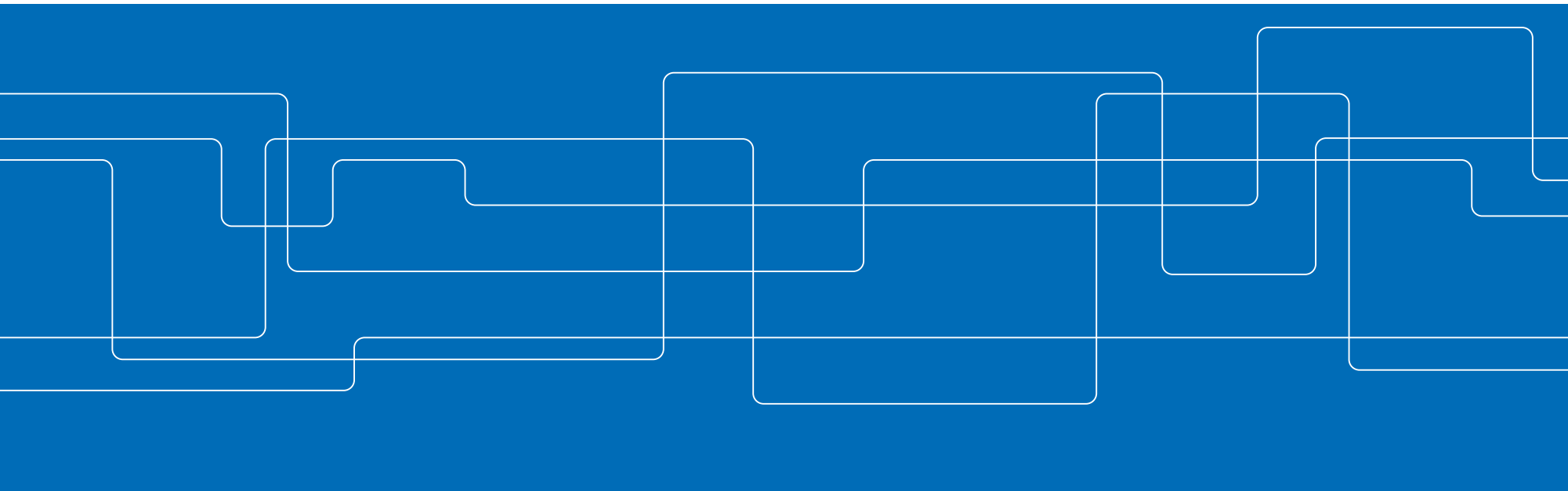
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Multi agent control with LTL specifications and abstraction with input memories

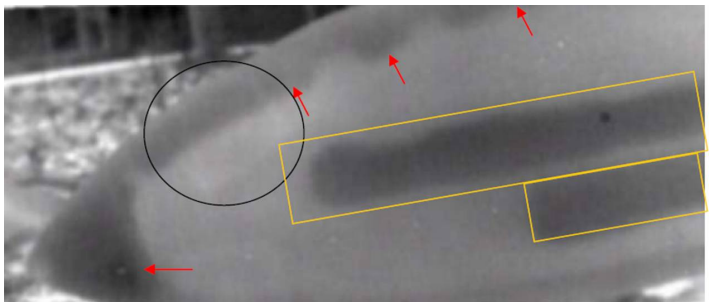
Paul Rousse

Automatic Departement

September 7, 2016



Wind turbine maintenance scenario



Wind turbine maintenance scenario

- Motion and task specifications
- Safety requirements (avoid collision, stay inside a safety area)

Wind turbine maintenance scenario

- Motion and task specifications
- Safety requirements (avoid collision, stay inside a safety area)

$\Rightarrow$ Linear Temporal Logic (LTL)

$\Rightarrow$ formal controller synthesis

Outline

High level specification: the temporal logic

Linear temporal logic (LTL)

$$\varphi ::= \top \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

Temporal operators:

- $\bigcirc$ next
- U until
- $\Box$ always
- $\Diamond$ eventually

High level specification: the temporal logic

Linear temporal logic (LTL)

$$\varphi ::= \top \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

Temporal operators:

- $\bigcirc$ next
- U until
- $\Box$ always
- $\Diamond$ eventually

$$\varphi = \Box\Diamond a$$

$$\begin{aligned} \sigma_1 &= 111a111a\overline{111a} \\ \Longrightarrow \sigma_2 &= 111a111a111\overline{1} \\ \sigma_3 &= 11111111111\overline{1} \end{aligned}$$

Büchi Automaton

Nondeterministic Büchi Automaton

$\mathcal{A}_\varphi = (Q, Q_0, 2^{AP}, \delta, \mathcal{F})$ where:

- Q finite set of states;
- $Q_0 \subset Q$ a set of initial states;
- 2^{AP} the alphabet;
- $\delta : Q \times 2^{AP} \times Q$ a transition relation ;
- $\mathcal{F} \subseteq Q$ set of accepting states.

Büchi Automaton

Nondeterministic Büchi Automaton

$\mathcal{A}_\varphi = (Q, Q_0, 2^{AP}, \delta, \mathcal{F})$ where:

- Q finite set of states;
- $Q_0 \subset Q$ a set of initial states;
- 2^{AP} the alphabet;
- $\delta : Q \times 2^{AP} \times Q$ a transition relation ;
- $\mathcal{F} \subseteq Q$ set of accepting states.

$$\sigma = l_1, l_2, l_3, ... \rightarrow q_0, q_1, q_2, ...$$

Büchi Automaton

Nondeterministic Büchi Automaton

$\mathcal{A}_\varphi = (Q, Q_0, 2^{AP}, \delta, \mathcal{F})$ where:

- Q finite set of states;
- $Q_0 \subset Q$ a set of initial states;
- 2^{AP} the alphabet;
- $\delta : Q \times 2^{AP} \times Q$ a transition relation ;
- $\mathcal{F} \subseteq Q$ set of accepting states.

Acceptance condition: $Inf(\{q_0, q_1, q_2, ...\}) \cap \mathcal{F} \neq \emptyset$

Büchi Automaton

Nondeterministic Büchi Automaton

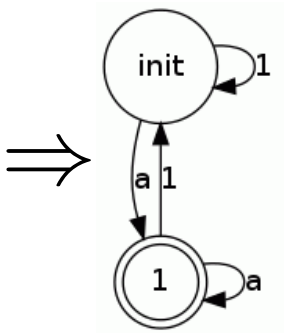
$\mathcal{A}_\varphi = (Q, Q_0, 2^{AP}, \delta, \mathcal{F})$ where:

- Q finite set of states;
- $Q_0 \subset Q$ a set of initial states;
- 2^{AP} the alphabet;
- $\delta : Q \times 2^{AP} \times Q$ a transition relation ;
- $\mathcal{F} \subseteq Q$ set of accepting states.

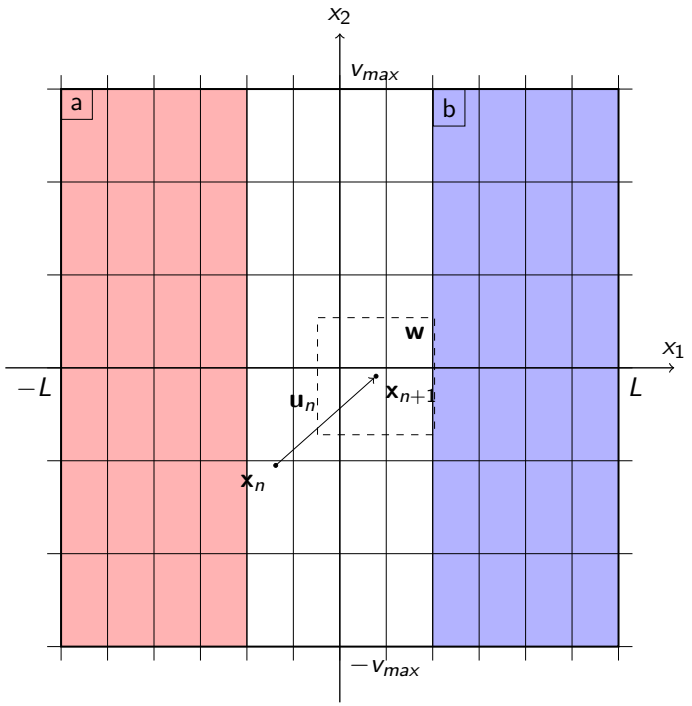
$\sigma_1 = 111a111a\overline{111a}$

$\sigma_2 = 111a111a111\overline{1}$

$\sigma_3 = 111111111111\overline{1}$



Abstraction



$\rightarrow$

$\mathcal{T}_c = (X, X_0, \mathcal{U}, Post_{\mathbf{u}}, H)$

- X states
- $X_0 \subseteq X$ initial states
- U input set
- $Post_{\mathbf{u}} \subseteq X \times U \times X$ transition relation
- $H : X \rightarrow Y$ output map

