Model Checking for LTL – Part 2

SCIENCE
PASSION
TECHNOLOGY

Bettina Könighofer
bettina.koenighofer@tugraz.at

Plan for Today

- Presentation of Homework
- Part 1 - LTL Model Checking
  - Generalized Büchi Automata
  - Translation of LTL to Büchi Automata
- Part 2 – Shielded Reinforcement Learning

Algorithm: Intersection of Büchi Automata (last week)

- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ is defined as follows:
  - $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$
  - $Q^0 = Q_1^0 \times Q_2^0 \times \{0\}$
  - $F = Q_1 \times Q_2 \times \{2\}$
  - $((q_1,q_2,x), a, (q'_1,q'_2,x')) \in \Delta \Leftrightarrow$
    1. $(q_1,a,q'_1) \in \Delta_1$ and $(q_2,a,q'_2) \in \Delta_2$ and
    2. If $x=0$ and $q'_1 \in F_1$ then $x'=1$
       If $x=1$ and $q'_2 \in F_2$ then $x'=2$
       If $x=2$ then $x'=0$
       Else, $x'=x$

Intuition:
$x=0$ … waiting for $s \in F_1$
$x=1$ … waiting for $s \in F_2$

If some s with $x=2$ is visited inf often, then states from F_1 and states from F_2 have been visited inf often.

- **Homework: Define the transition relation for $\mathcal{B}$ using $x \in \{0, 1\}$**

Algorithm: Intersection of Büchi Automata (last week)

- $\mathcal{B} = (\mathbf{\Sigma},\mathbf{Q},\mathbf{\Delta},\mathbf{Q}^0,\mathbf{F})$ s.t. $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ is defined as follows:
  - $\mathbf{Q} = \mathbf{Q}_1 \times \mathbf{Q}_2 \times \{0, 1\}$
  - $\mathbf{Q}^0 = \mathbf{Q}_1^{\ 0} \times \mathbf{Q}_2^{\ 0} \times \{0\}$
  - $\mathbf{F} = \mathbf{Q}_1 \times \mathbf{Q}_2 \times \{2\}$
  - $\mathbf{F} = \mathbf{F}_1 \times \mathbf{Q}_2 \times \{0\}$
  - $((q_1,q_2,x), a, (q'_1,q'_2,x')) \in \mathbf{\Delta} \Leftrightarrow$
    1. $(q_1,a,q'_1) \in \mathbf{\Delta}_1$ and $(q_2,a,q'_2) \in \mathbf{\Delta}_2$ and
    2. If $x=0$ and $q'_1 \in \mathbf{F}_1$ then $x'=1$
       If $x=1$ and $q'_2 \in \mathbf{F}_2$ then **$x'=0$**
       Else, $x'=x$

Intuition:
$x=0$ … waiting for $s \in \mathbf{F}_1$
$x=1$ … waiting for $s \in \mathbf{F}_2$

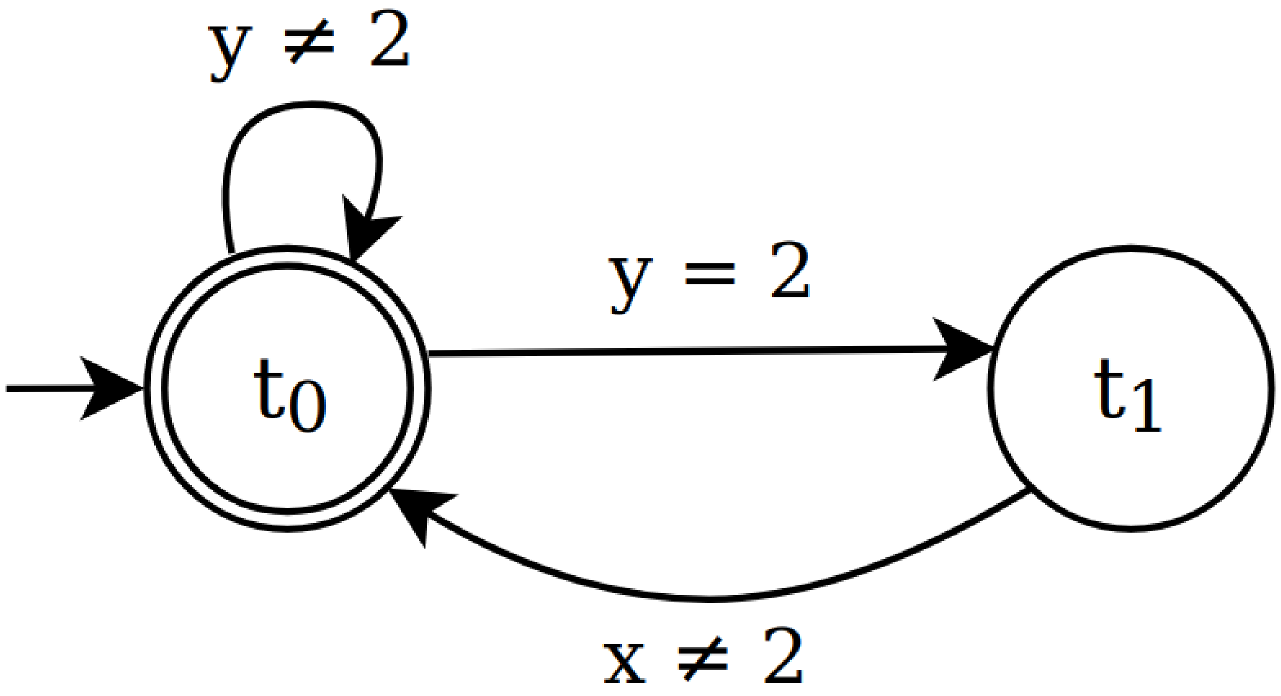
If some s with $x=2$ is visited inf often, then states from $\mathbf{F}_1$ and states from $\mathbf{F}_2$ have been visited inf often.

2b) 1. Construct $\neg\varphi_2$

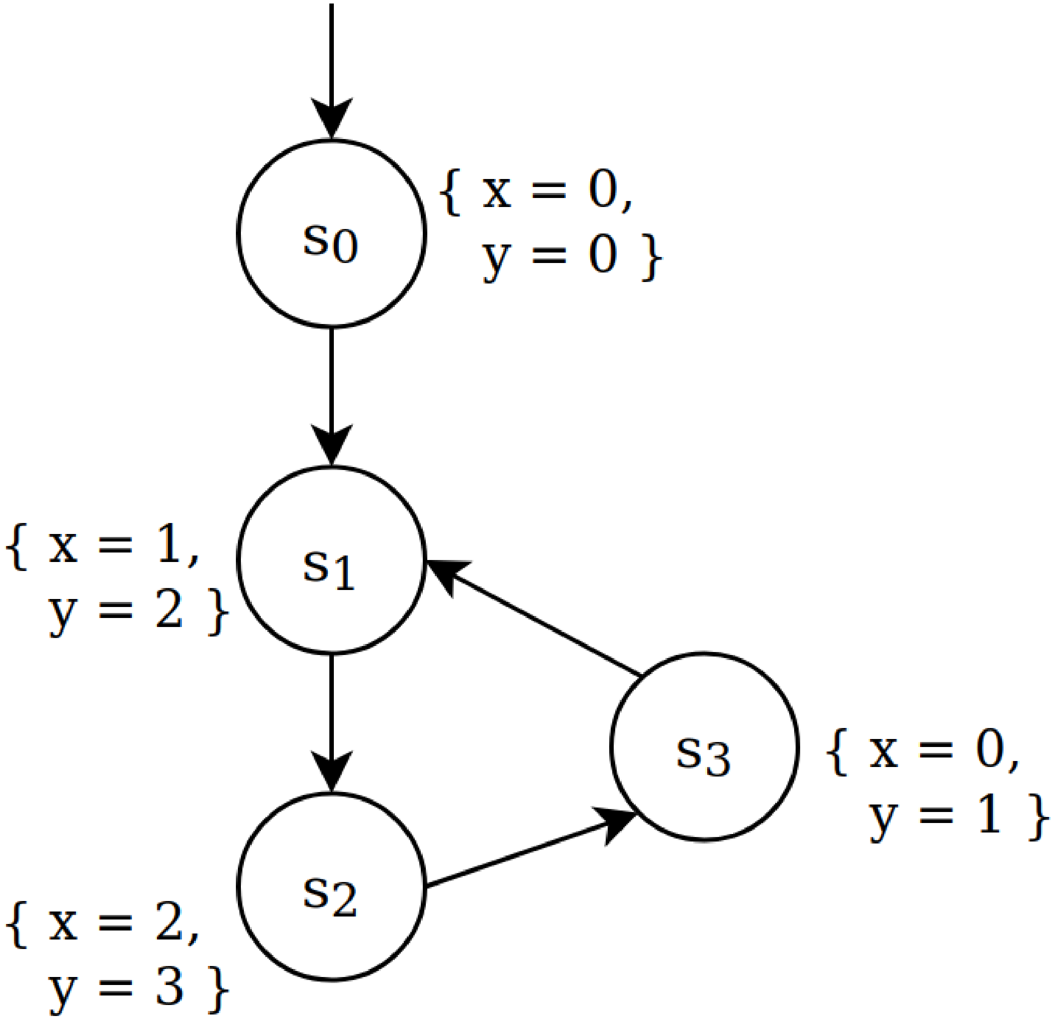
$\neg\varphi_2 \equiv \neg[\mathbf{F}\ ((y = 2) \land \mathbf{X}(x = 3))]$

2b) 2. Construct Büchi automaton $\mathcal{S}_{\neg\varphi}$

(already given)



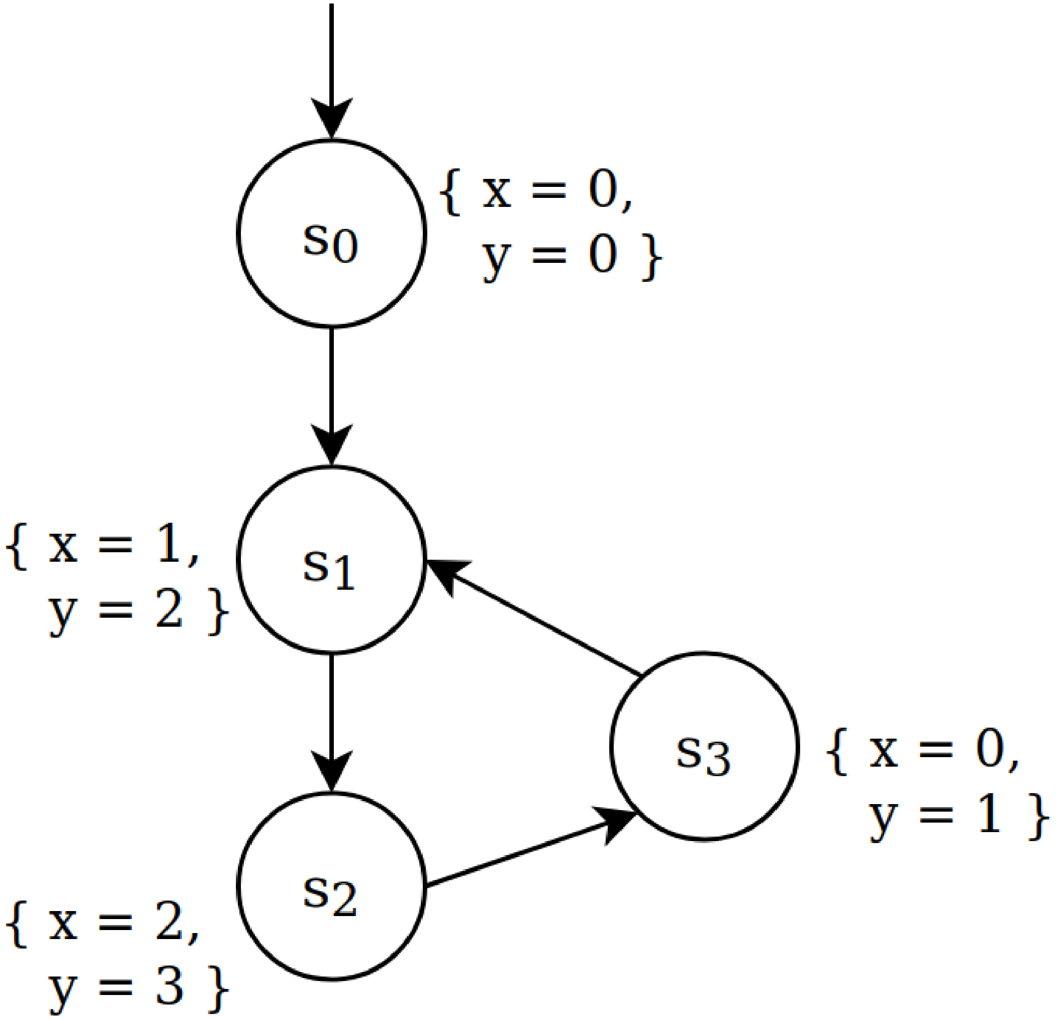
2b) 3. Translate M to an automaton $\mathcal{A}$

