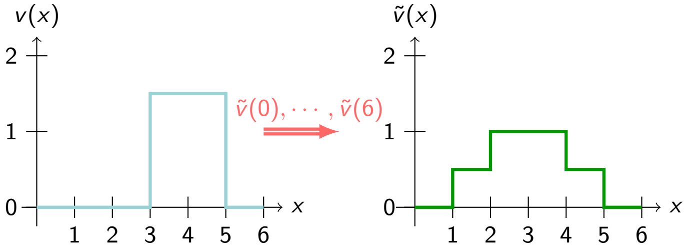
SWA-SMT solver

SMT solver

SMT solver: relax the stopping assumption

- Relax the immediate stop assumption for each car i

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

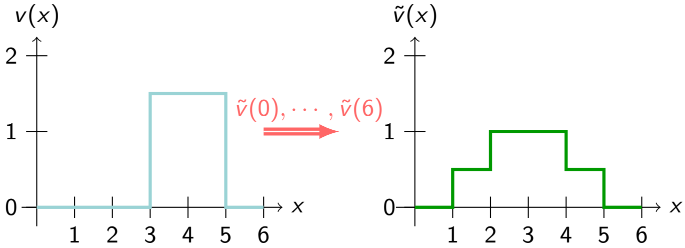


New positions : $\tilde{x}_i(k) = \sum_{l=0}^{k-1} \tilde{v}_i(l)$

SMT solver: relax the stopping assumption

- Relax the immediate stop assumption for each car i

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



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

- 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) $0 \leq \tilde{v}_i(k) \leq v_{\max}$

Appendix

Appendix

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 $\mathrm{sect}_d(c, t) = \mathrm{sect}_d(c', t) = \boldsymbol{s}$ 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. $\mathrm{sect}_d(c_i, t) = (s', \boldsymbol{d}'_i)$, $\mathrm{sect}_d(c_j, t) = (s'', \boldsymbol{d}''_j)$ 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. $\mathrm{sect}_d(c_i, t) = (s', \boldsymbol{d}'_i)$, $\mathrm{sect}_d(c_j, t) = (s, \boldsymbol{d}_j)$ we have
  $$L' - p_{(s', \boldsymbol{d}'_i)}(c_i, t) + L - p_{(s, \boldsymbol{d}_j)}(c_j, t) \geq \varepsilon$$
- **Same section, opposite direction**: for all section $s \in \mathcal{S}$, for all $t \geq 0$ and for each pair of cars $c_i, c_j \in \mathcal{C}$:

$$\neg(\mathrm{sect}_d(c_i, t) = (s, \rightarrow) \wedge \mathrm{sect}_d(c_j, t) = (s, \leftarrow))$$

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