Figure : Model of the robot

Abstraction

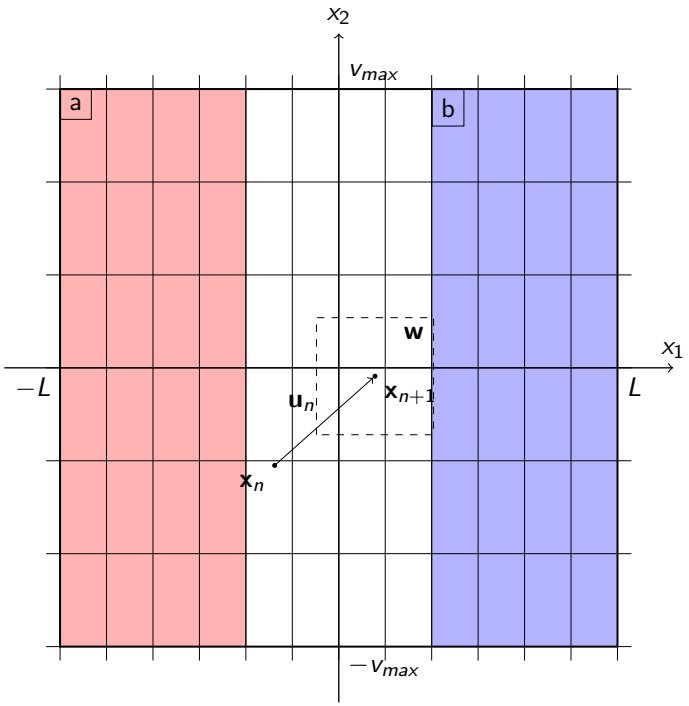


Figure : Model of the robot

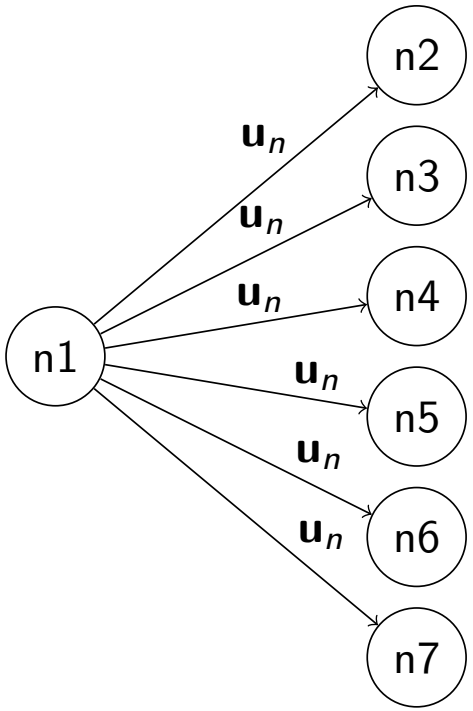


Figure : Finite Transition System

Controller synthesis

Product of a NBA and a FTS

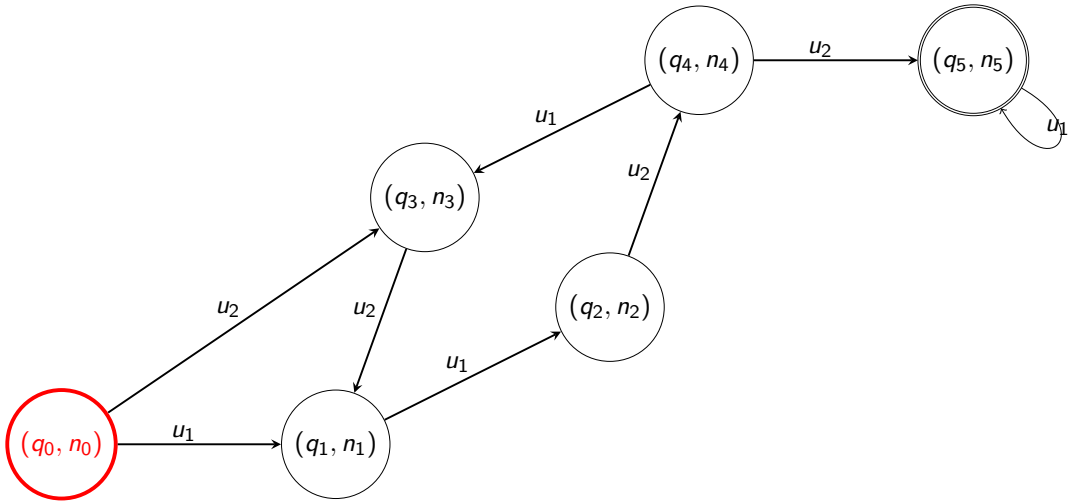
$\mathcal{A}_p = \mathcal{T}_c \otimes \mathcal{A}_\varphi = (Q', Q'_0, \delta', \mathcal{F}')$

- $Q' = X \times Q$ is the set of states,
- $Q'_0 = X_0 \times Q_0$ is the set of initial states,
- $\mathcal{F}' = X \times \mathcal{F}$ the acceptance set,
- $\delta' \subseteq Q' \times U \times Q'$ is the transition relation defined by $\langle x', q' \rangle \in \delta'(\langle x, q \rangle, u)$ iff $x' \in Post_u(x)$, $q' \in \delta(q, L_c(x'))$ and

$$\forall h \in H(Post_u(x)), \exists y \in H^{-1}(h) \cap Post_u(x), \delta(q, L_c(y)) \neq \emptyset$$

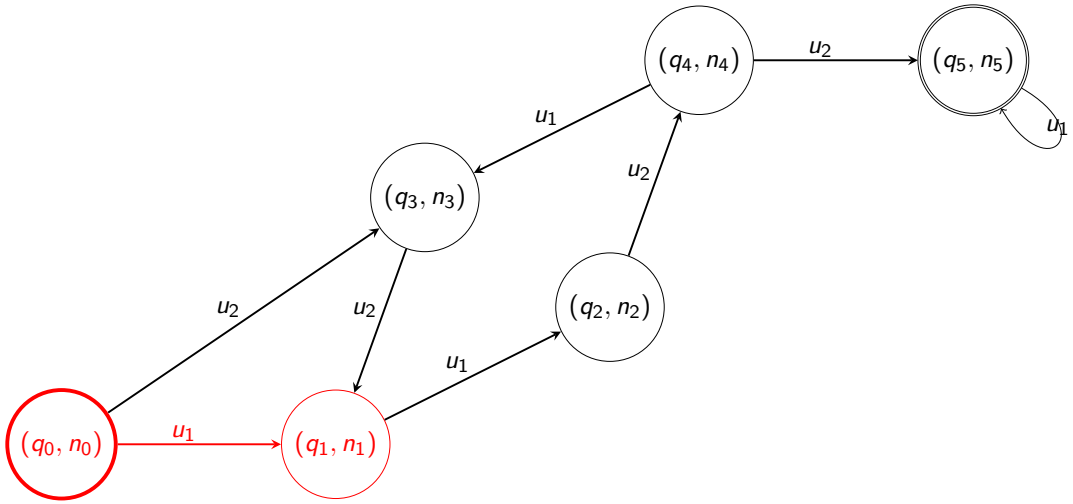
Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



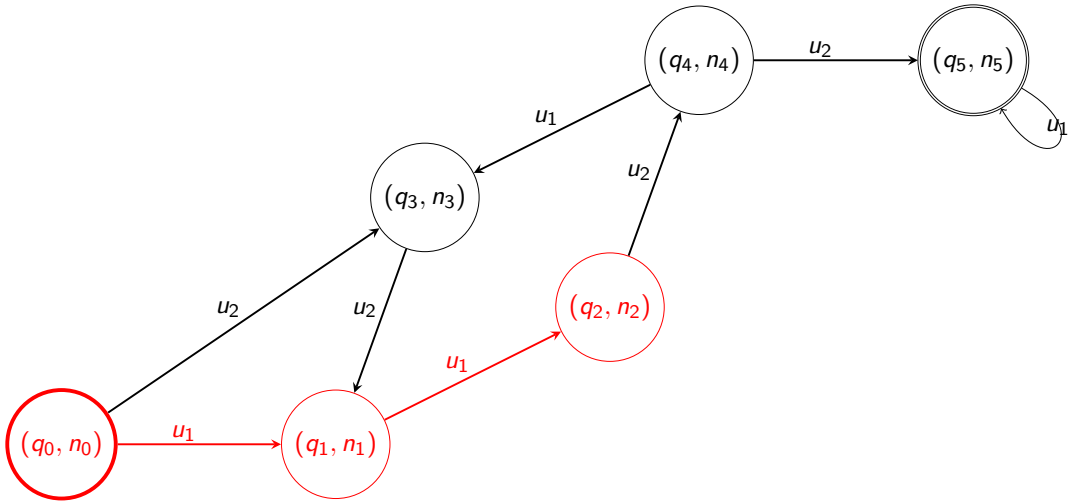
Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