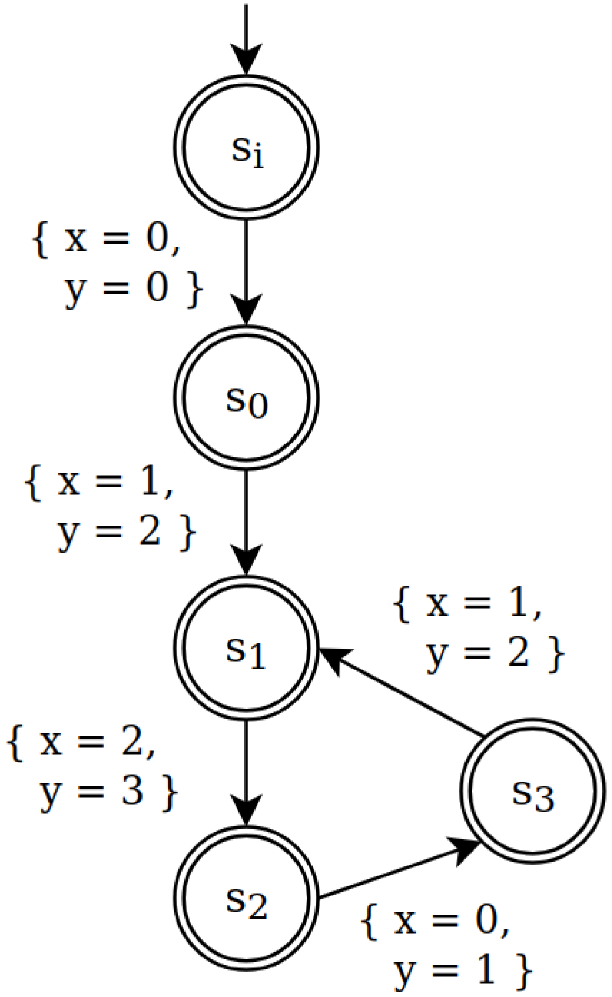


Reminder

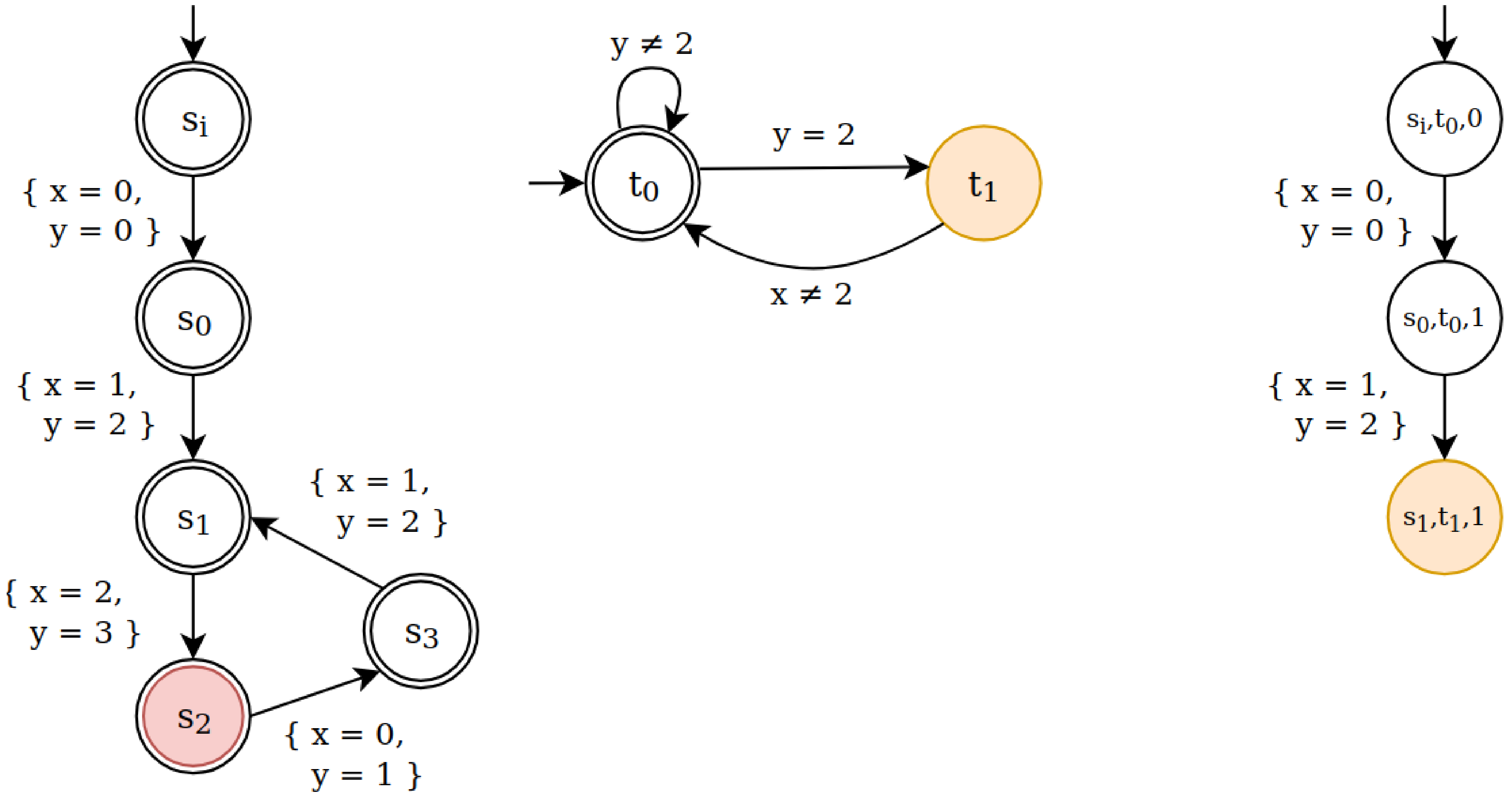
- Move labels to incoming transitions
- All states are accepting

2b) 3. Translate M to an automaton $\mathcal{A}$

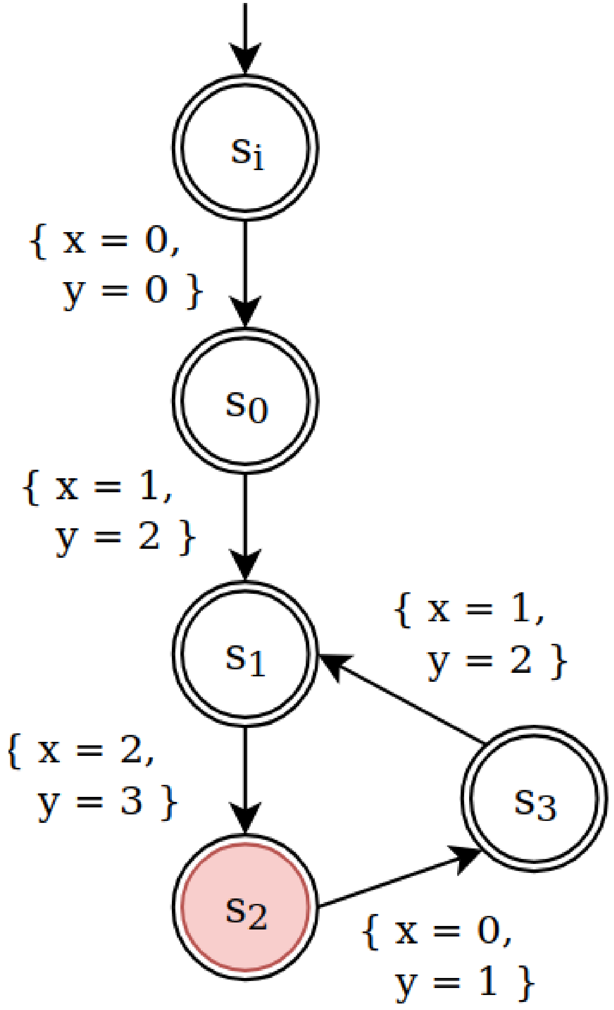


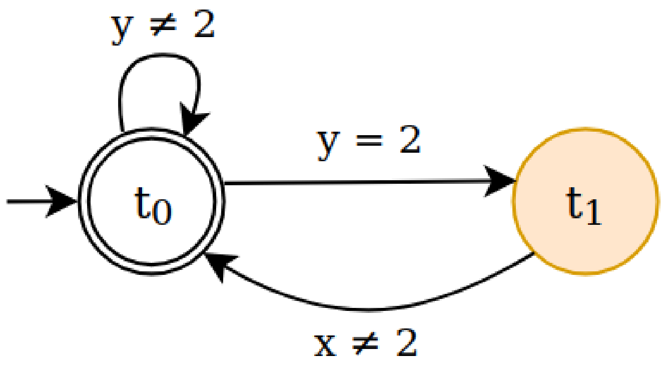


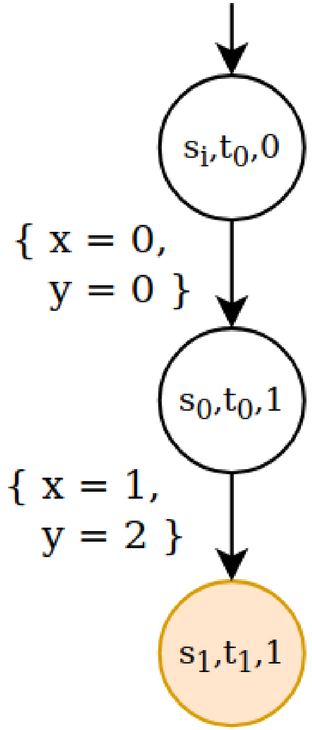
2b) 4. Construct automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$



2b) 5. $\mathcal{L}(\mathcal{B}) = \emptyset$?







If $\mathcal{L}(\mathcal{B}) = \emptyset \implies M \models \phi_2$

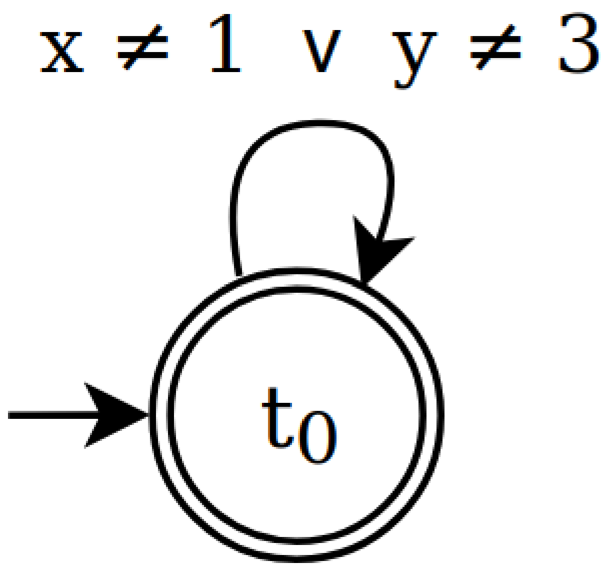
- $\mathcal{L}(\mathcal{B}) = \emptyset$ is evident, as $F_{\mathcal{B}} = \emptyset$.
- Thus, $M \models \phi_2$ holds.

2a) 1. Construct $\neg\varphi_1$

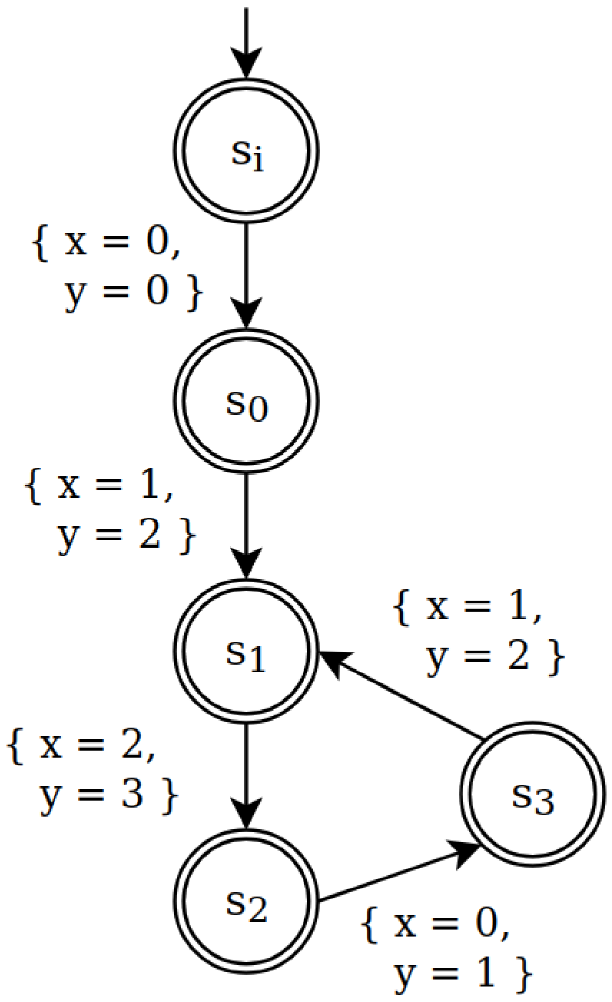
$\neg\varphi_1 \equiv \neg[\mathbf{F}\ ((x = 1) \land (y = 3))]$

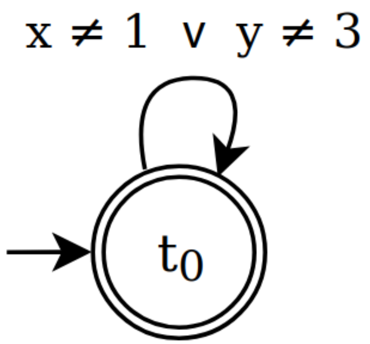
2a) 2. Construct Büchi automaton $\mathcal{S}_{\neg\varphi}$

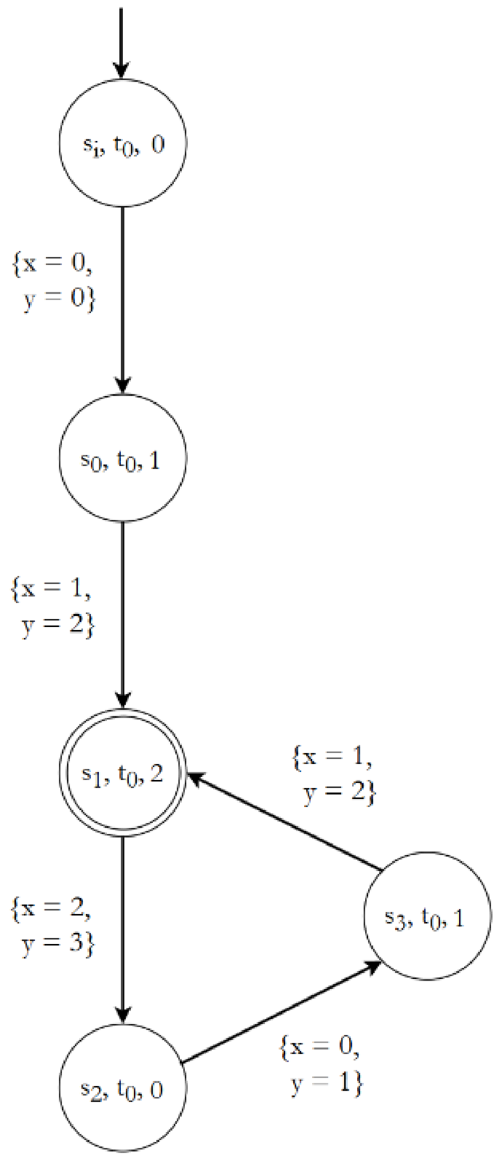
(already given)



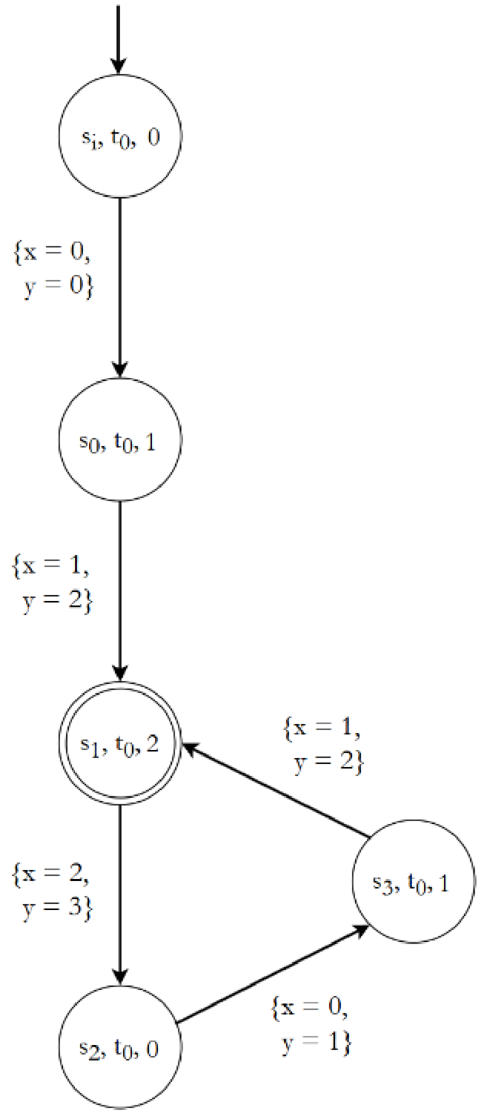
2a) 4. Construct automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$







2a) 5. $\mathcal{L}(\mathcal{B}) = \emptyset$?