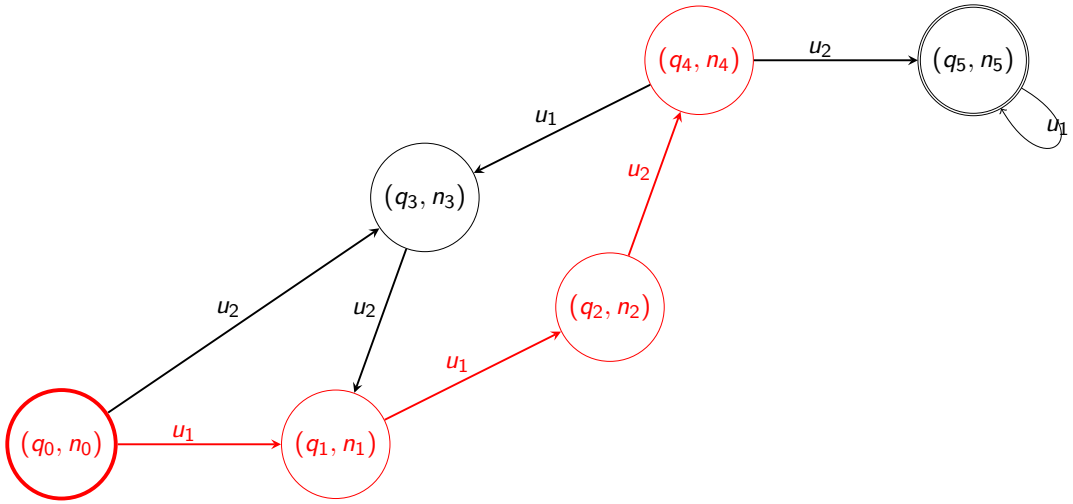
Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



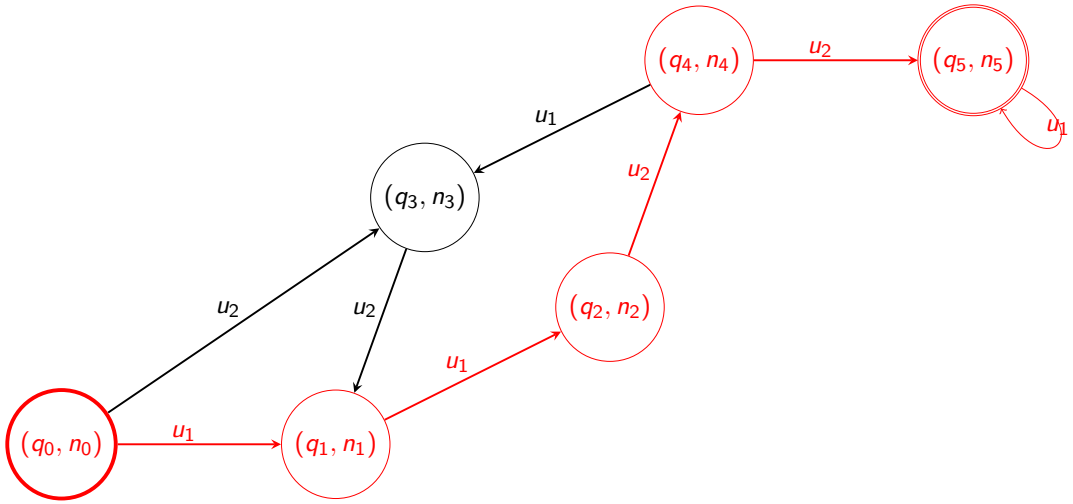
Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



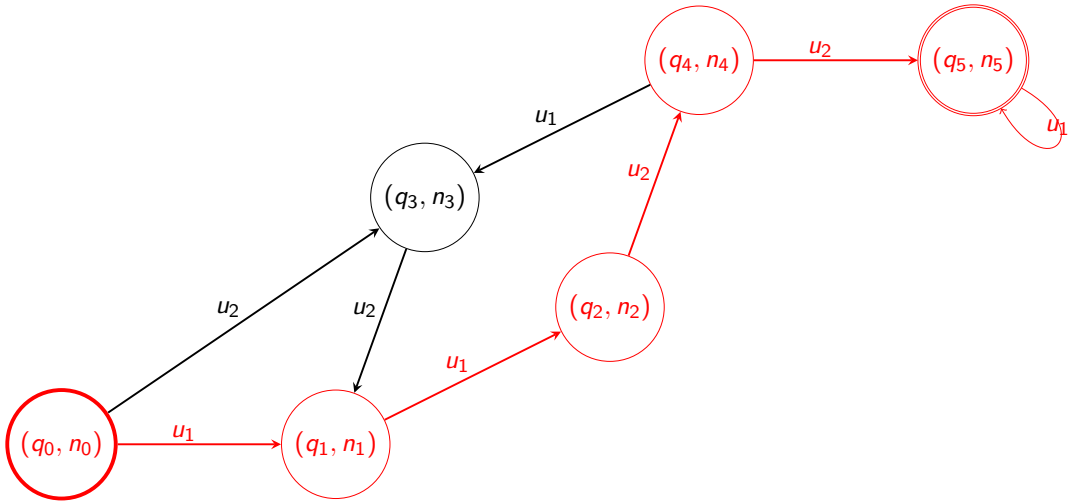
Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



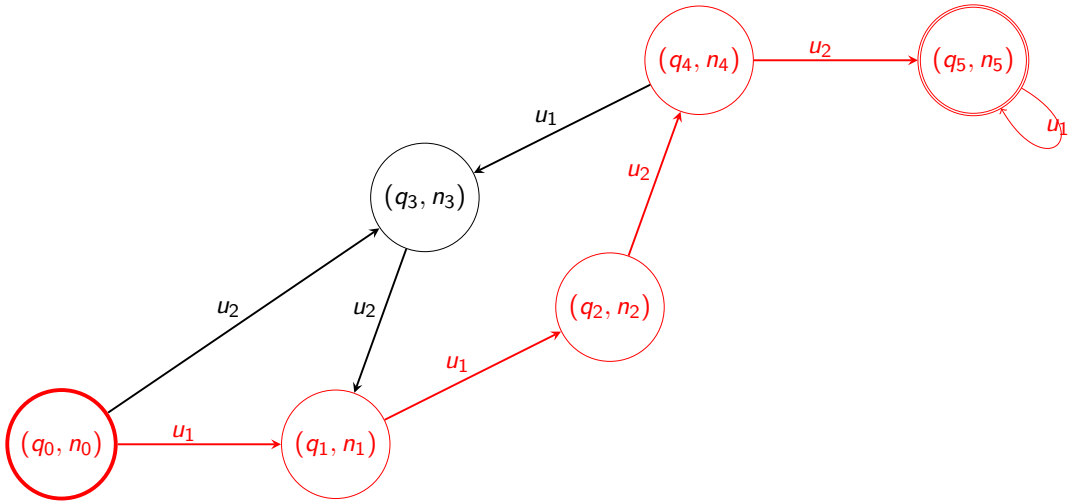
Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



Controller synthesis

Use your favourite graph search algorithm (Depth/Breadth First Search, Fixed points...)



$\Rightarrow$ Controller $=$
$\{((q_1, n_1), u_1), ((q_2, n_2), u_2), \ldots, ((q_n, n_n), u_n)\}$

Recap

Model of the robot

Controller

High level specifications

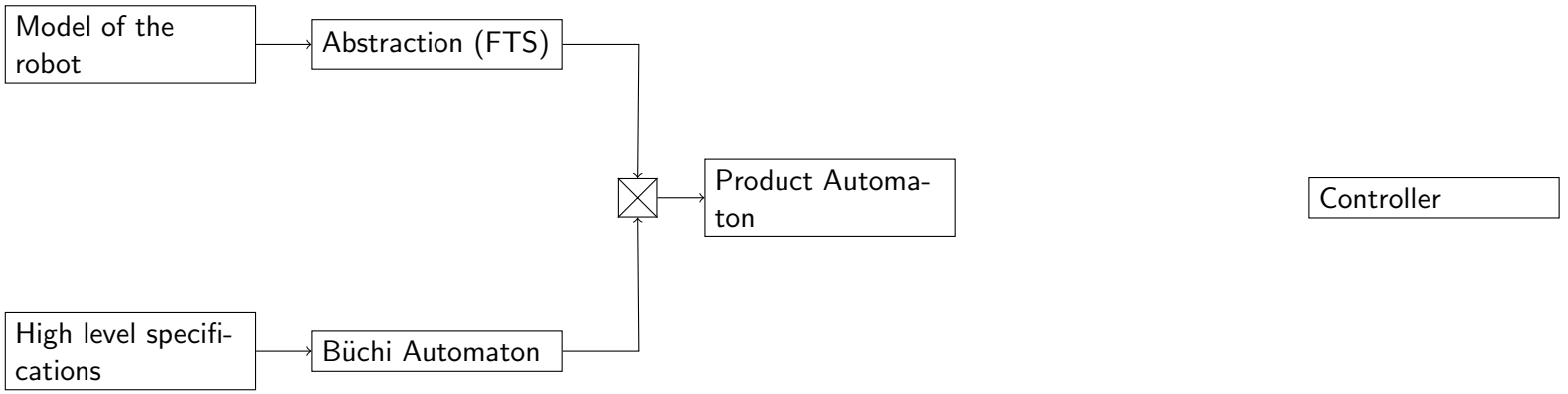
Recap


Model of the robot
Controller
High level specifications
Büchi Automaton

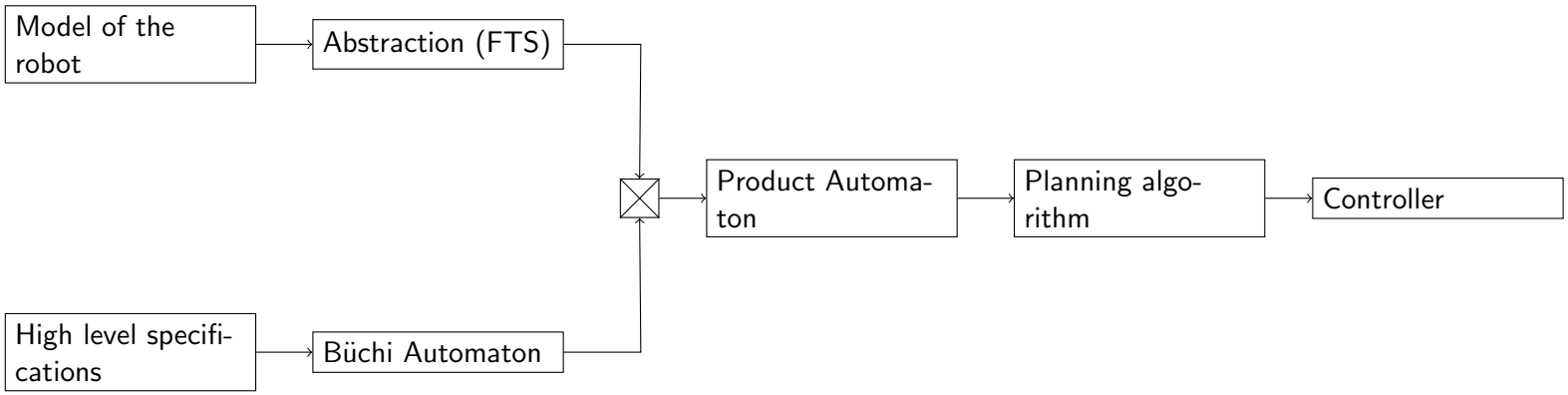
Recap


Model of the robot
Abstraction (FTS)
Controller
High level specifications
Büchi Automaton

Recap


Model of the robot
Abstraction (FTS)
Product Automaton
Controller
High level specifications
Büchi Automaton

Recap


Model of the robot
Abstraction (FTS)
High level specifications
Büchi Automaton
Product Automaton
Planning algorithm
Controller

State extended abstraction

Dynamical system:

- state: $(\mathbf{x}^{obs}, \mathbf{x}^{unobs})$
- transition: $(\mathbf{x}^{obs}, \mathbf{x}^{unobs}) \xrightarrow[S]{\mathbf{u}} (\mathbf{x}_+^{obs}, \mathbf{x}_+^{unobs})$

State extended abstraction

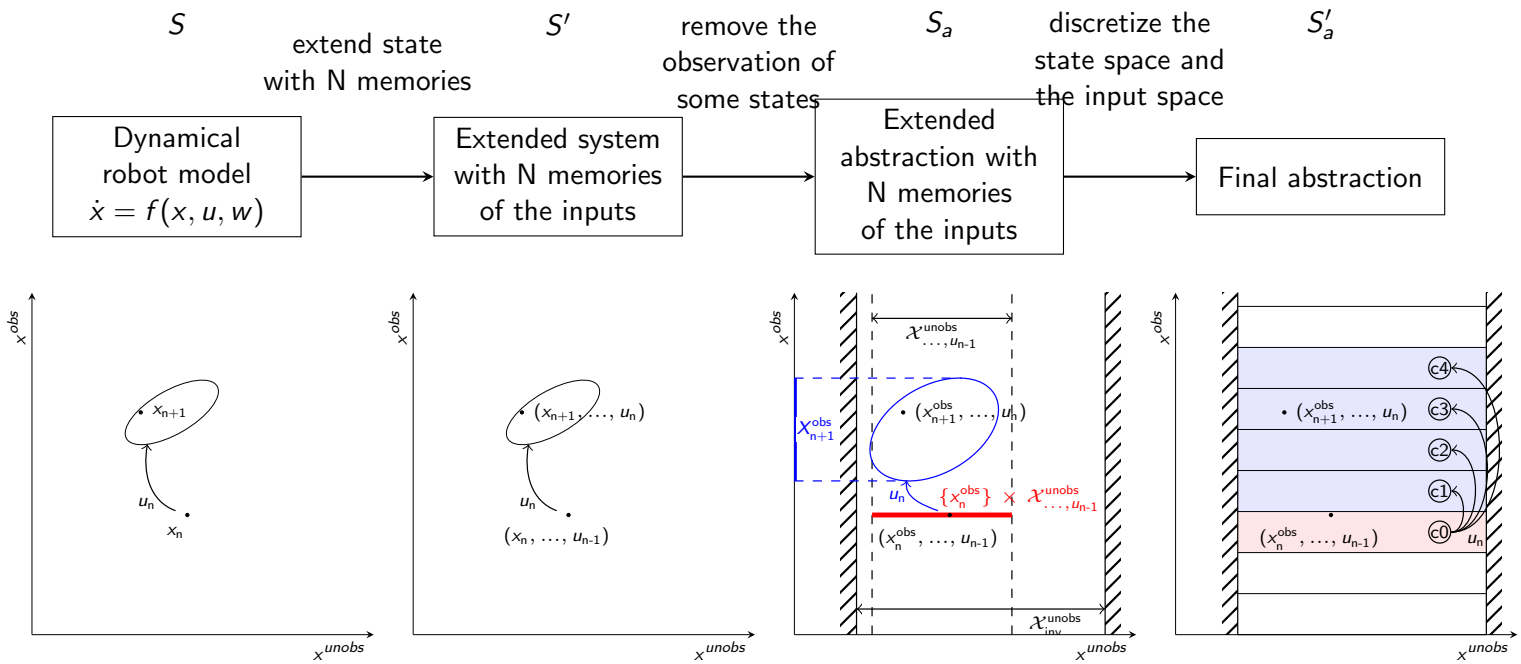
Dynamical system:

- state: $(\mathbf{x}^{obs}, \mathbf{x}^{unobs})$
- transition: $(\mathbf{x}^{obs}, \mathbf{x}^{unobs}) \xrightarrow[S]{\mathbf{u}} (\mathbf{x}_+^{obs}, \mathbf{x}_+^{unobs})$