A counterexample $v \cdot w^\omega \in \mathcal{L}(\mathcal{B})$ exists

- A counter example for ϕ_1 is

$$\{x = 0, y = 0\} \cdot (\{x = 1, y = 2\} \cdot \{x = 2, y = 3\} \cdot \{x = 0, y = 1\})^\omega$$

- Thus, $M \not\models \phi_1$.

Plan for Today

- Presentation of Homework
- Part 1 - LTL Model Checking
  - Generalized Büchi Automata
  - Translation of LTL to Büchi Automata
- Part 2 – Shielded Reinforcement Learning

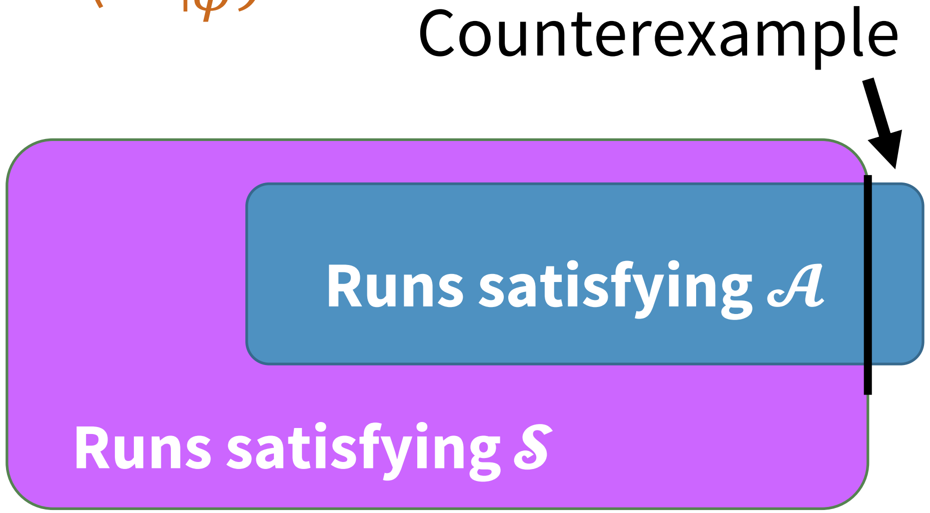
Model Checking of LTL

- Given a Kripke structure M and a LTL formula φ: Does $M \models \varphi$?

- **Automata-based Algorithm**

Today!

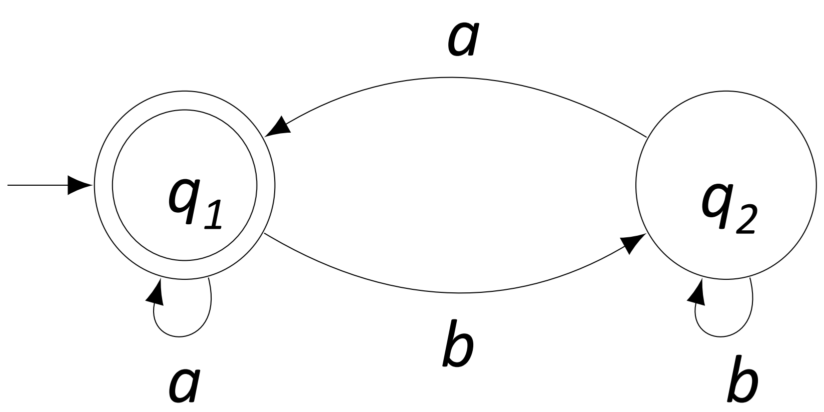
1. Construct $\neg\varphi$
2. **Construct a Büchi automaton $\mathcal{S}_{\neg\varphi}$**
3. Translate M to an automaton $\mathcal{A}$.
4. Construct the automaton $\mathcal{B}$ with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{S}_{\neg\varphi})$
5. If $\mathcal{L}(\mathcal{B}) = \emptyset \Rightarrow M \models \varphi$
6. If $\mathcal{L}(\mathcal{B}) \neq \emptyset \Rightarrow M \not\models \varphi$.
   A word $v \cdot w^{\omega} \in \mathcal{L}(\mathcal{B})$ is a **counterexample**
   → a trace in M that does not satisfy φ



Büchi Automata

- $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$

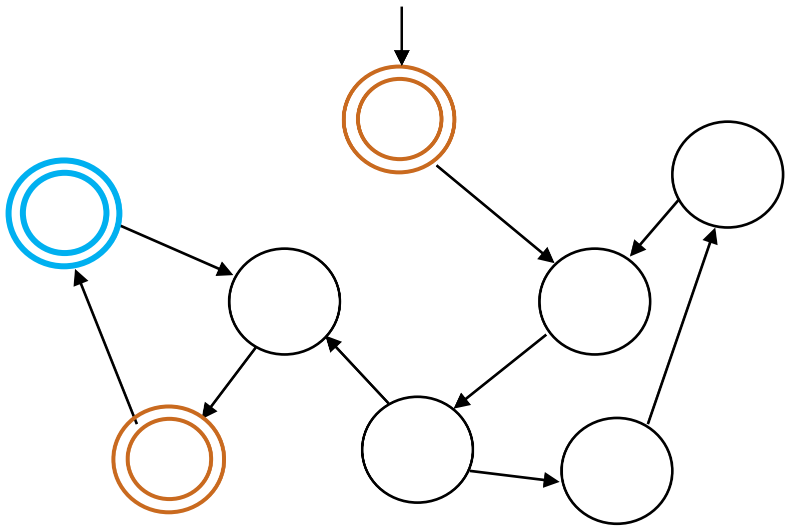
- An **run ρ** is **accepting** ⇔ **ρ** visits an **accepting state infinitely** often.



$\mathcal{L}(\mathcal{B})$ = {words with infinitely many a}

Generalized Büchi Automata

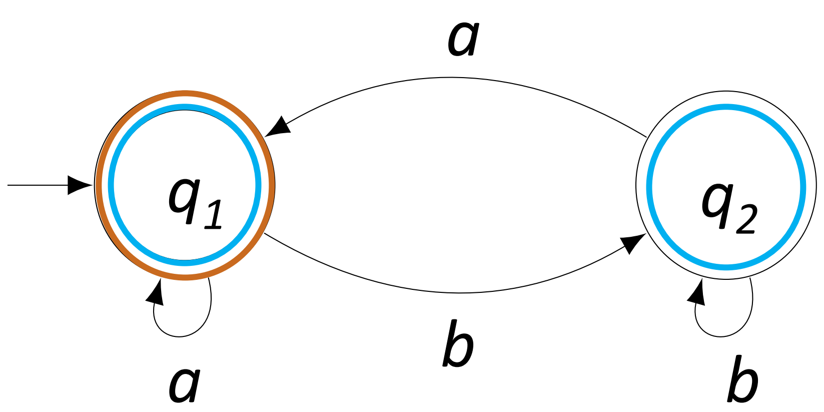
- Have several sets of accepting states
- $\mathcal{B} = (\mathbf{\Sigma}, \mathbf{Q}, \mathbf{\Delta}, \mathbf{Q}^0, \mathbf{F})$
  - $\mathbf{F} = \{F_1, \ldots, F_k\}$, where for every $1 \leq i \leq k, F_i \subseteq Q$
- A run ρ of $\mathcal{B}$ is accepting if for each $F_i \in F,\ \inf(\rho) \cap F_i \neq \emptyset$



Generalized Büchi Automata

- A run ρ of $\mathcal{B}$ is accepting if for each $F_i \in F,\ \inf(\rho) \cap F_i \neq \emptyset$

- What words are accepted?
  a. The infinite word b^ω? ✗
  b. The infinite word a^ω? ✓
  c. The infinite word $(ab)^\omega$? ✓

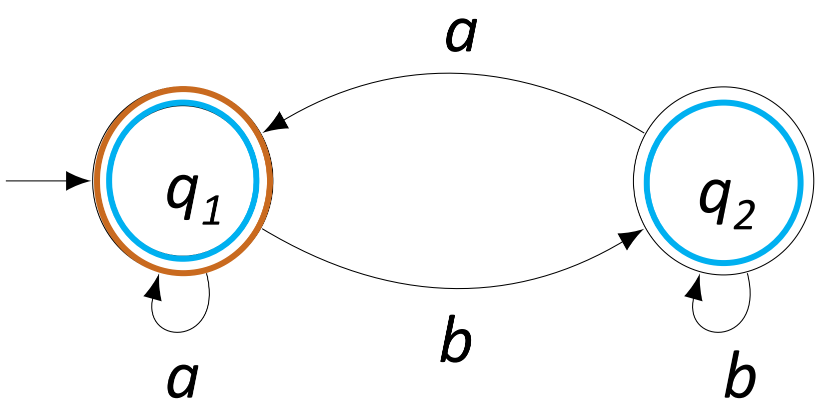


$F_1 = \{q_1, q_2\}, \quad F_2 = \{q_1\}$

Algorithm: Generalized Büchi To Büchi Automata

- Given generalized Büchi Automaton $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ with $\mathbf{F} = \{F_1, \ldots, F_k\}$
- Construct Büchi Automaton $\mathcal{B}'$ that accepts the same language

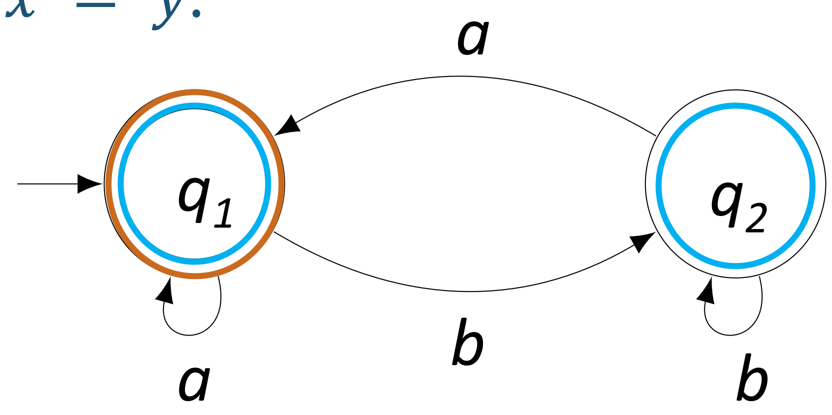
- **Idea:**
  - Introduce counter from 1 … k → k copies of the state space
  - In copy i we wait for accepting state in F_i
  - When F_i is visited in copy i, redirect edges to move to copy $i + 1$ (from F_k to copy $i = 1$)
  - → A cycle through all copies will contain accepting states from each set $F_1, \ldots, F_k$



Algorithm: Generalized Büchi To Büchi Automata

- Given generalized Büchi Automaton $\mathcal{B} = (\Sigma, Q, \Delta, Q^0, F)$ with $\mathbf{F} = \{F_1, \dots, F_k\}$
- Construct Büchi Automaton $\mathcal{B}'$ that accepts the same language

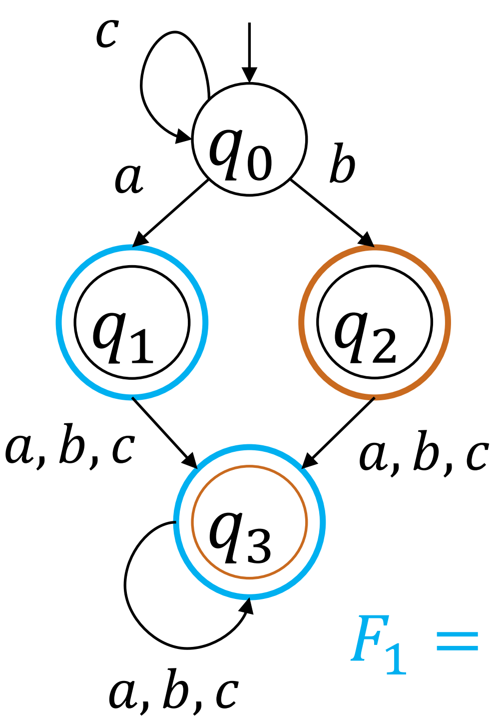
- $\mathcal{B}' = (\Sigma, Q \times \{1, \dots, k\}, \Delta', Q^0 \times 1, F_k \times k)$ with:
- Δ': $\big((q, x), a, (q', y)\big) \in \Delta'$ if
  - $(q, a, q') \in \Delta$
  - If $q \in F_i$ and $x = i$, then $y = i + 1$ for $i < k$
  - If $q \in F_k$ and $x = k$, then $y = 1$
  - Otherwise, $x = y$.