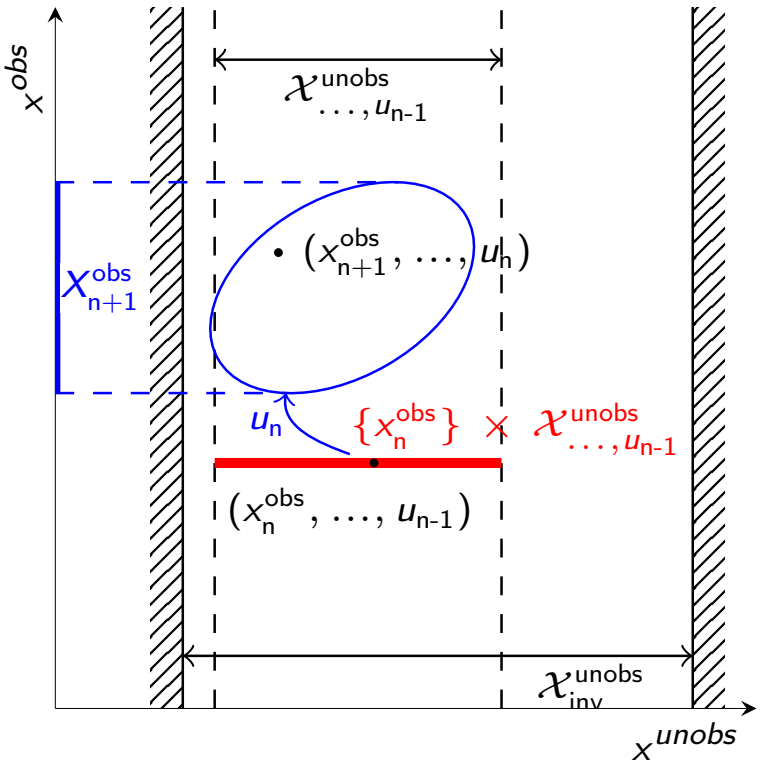
Abstraction:

- $(\mathbf{x}^{obs}, \mathbf{u}_{n-N}, \ldots, \mathbf{u}_{n-1})$
- transition:
  $(\mathbf{x}^{obs}, \mathbf{u}_{n-N}, \ldots, \mathbf{u}_{n-1}) \xrightarrow[S_a]{\mathbf{u}} (\mathbf{x}_+^{obs}, \mathbf{u}_{n+1-N}, \ldots, \mathbf{u}_{n-1}, \mathbf{u})$

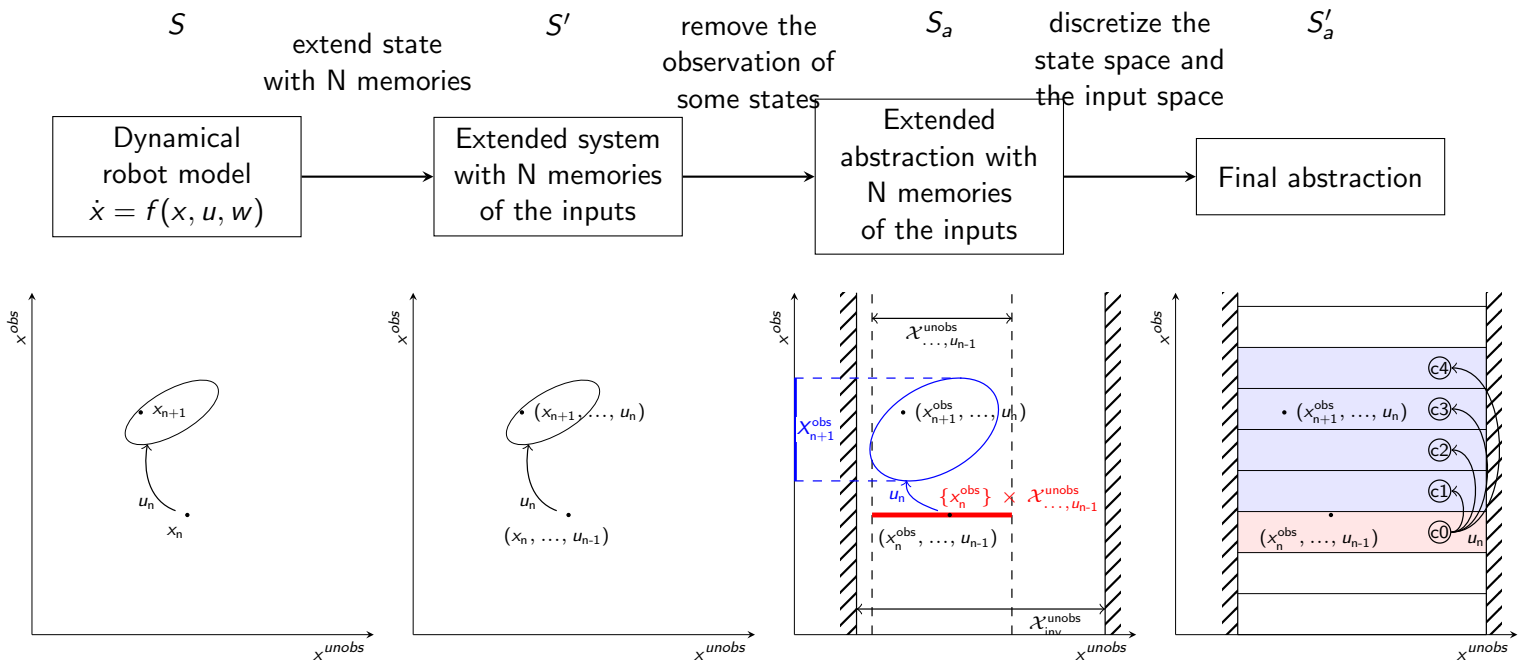
State extended abstraction



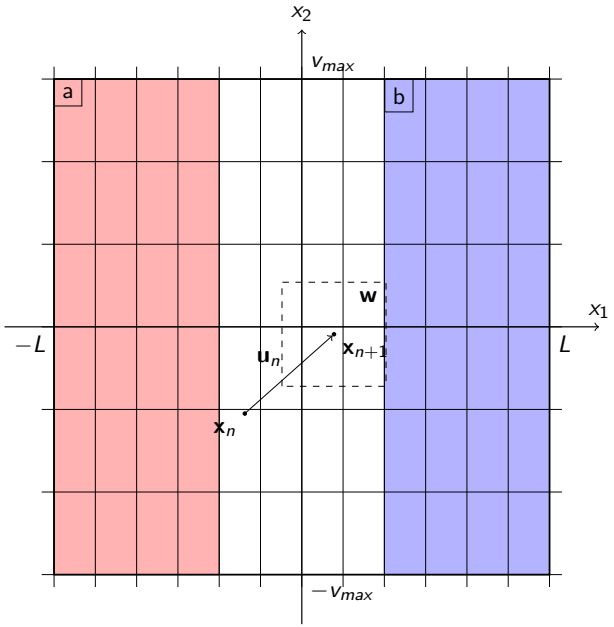
State extended abstraction

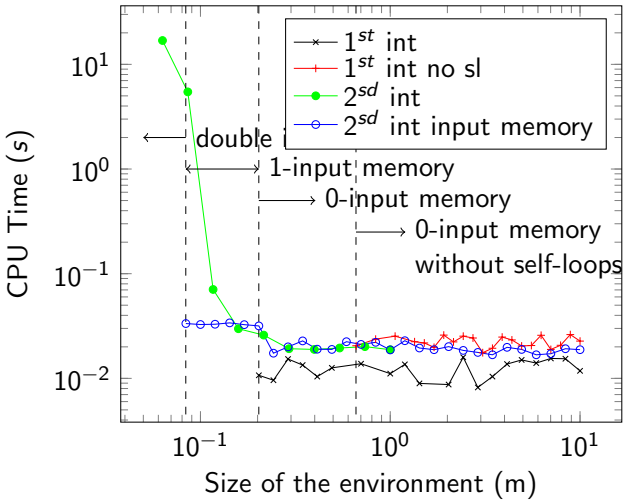


State extended abstraction



Comparison with uniform discretization



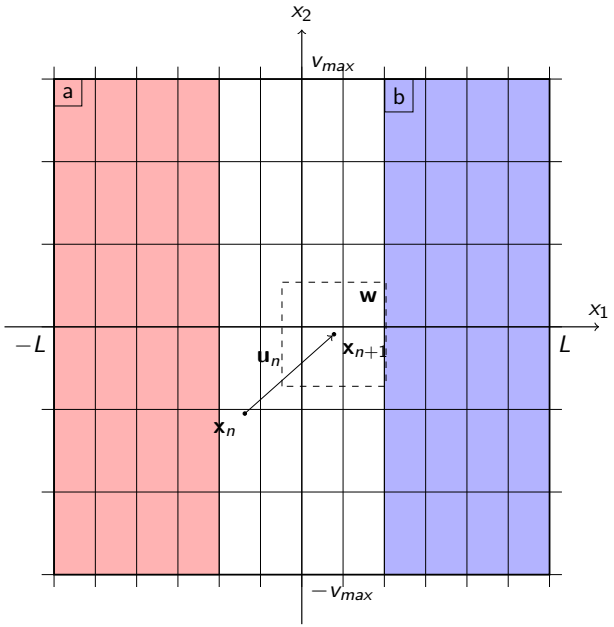


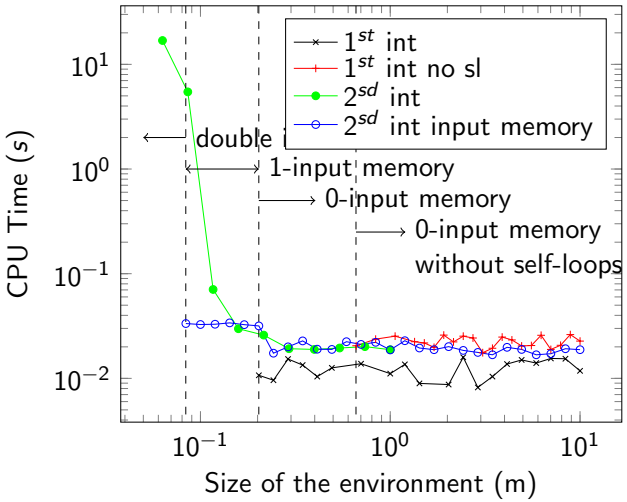
$$\dot{\mathbf{v}} = k(\mathbf{v}_{ref} - \mathbf{v}) + \mathbf{w}_v$$

$$\dot{\mathbf{x}} = \mathbf{v} + \mathbf{w}_x$$

$$\varphi = (\square\diamond a) \wedge (\square\diamond b)$$

Comparison with uniform discretization





- more conservative (the information is selected more wisely)

Admissible noise

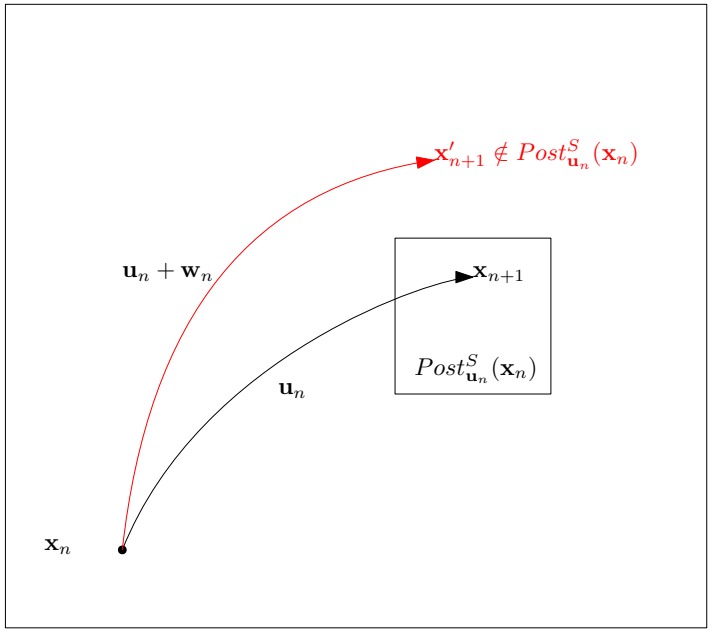


Figure : Dynamical system

Admissible noise

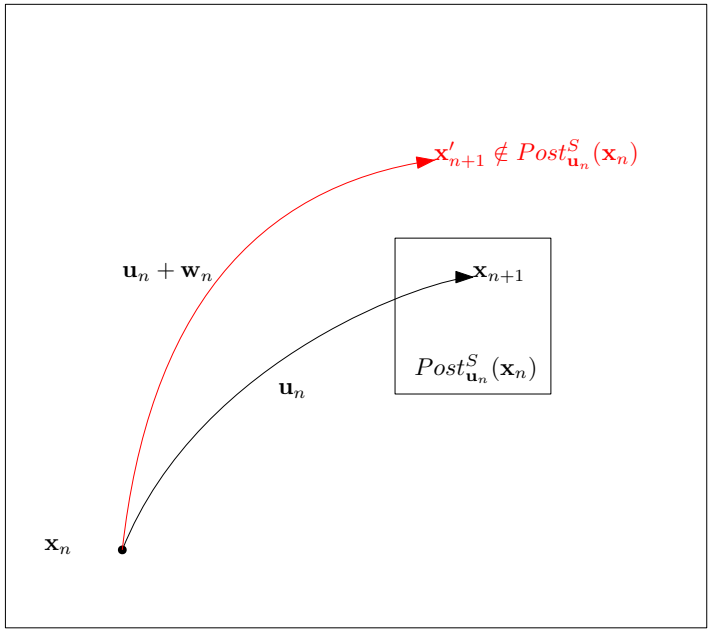


Figure : Dynamical system

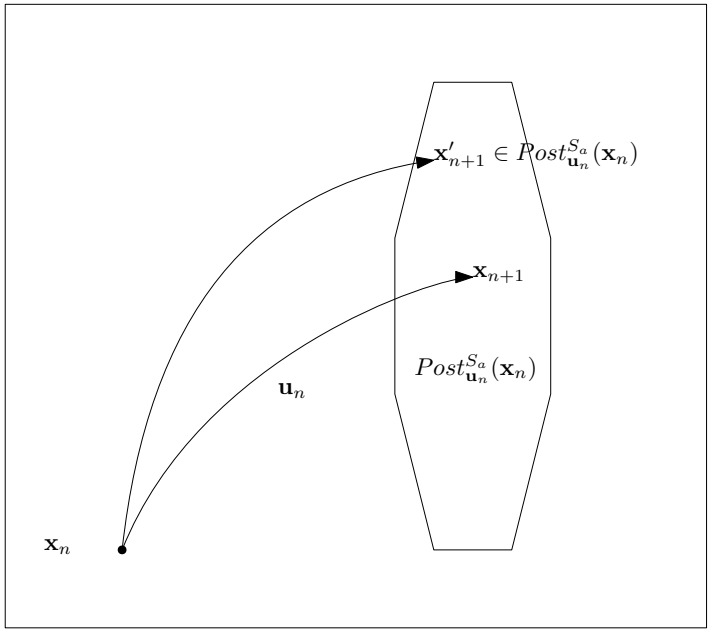
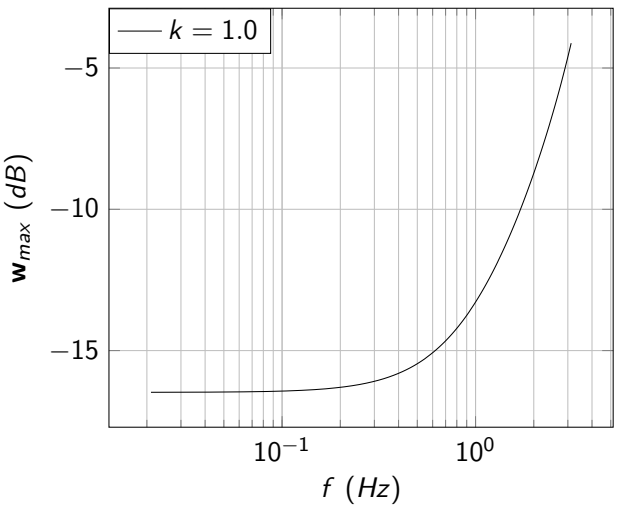
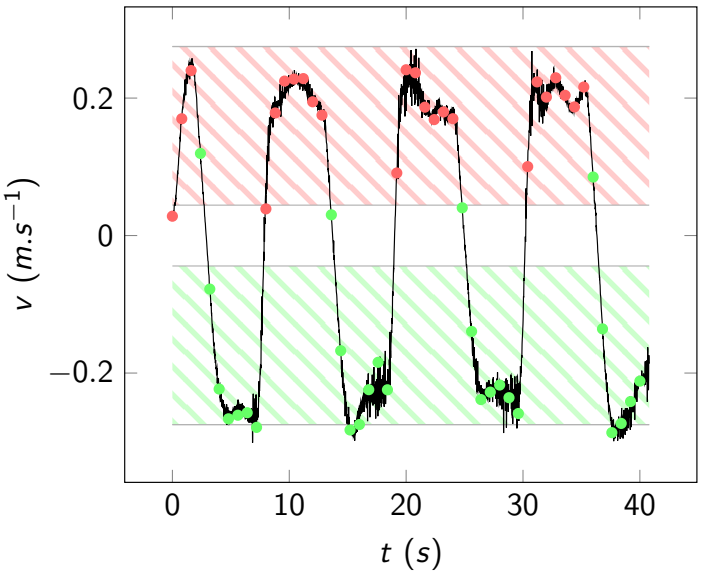