Size of $\mathcal{B}'$ = (size of $\mathcal{B}$)×k

Example: Generalized Büchi To Büchi Automata

- $\mathcal{L}(\mathcal{B}) = c^*(a|b)(a|b|c)^\omega$

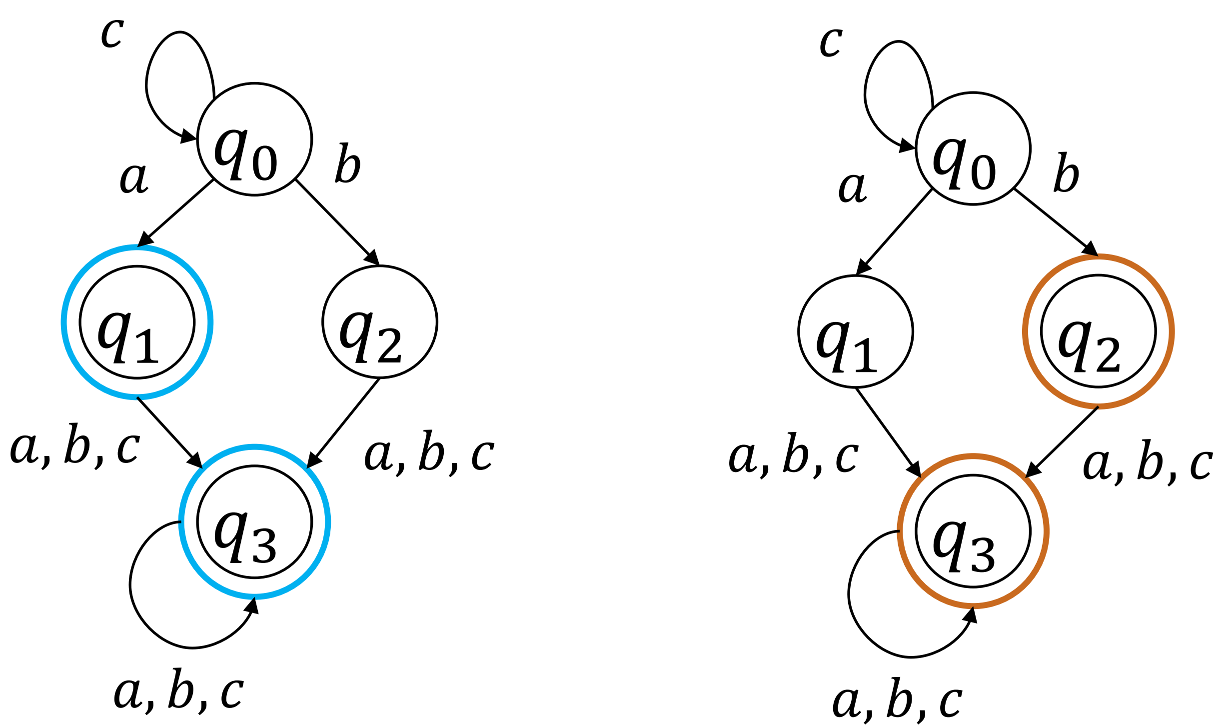


$F_1 = \{q_1, q_3\}$, $F_2 = \{q_2, q_3\}$

Example: Generalized Büchi To Büchi Automata

- Translate generalized Büchi Automaton $\mathcal{B}$ to a Büchi automaton $\mathcal{B}$'

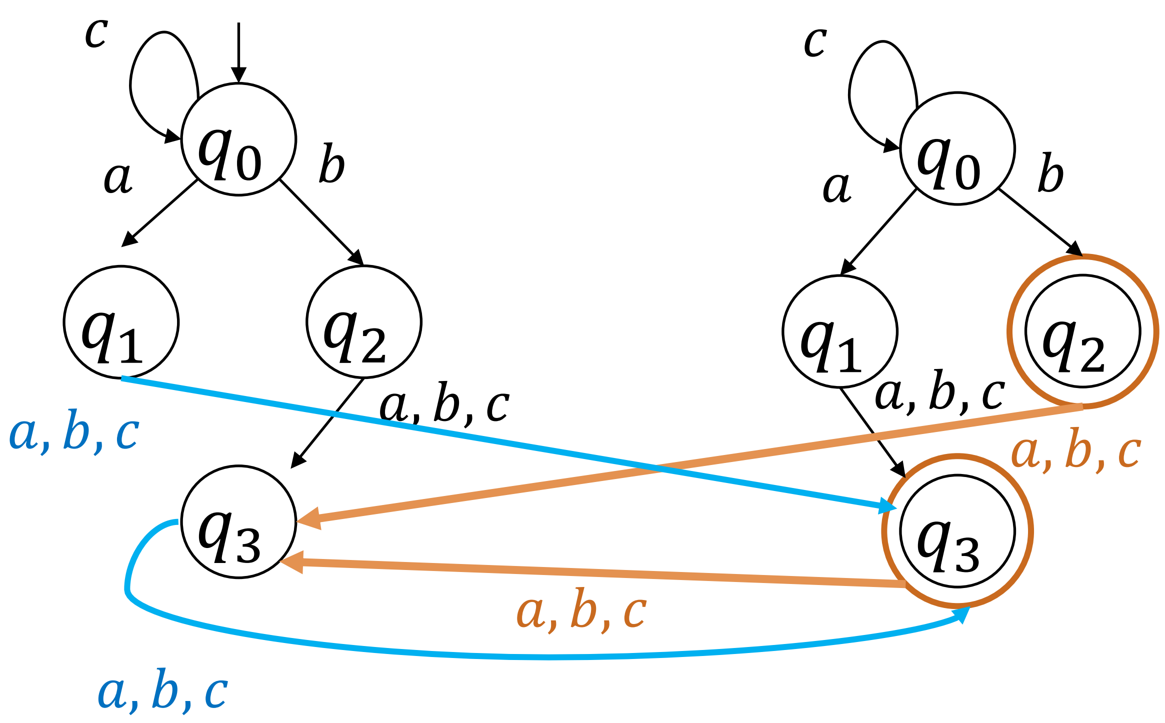
1. Create two copies, since we have two accepting sets



Example: Generalized Büchi To Büchi Automata

- Translate generalized Büchi Automaton $\mathcal{B}$ to a Büchi automaton $\mathcal{B}$'

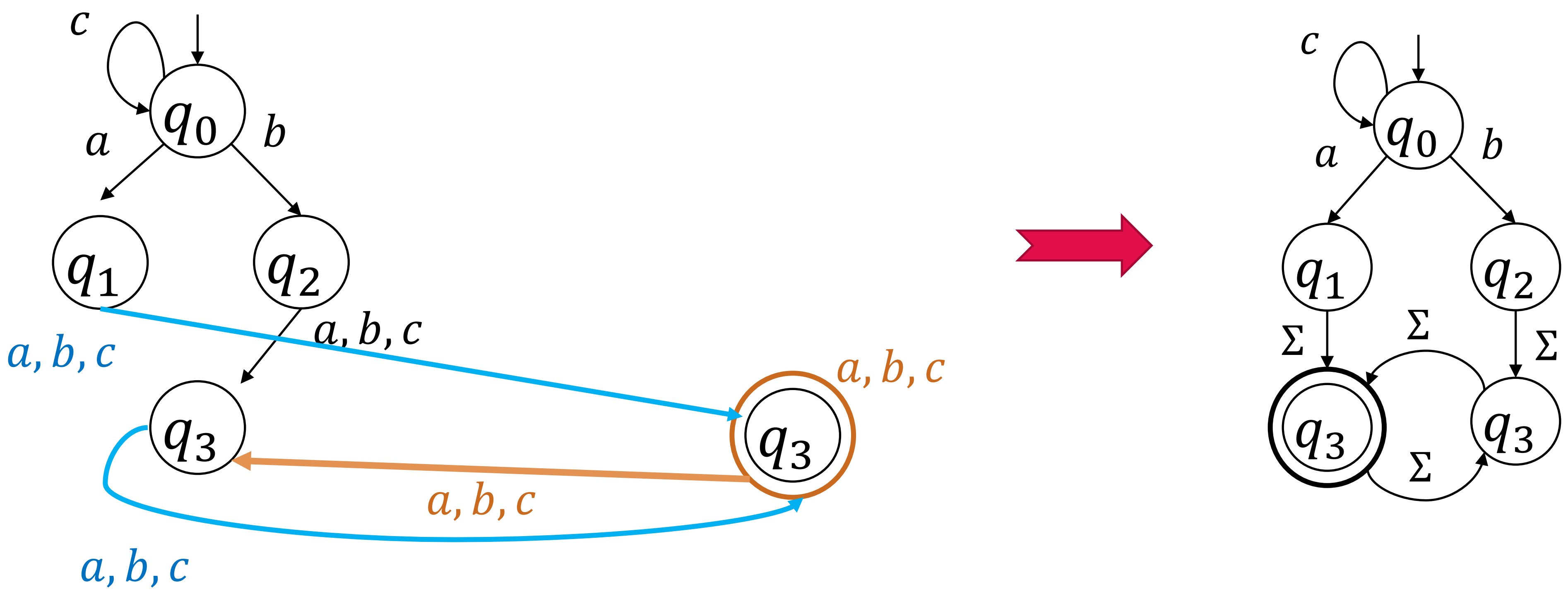
4. Only one copy is accepting



Example: Generalized Büchi To Büchi Automata

- Translate generalized Büchi Automaton $\mathcal{B}$ to a Büchi automaton $\mathcal{B}$'

4. Only one copy is accepting
5. Remove unreachable states



Plan for Today

- Presentation of Homework

- Part 1 - LTL Model Checking
  - Generalized Büchi Automata
  - **Translation of LTL to Büchi Automata**

- Part 2 – Reactive Synthesis
  - Safety Games
  - Reachability Games
  - Büchi Games

Translation of LTL to Büchi Automata

- Today: discuss simple algorithm from Vardi and Wolper (book, page 98)
- Size of automaton always exponential in the size of the specification

M.Y. Vardi, and P. Wolper.
An automata-theoretic approach to automatic program verification.
In Logic in Computer Science (LICS), pages 332-344, 1986

- More efficient algorithm by Gerth, Peled, Vardi and Wolper (book, page 101)

R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper.
Simple on-the-fly automatic verification of linear temporal logic.
In Protocol Specification Testing and Verification, pages 3-18, 1995

Algorithm: LTL to Büchi Automata

- **Input:** LTL specification φ
- **Output:** Büchi automaton $\mathcal{A}_\varphi$ s. t. $\mathcal{A}_\varphi$ accepts exactly all the traces that satisfy φ

- **Steps of the Algorithm:**
  1. Rewrite of φ to use only $\neg, \wedge, \vee, X, U$ operators
     - via rewriting rules e.g., $F\varphi = true\ U\varphi$, $G\varphi = \neg F \neg \varphi$ etc ...
  2. **Translate φ into generalized Büchi Automaton** ⬅
  3. Translate generalized Büchi to Büchi automaton

LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- **Input:** LTL specification φ
- **Output:** Büchi automaton $\mathcal{A}_\varphi$ s. t. $\mathcal{A}_\varphi$ accepts exactly all the traces that satisfy φ

- Step 1: Defining the **state space** of $\mathcal{A}_\varphi$:
  - Idea: Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**
  - Algorithm:
    1. Build the **closure cl(**φ**)** of $\varphi \equiv$ subformulas of φ and their negation
       - $\varphi \in cl(\varphi)$.
       - If $\varphi_1 \in cl(\varphi), then\ \neg\varphi_1 \in cl(\varphi)$.
       - If $\neg\varphi_1 \in cl(\varphi), then\ \varphi_1 \in cl(\varphi)$.
       - If $\varphi_1 \vee \varphi_2 \in cl(\varphi), then\ \varphi_1 \in cl(\varphi)\ and\ \varphi_2 \in cl(\varphi)$.
       - If $X\ \varphi_1 \in cl(\varphi), then\ \varphi_1 \in cl(\varphi)$.
       - If $\varphi_1\ U\ \varphi_2 \in cl(\varphi), then\ \varphi_1 \in cl(\varphi)\ and\ \varphi_2 \in cl(\varphi)$.

LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- **Input:** LTL specification φ
- **Output:** Büchi automaton $\mathcal{A}_\varphi$ s. t. $\mathcal{A}_\varphi$ accepts exactly all the traces that satisfy φ

- Step 1: Defining the **state space** of $\mathcal{A}_\varphi$:
  - Idea: Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**
  - Algorithm:
    1. Build the **closure cl(**φ**)** of $\varphi \equiv$ subformulas of φ and their negation
    2. Compute the **good sets** $S \subseteq cl(\varphi)$ $\equiv$ **maximal sets** of formulas in $cl(\varphi)$ that are **consistent**
       - For all $\varphi_1 \in cl(\varphi)$: $\varphi_1 \in S \Leftrightarrow \neg \varphi_1 \notin S$,
       - For all $\varphi_1 \vee \varphi_2 \in cl(\varphi)$: at least one of φ_1, φ_2 is in S.

LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- **Input:** LTL specification φ
- **Output:** Büchi automaton $\mathcal{A}_\varphi$ s. t. $\boldsymbol{\mathcal{A}_\varphi}$ accepts exactly all the traces that satisfy φ

- Step 1: Defining the **state space** of $\mathcal{A}_\varphi$:
  - Idea: Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**
  - Algorithm:
    1. Build the **closure cl(φ)** of $\varphi \equiv$ subformulas of φ and their negation
    2. Compute the **good sets** $\boldsymbol{S} \subseteq cl(\varphi) \equiv$ **maximal sets** of formulas in *cl(φ)* that are **consistent**
    3. All **good sets of cl(φ)** define the state space of $\boldsymbol{\mathcal{A}_\varphi}$

LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- **Example: Define the state space** of $\mathcal{A}_\varphi$:
  - $\varphi \coloneqq \neg h \,\mathrm{U}\, c$
  - Algorithm:
    1. Build the **closure cl(φ)** of φ ≡ subformulas of φ and their negation
    2. Compute the **good sets *S*** ⊆ *cl*(φ) ≡ **maximal sets** of formulas in *cl(φ)* that are **consistent**
    3. All **good sets of cl(φ)** define the state space of $\mathcal{A}_\varphi$
  - Solution:

  - $cl(\varphi) \coloneqq \{h, \neg h, c, \neg c, \varphi, \neg\varphi\}$
  - $Q = \{\{h, c, \varphi\}, \{\neg h, c, \varphi\}, \{h, c, \varphi\}, \{\neg h, \neg c, \varphi\},$
    $\{h, c, \neg\varphi\}, \{\neg h, c, \neg\varphi\}, \{h, c, \neg\varphi\}, \{\neg h, \neg c, \neg\varphi\}\}$

LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- **Input:** LTL specification φ
- **Output:** Büchi automaton $\mathcal{A}_\varphi$ s.t. $\mathcal{A}_\varphi$ accepts exactly all the traces that satisfy φ

- Step 1: Defining the **state space** of $\mathcal{A}_\varphi$
  - Idea: Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**
- Step 2: Defining the **transition relation** of $\mathcal{A}_\varphi$

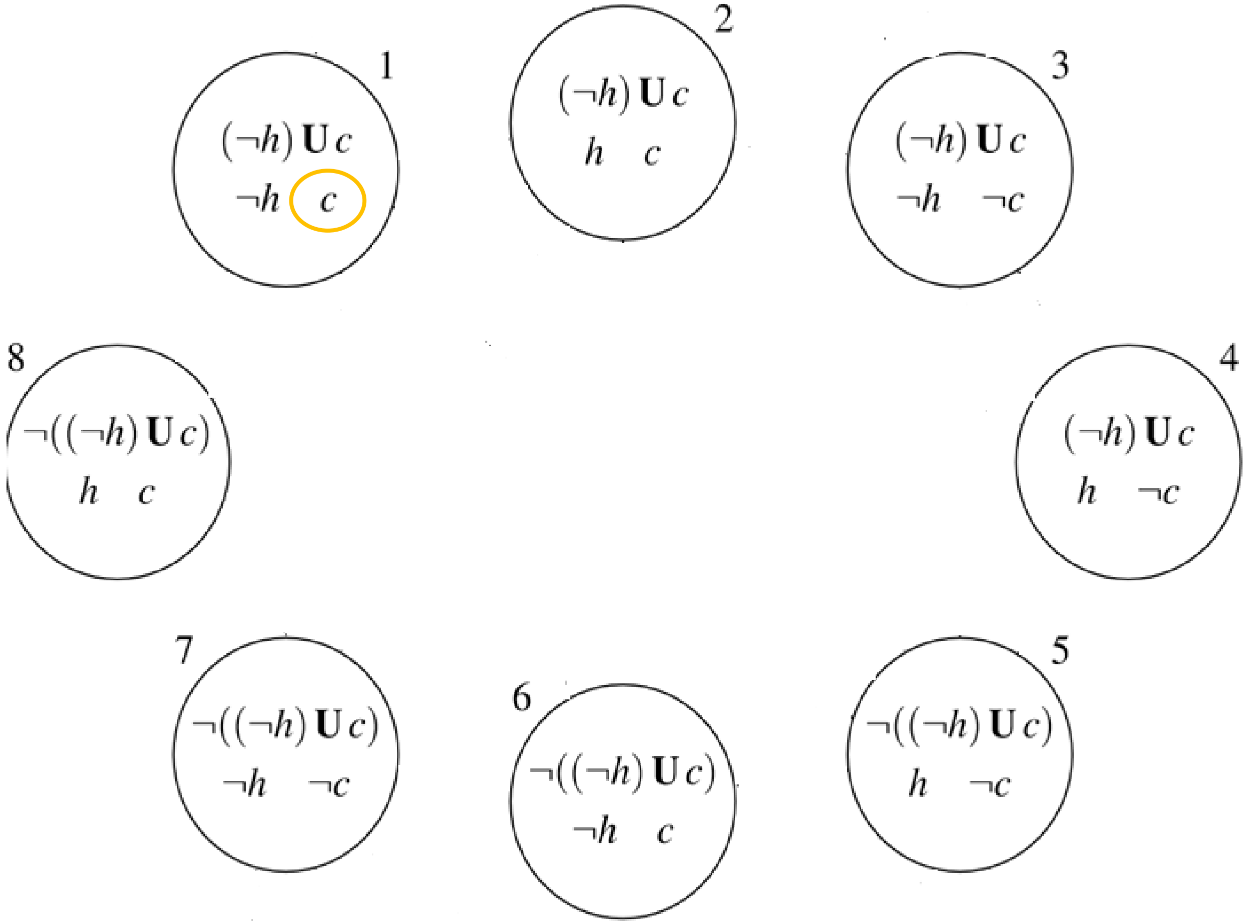
LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- $\mathcal{A}_\varphi = (\Sigma, \mathbf{Q}, \mathbf{\Delta}, \mathbf{Q}^0, \mathbf{F})$
- $\mathbf{Q}$ = set of all the good sets in *cl(φ)*
  - *Idea:* Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**

- For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:
  - $\sigma = q' \cap AP$
  - For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$: if $\boldsymbol{X}\varphi_1 \in q$ then $\varphi_1 \in q'$
  - For all $\varphi_1 U \varphi_2 \in cl(\varphi)$: if $\varphi_1 \,\mathrm{U}\, \varphi_2 \in$ q then either $\varphi_2 \in$ q or **both** $\varphi_1 \in$ q **and** $\varphi_1 \,\mathrm{U}\, \varphi_2 \in q'$

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi := \neg h \; \mathrm{U}\, c$
- Draw the transitions of $\mathcal{A}_\varphi$

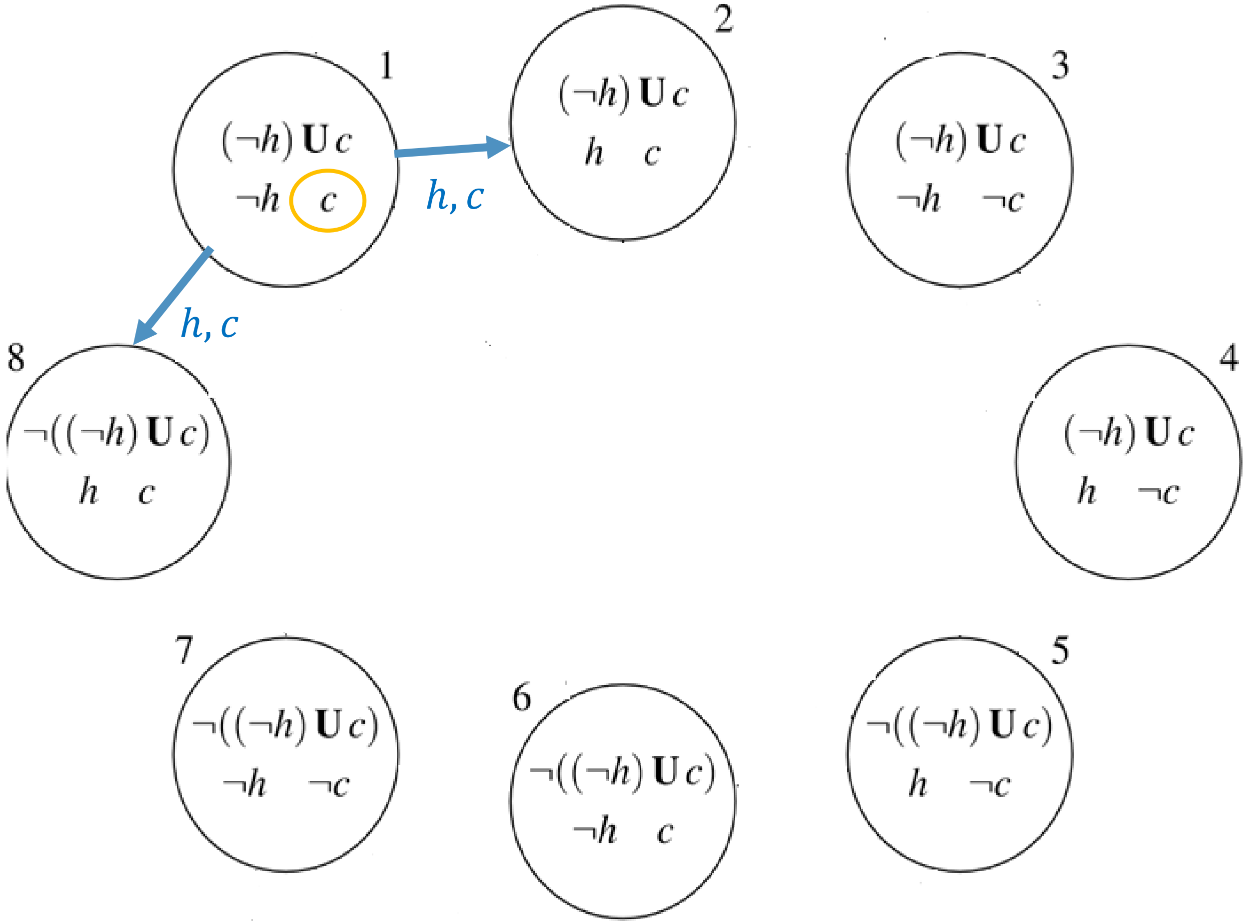


For $q, q' \in Q$ and $\sigma \subseteq AP$, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$:
  - $\boldsymbol{X}\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U \varphi_2 \in cl(\varphi)$:
  - $\varphi_1 \mathrm{U} \varphi_2 \in q \Leftrightarrow$ either $\varphi_2 \in q$ or both $\varphi_1 \in q$ **and** $\varphi_1 \mathrm{U} \varphi_2 \in q'$

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi := \neg h \ \mathrm{U}\ c$
- Draw the transitions of $\mathcal{A}_\varphi$

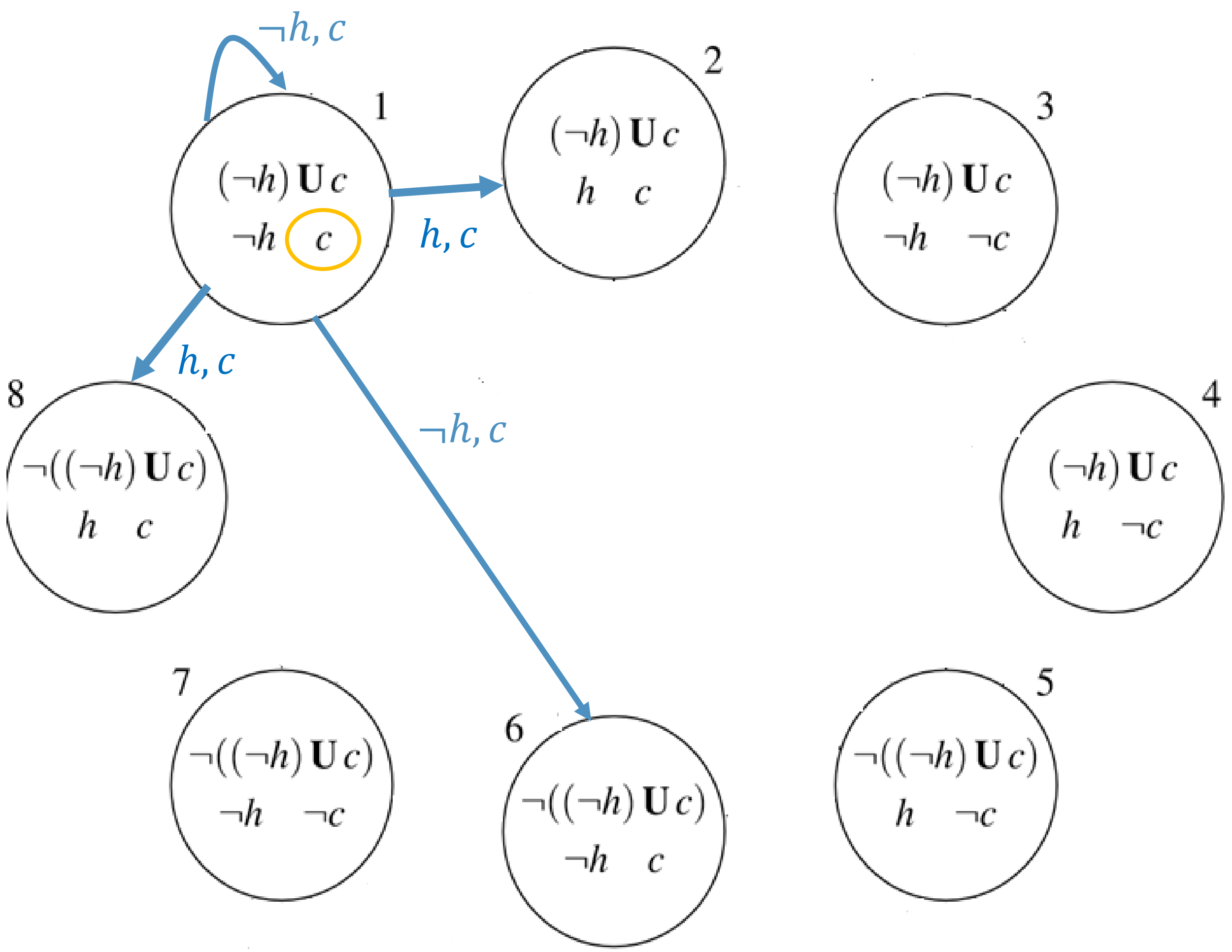


For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $X\varphi_1 \in cl(\varphi)$:
  - $X\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U\, \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in$ q'

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi := \neg h \text{ U } c$
- Draw the transitions of $\mathcal{A}_\varphi$



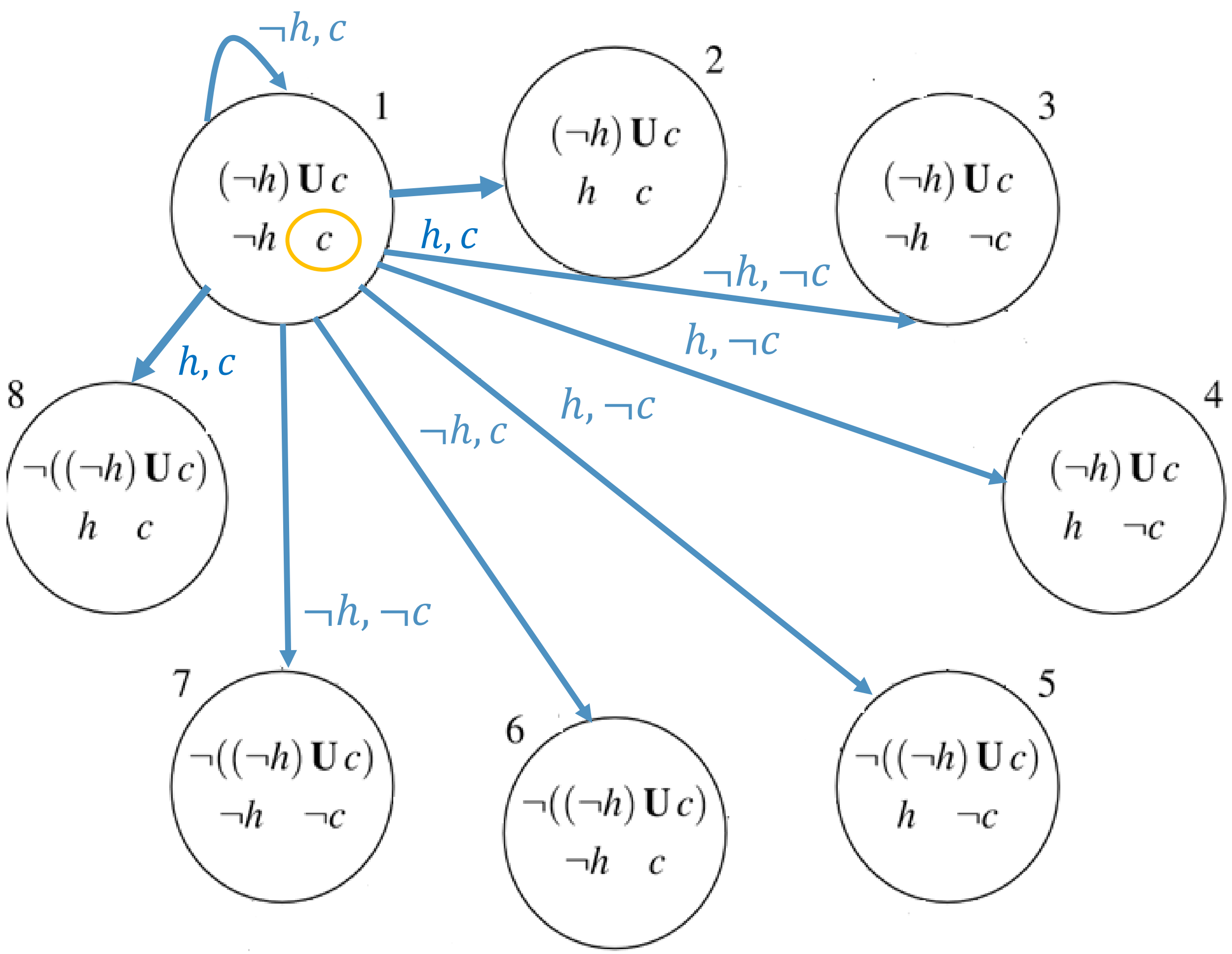
For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:
- $\sigma = q' \cap AP$
- For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$:
  - $\boldsymbol{X}\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in$ q'

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi := \neg h \text{ U } c$
- Draw the transitions of $\mathcal{A}_\varphi$

For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$:
  - $\boldsymbol{X}\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in$ q'

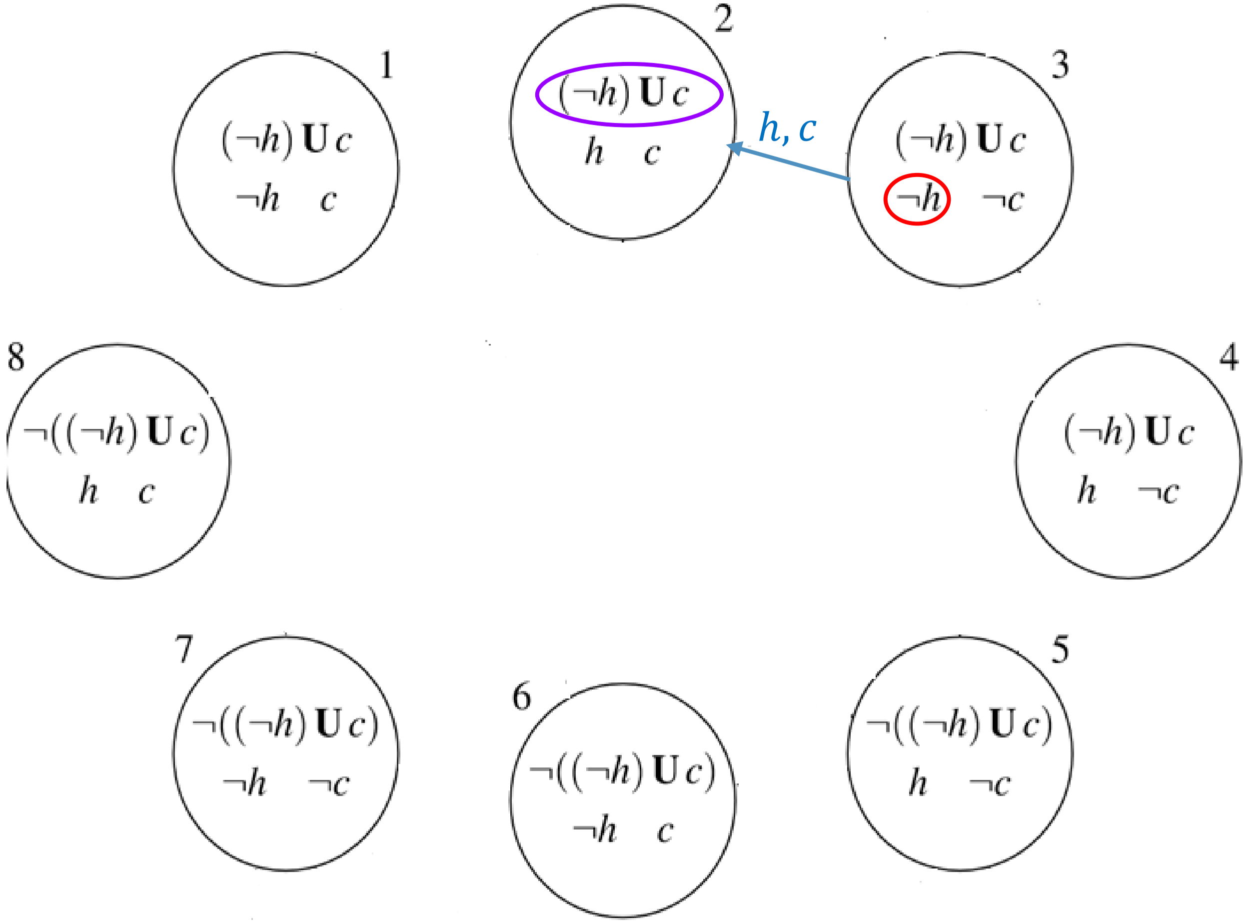


Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi := \neg h \text{ U } c$
- Draw the transitions of $\mathcal{A}_\varphi$

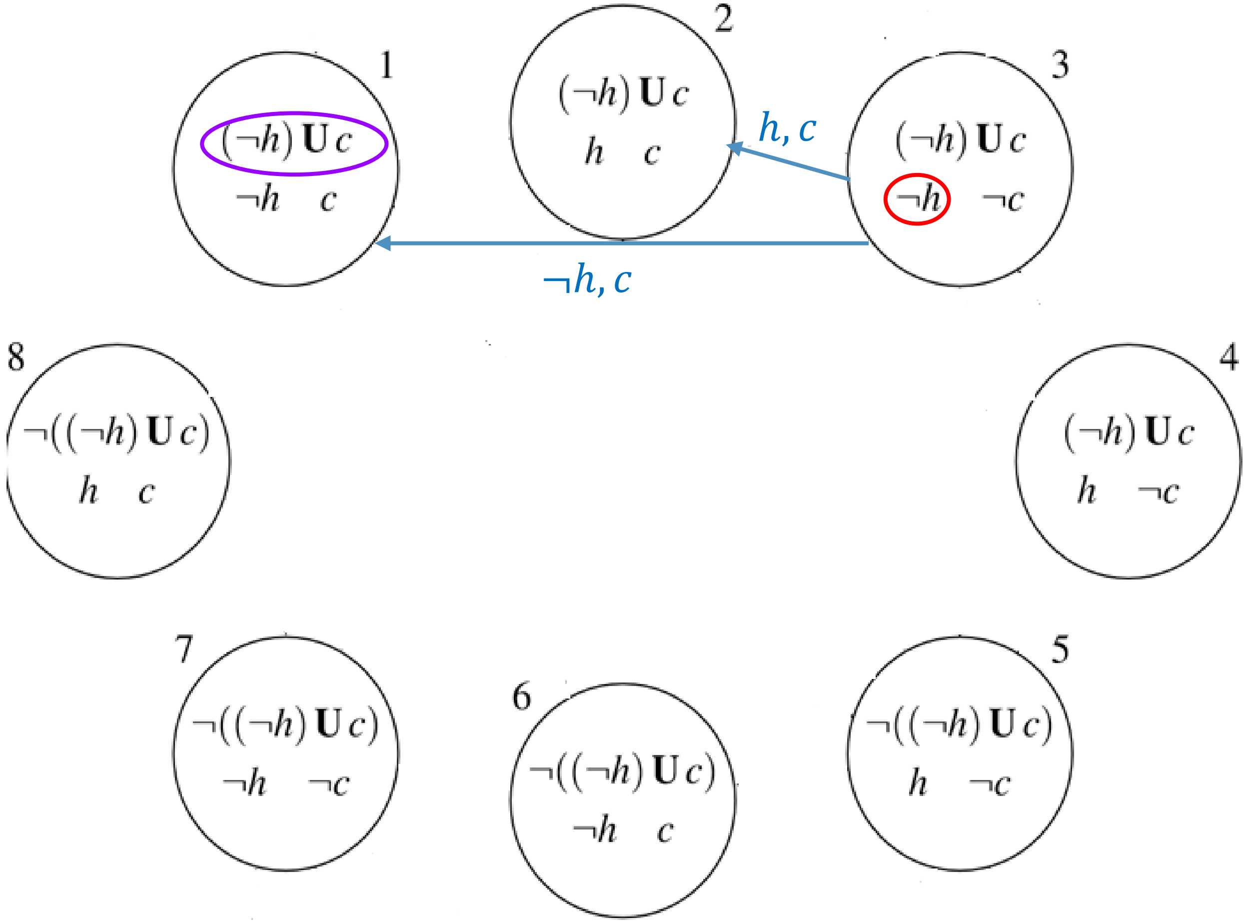
For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $X\varphi_1 \in cl(\varphi)$:
  - $X\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in$ q'



Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi \coloneqq \neg h \; \mathrm{U} \; c$
- Draw the transitions of $\mathcal{A}_\varphi$

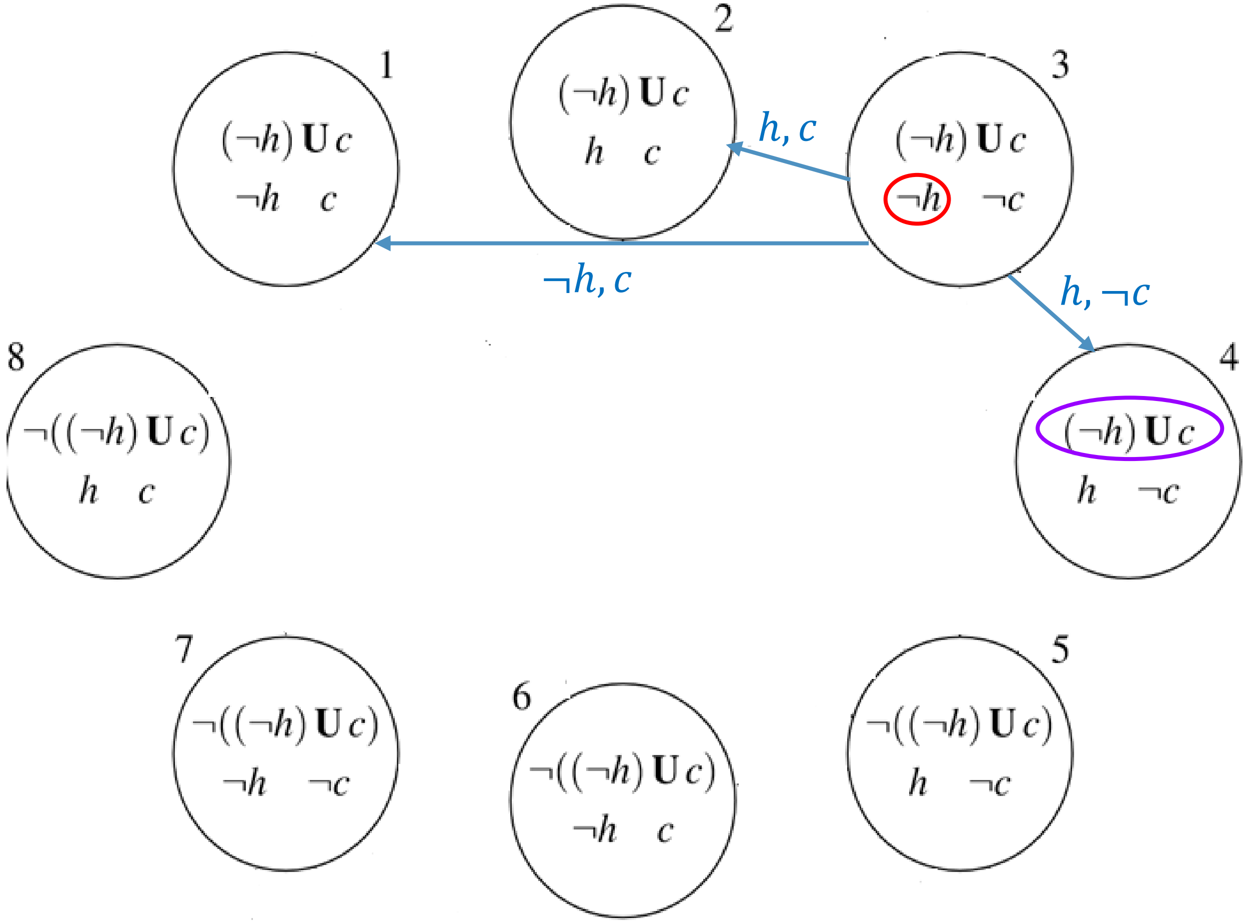


For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $X\varphi_1 \in cl(\varphi)$:
  - $X\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in$ q'