Figure : Abstraction

Admissible noise

$$\mathcal{W}_a = \{\{\mathbf{w}_n\}_{n\in\mathbb{N}} \mid \mathcal{L}_1(G^{unobs} * G_{\mathbf{w}}) \leq \sigma\}$$
$$\mathcal{W} \subset \mathcal{W}_a$$





Path planning algorithm

Planner specifications:

Path planning algorithm

Planner specifications:

- Non deterministic FTS

Path planning algorithm

Planner specifications:

- Non deterministic FTS
- Self-loops and cycles

Path planning algorithm

Planner specifications:

- Non deterministic FTS
- Self-loops and cycles
- go to the goal set in finite time

Path planning algorithm

Planner specifications:

- Non deterministic FTS
- Self-loops and cycles
- go to the goal set in finite time

$\Rightarrow$ Strong cyclic planner with local fairness property

Path planning algorithm

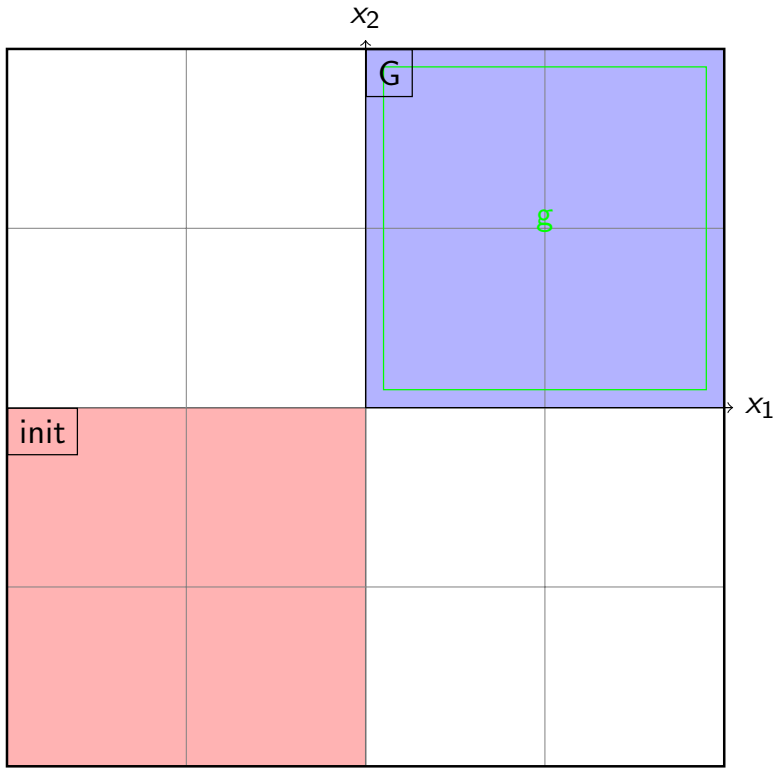
Planner specifications:

- Non deterministic FTS
- Self-loops and cycles
- go to the goal set in finite time

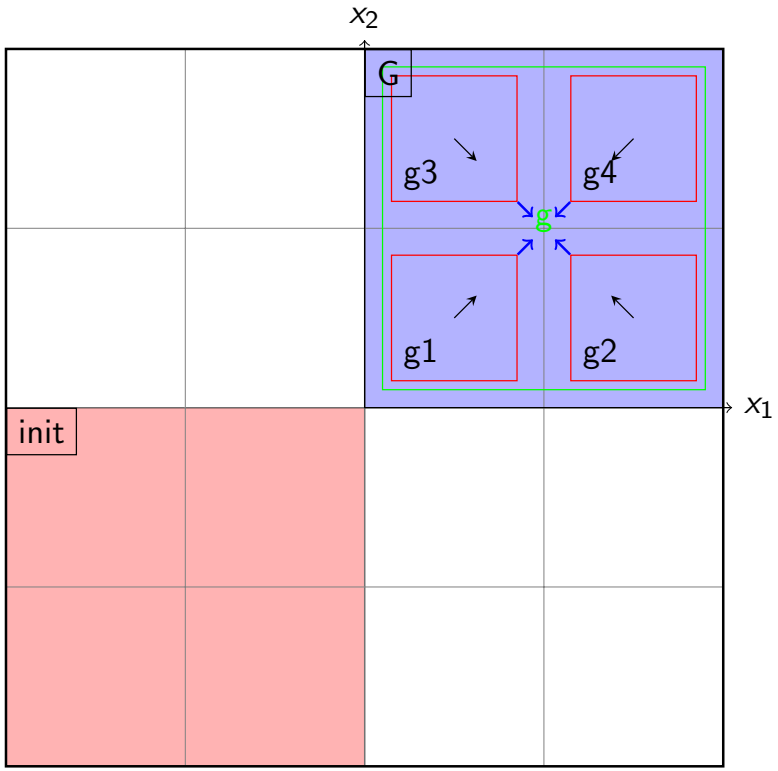
$\Rightarrow$ Strong cyclic planner with local fairness property

Escape time property in the cycle $\mathcal{C}$:
$\forall \{\mathbf{x}_k\}_k \in \mathcal{X}^*, \mathbf{x}_0 \in \mathcal{C} \Rightarrow \exists k \in \mathbb{N}, \mathbf{x}_k \notin \mathcal{C}$

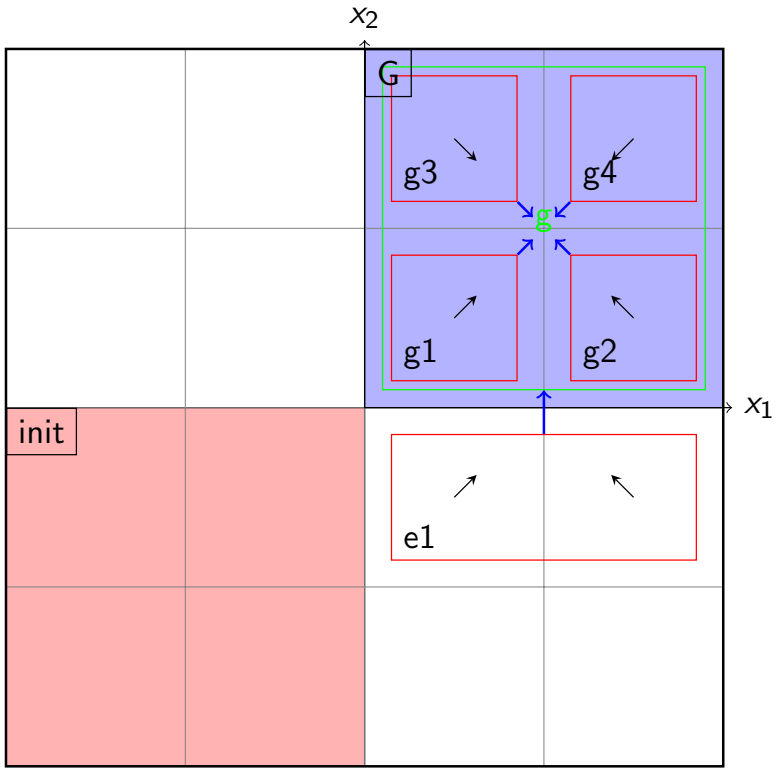
Backward Reachability Algorithm - $\varphi = \Diamond\Box G$



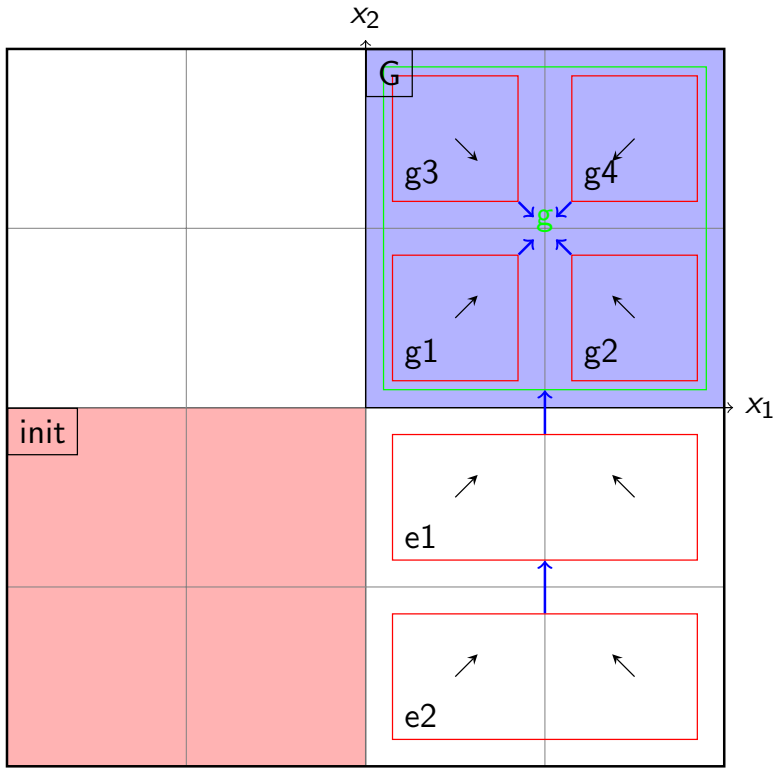
Backward Reachability Algorithm - $\varphi = \Diamond\Box G$