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi \coloneqq \neg h \text{ U } c$
- Draw the transitions of $\mathcal{A}_\varphi$

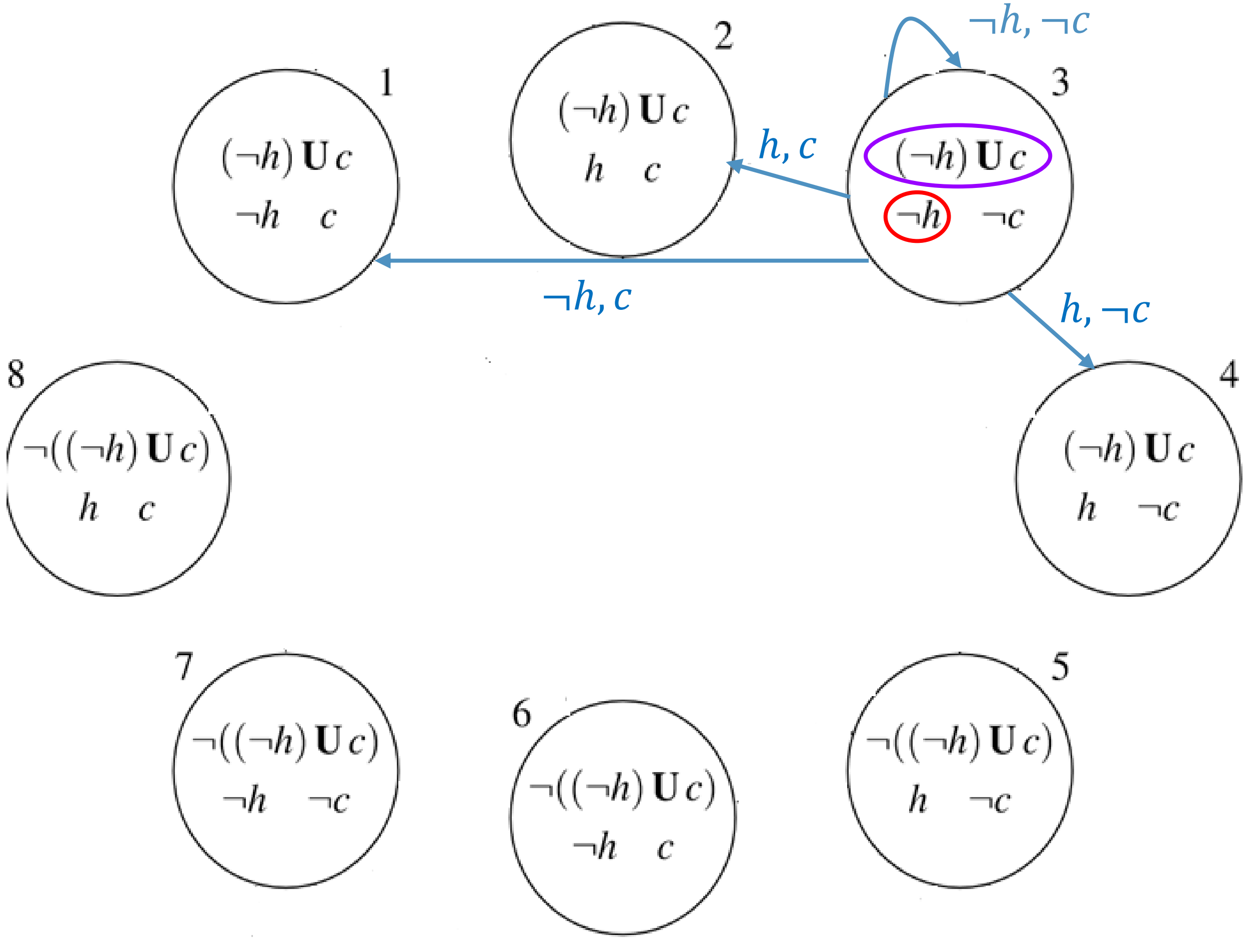


For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$:
  - $\boldsymbol{X}\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U\, \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q and φ_1 U $\varphi_2 \in$ q'

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi \coloneqq \neg h \text{ U } c$
- Draw the transitions of $\mathcal{A}_\varphi$



For $q, q' \in Q$ and $\sigma \subseteq AP$, $(q, \sigma, q') \in \Delta$ if:

- $\sigma = q' \cap AP$
- For all $X\varphi_1 \in cl(\varphi)$:
  - $X\varphi_1 \in q \Leftrightarrow \varphi_1 \in q'$
- For all $\varphi_1 U\, \varphi_2 \in cl(\varphi)$:
  - φ_1 U $\varphi_2 \in$ q $\Leftrightarrow$ either $\varphi_2 \in$ q or both $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in$ q'

LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

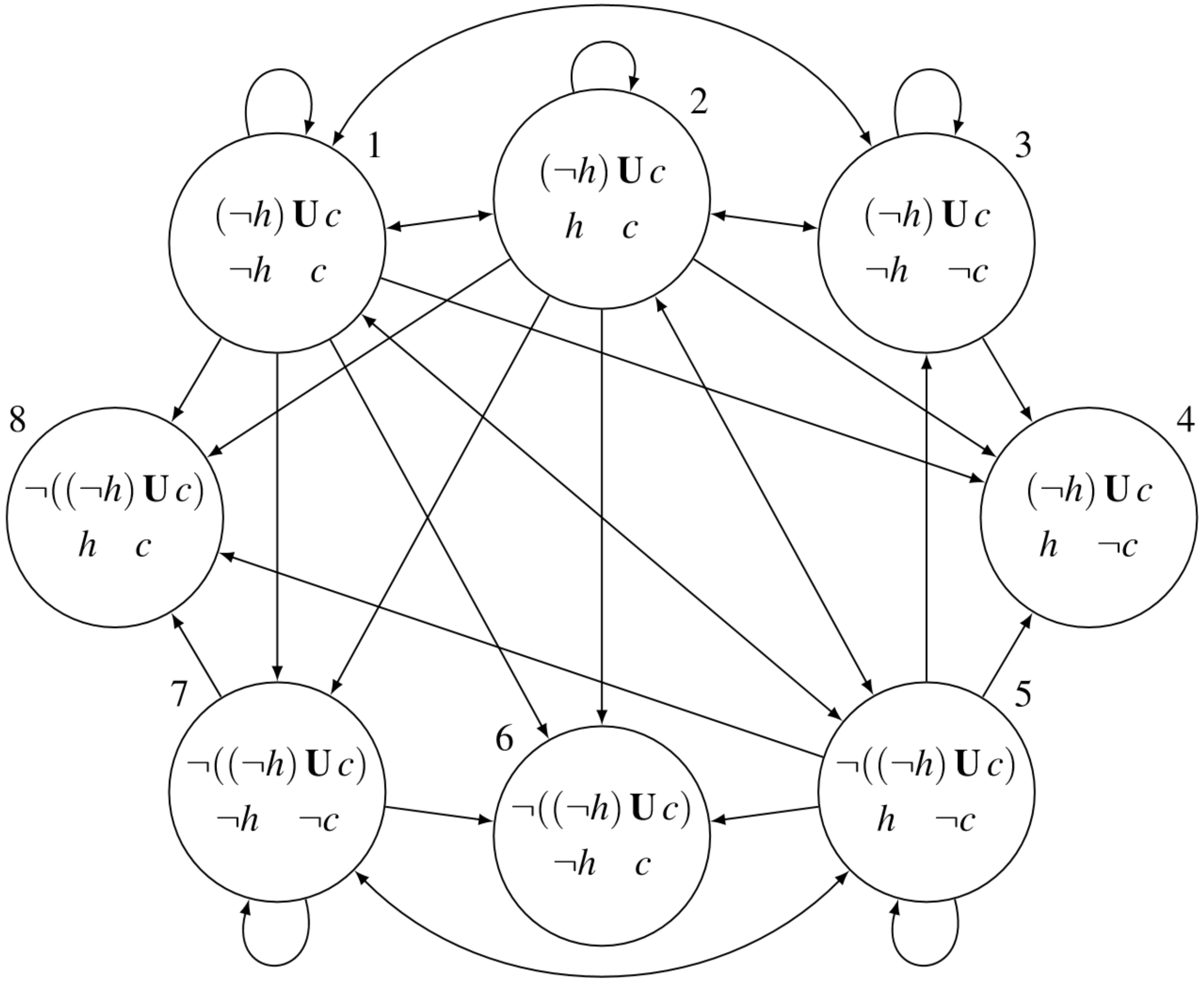
- $\mathcal{A}_\varphi = (\Sigma, \mathbf{Q}, \mathbf{\Delta}, \mathbf{Q}^0, \mathbf{F})$
- $\mathbf{Q}$ = set of all the good sets in *cl(φ)*
  - *Idea:* Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**

- For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:
  - $\sigma = q' \cap AP$
  - For all $\boldsymbol{X}\varphi_1$: if $\boldsymbol{X}\varphi_1 \in q$ then $\varphi_1 \in q'$
  - For all $\neg X\varphi_1$: if $\neg X\varphi_1 \in q$ then $\neg\varphi_1 \in q'$

  - For all $\varphi_1 U\, \varphi_2$: if $\varphi_1 \,\mathrm{U}\, \varphi_2 \in q$ then either $\varphi_2 \in q$ or **both** $\varphi_1 \in q$ **and** $\varphi_1 \,\mathrm{U}\, \varphi_2 \in q'$
  - For all $\neg(\varphi_1 U\, \varphi_2)$: if $\neg(\varphi_1 \,\mathrm{U}\, \varphi_2) \in q$ then either $\neg\varphi_2 \in q$ and **either** $\neg\varphi_1 \in q$ **or** $\neg(\varphi_1 \,\mathrm{U}\, \varphi_2) \in q'$

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- $\varphi \coloneqq \neg h \ \mathrm{U} \ c$
- Draw the transitions of $\mathcal{A}_\varphi$



LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

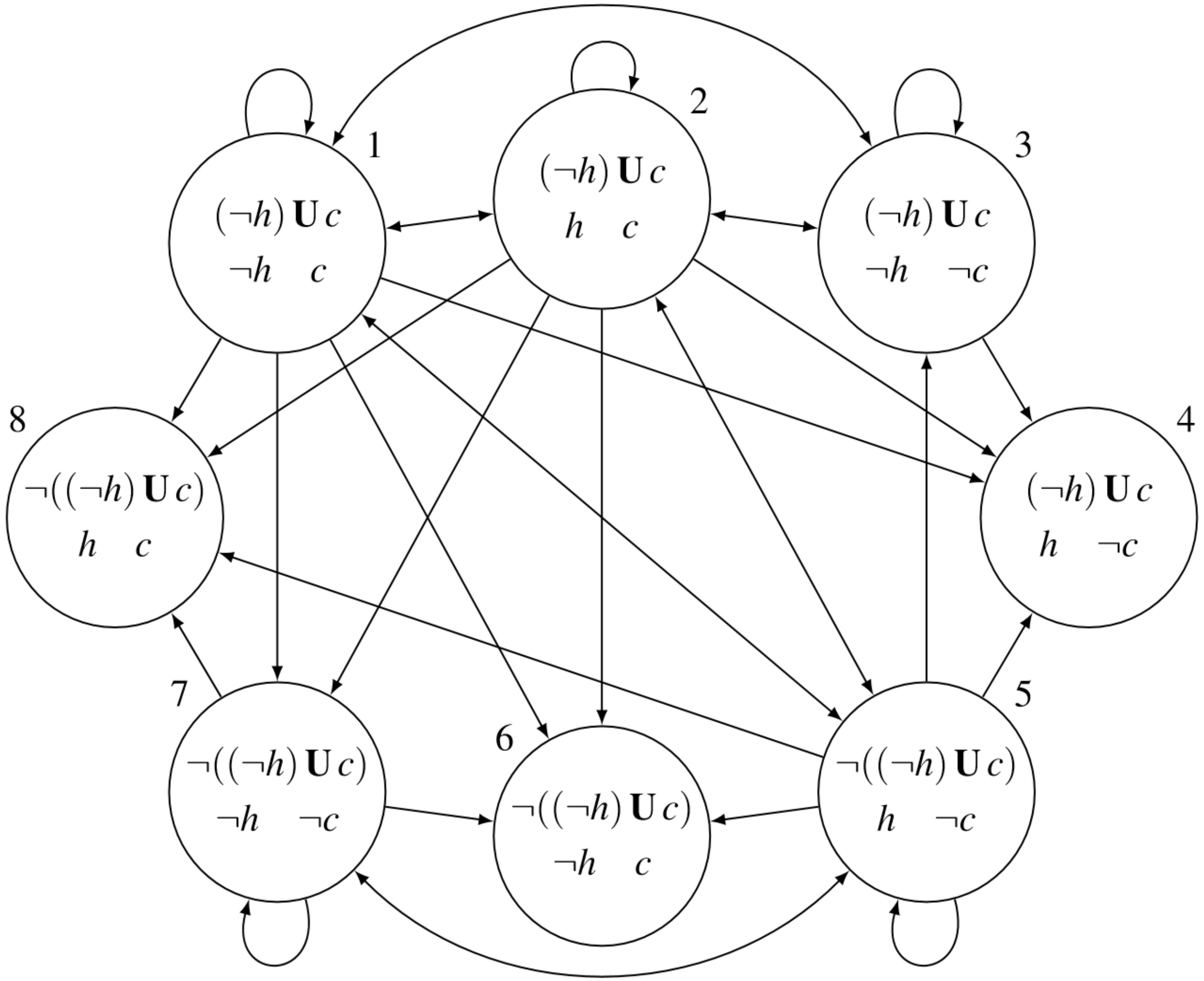
- $\mathcal{A}_\varphi = (\Sigma, \mathbf{Q}, \mathbf{\Delta}, \mathbf{Q}^0, \mathbf{F})$
- $\mathbf{Q}$ = set of all the good sets in *cl(φ)*
  - *Idea:* Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q.