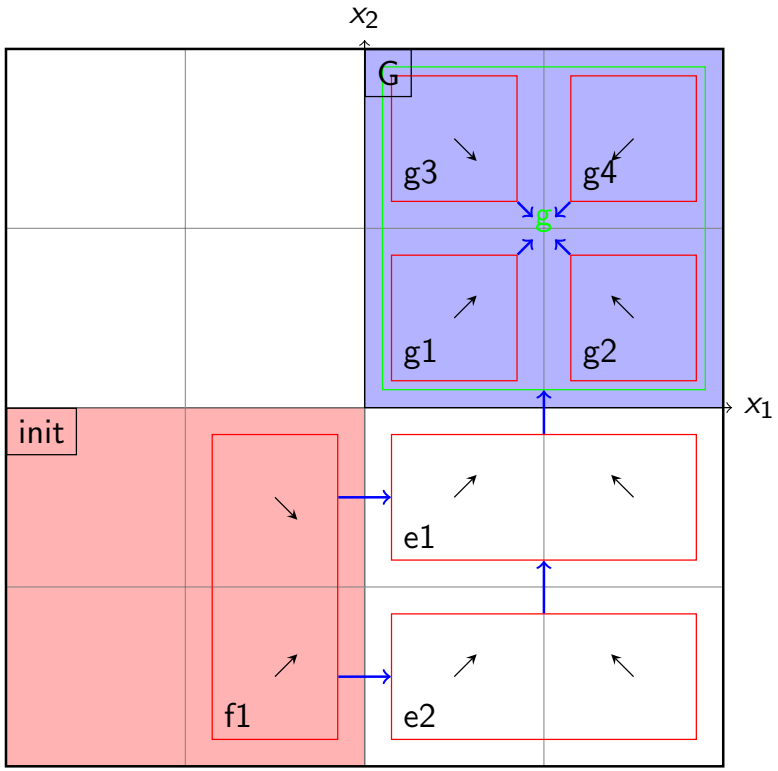
Backward Reachability Algorithm - $\varphi = \Diamond\Box G$



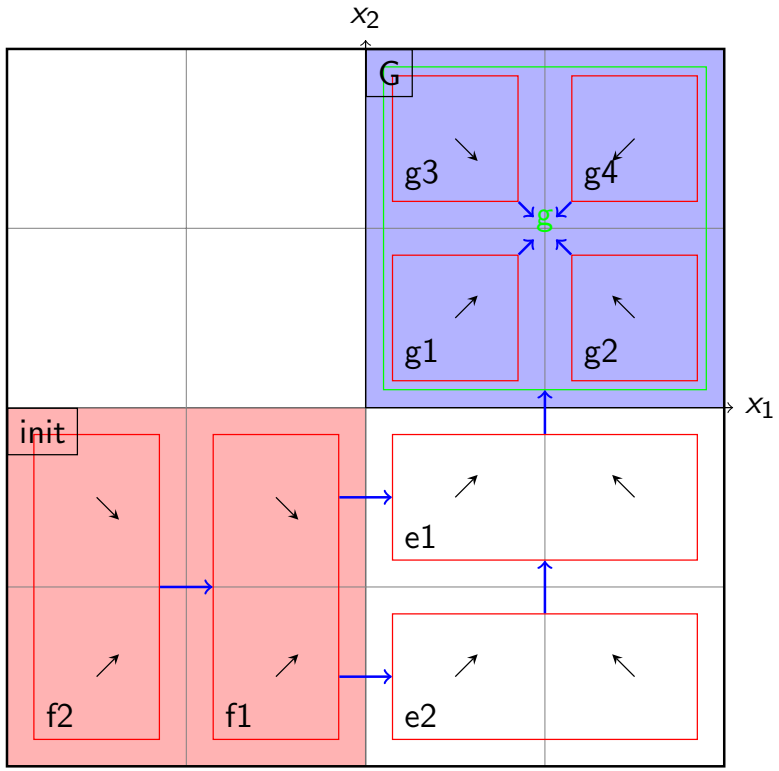
Backward Reachability Algorithm - $\varphi = \Diamond\Box G$



Backward Reachability Algorithm - $\varphi = \Diamond\Box G$

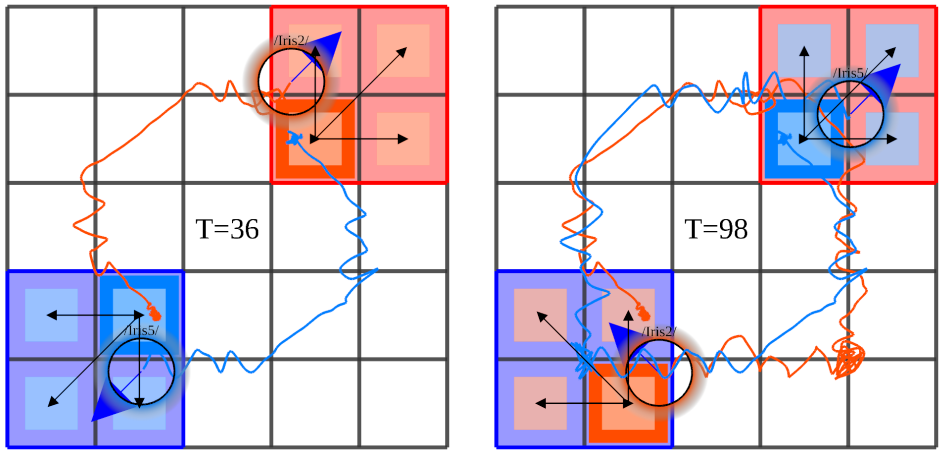


Backward Reachability Algorithm - $\varphi = \Diamond\Box G$



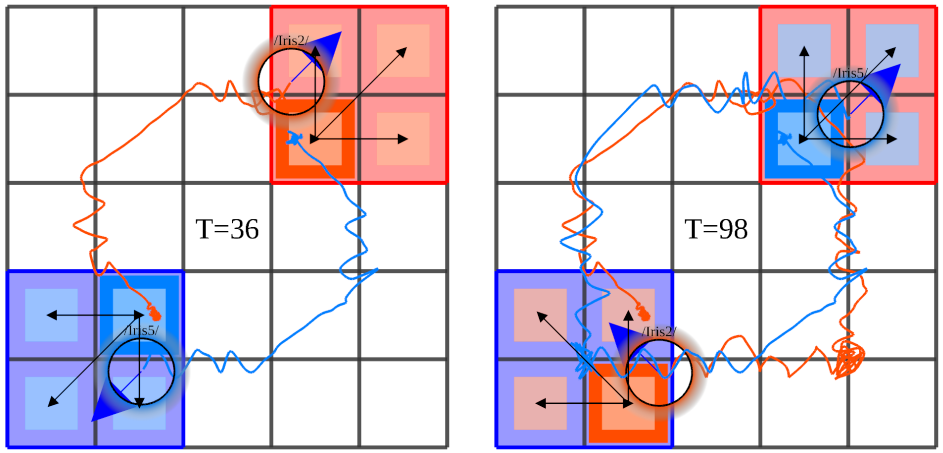
Multi agent experiment

- 0 memories (equivalent to the first integrator model)
- $\varphi = (\Box\neg out) \wedge (\Box\Diamond a) \wedge (\Box\Diamond b) \wedge (\Box\neg collide)$



Multi agent experiment

- 0 memories (equivalent to the first integrator model)
- $\varphi = (\Box\neg out) \wedge (\Box\Diamond a) \wedge (\Box\Diamond b) \wedge (\Box\neg collide)$



The video !

Tack så mycket!

Tack så mycket!
Du kan fråga vad du vill, jag ska svara (bara om det är möjligt).