- For $q, q' \in Q$ and $\sigma \subseteq$ AP, $(q, \sigma, q') \in \Delta$ if:
  - $\sigma = q' \cap AP$
  - For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$: if $\boldsymbol{X}\varphi_1 \in q$ then $\varphi_1 \in q'$
  - For all $\varphi_1 U \varphi_2 \in cl(\varphi)$: if φ_1 U $\varphi_2 \in$ q then either $\varphi_2 \in$ q or **both** $\varphi_1 \in$ q **and** φ_1 U $\varphi_2 \in q'$

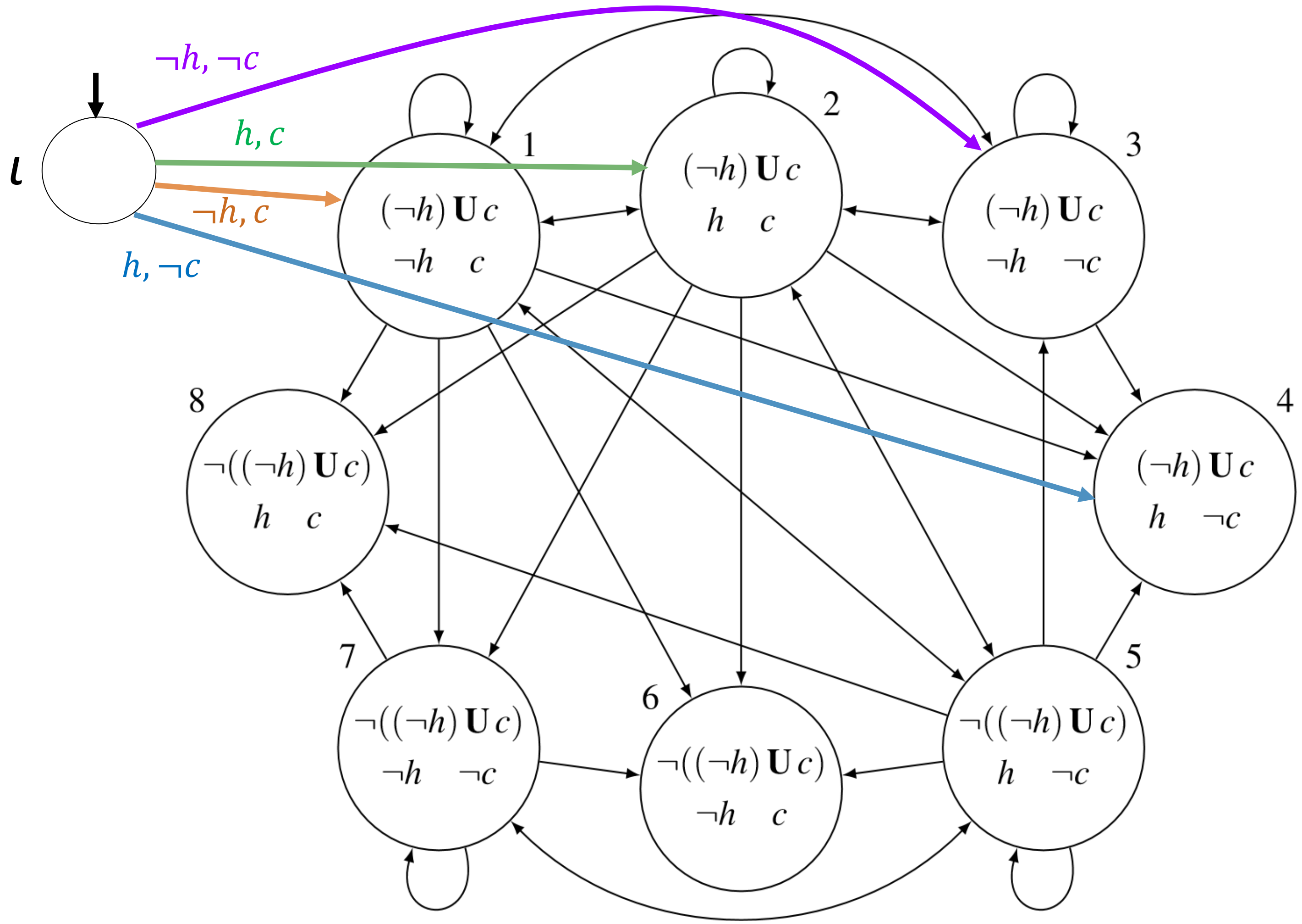
- Initial States?
- Accepting States?

Example: Transition Relation of GBA $\mathcal{A}_\varphi$

- Initial States?



Example: Transition Relation of GBA $\mathcal{A}_\varphi$



LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- $\mathcal{A}_\varphi = (\Sigma, \mathbf{Q}, \boldsymbol{\Delta}, \mathbf{Q}^0, \mathbf{F})$
- $\mathbf{Q}$ = set of all the good sets in $cl(\varphi) \cup \{\iota\}$
  - *Idea:* Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q.

- For $q, q' \in Q$ and $\sigma \subseteq AP$, $(q, \sigma, q') \in \Delta$ if:
  - $\sigma = q' \cap AP$
  - For all $\boldsymbol{X}\varphi_1 \in cl(\varphi)$: if $\boldsymbol{X}\varphi_1 \in q$ then $\varphi_1 \in q'$
  - For all $\varphi_1 U \varphi_2 \in cl(\varphi)$: if $\varphi_1 U \varphi_2 \in q$ then either $\varphi_2 \in q$ or **both** $\varphi_1 \in q$ **and** $\varphi_1 U \varphi_2 \in q'$
  - $(\boldsymbol{\iota}, \boldsymbol{\sigma}, \mathbf{q}) \in \boldsymbol{\Delta} \Leftrightarrow \boldsymbol{\varphi} \in \mathbf{q}$ **and** $\boldsymbol{\sigma} = \boldsymbol{q} \cap \boldsymbol{AP}$

- Accepting States?

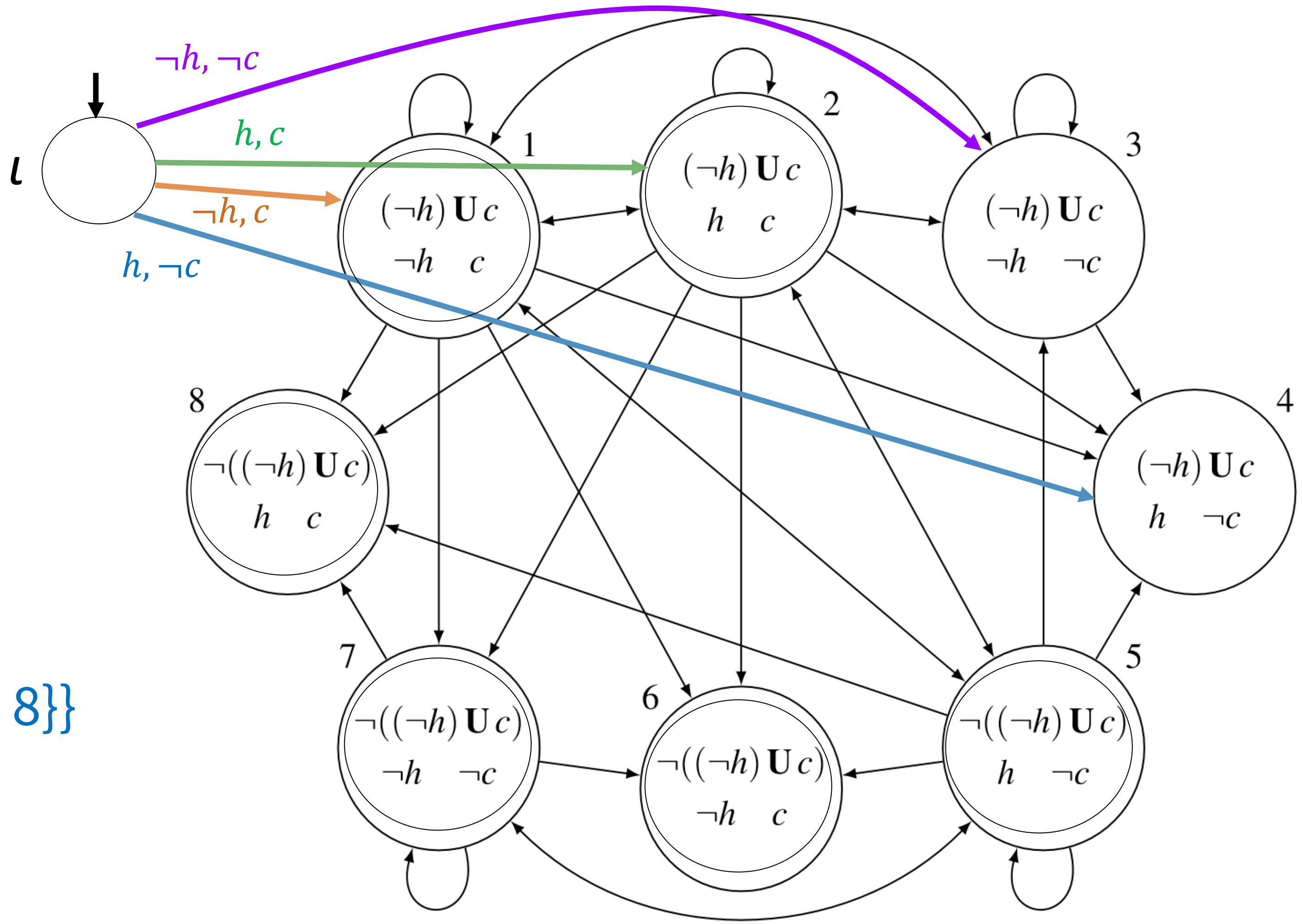
LTL formula φ to Generalized Büchi Automata $\mathcal{A}_\varphi$

- $\mathcal{A}_\varphi = (\Sigma, \mathbf{Q}, \boldsymbol{\Delta}, \iota, \mathbf{F})$
- $\mathbf{Q}$ = set of all the good sets in $cl(\varphi) \cup \{\iota\}$
  - *Idea:* Each state q is **labelled** with **a set of sub-formulas** that should be satisfied **on paths starting at** q**.**

- For $q, q' \in Q$ and $\sigma \subseteq AP$, $(q, \sigma, q') \in \Delta$ if:
  - $\sigma = q' \cap AP$
  - For all $X\varphi_1 \in cl(\varphi)$: if $X\varphi_1 \in q$ then $\varphi_1 \in q'$
  - For all $\varphi_1 U \varphi_2 \in cl(\varphi)$: if $\varphi_1 \cup \varphi_2 \in q$ then either $\varphi_2 \in q$ or **both** $\varphi_1 \in q$ **and** $\varphi_1 \cup \varphi_2 \in q'$
  - $(\iota, \sigma, q) \in \Delta \Leftrightarrow \varphi \in q$ **and** $\sigma = q \cap AP$

- Accepting States
  - For every $\varphi_1 \cup \varphi_2$, $\mathbf{F}$ includes the set $F_{\varphi_1 \cup \varphi_2} = \{q \in \mathbf{Q} \mid \varphi_2 \in q \text{ or } \neg(\varphi_1 \cup \varphi_2) \in q\}$.

Example: Transition Relation of GBA $\mathcal{A}_\varphi$



F = {{1, 2, 5, 6, 7, 8}}

Plan for Today

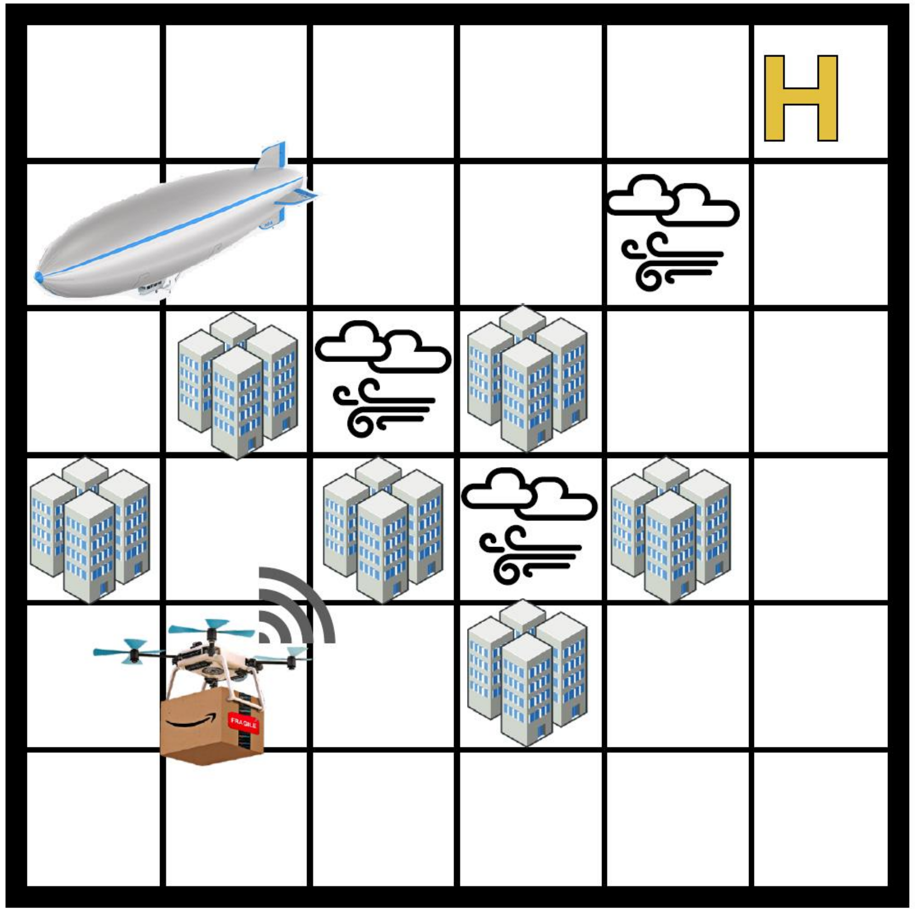
- Presentation of Homework
- Part 1 - LTL Model Checking
  - Generalized Büchi Automata
  - Translation of LTL to Büchi Automata
- Part 2 – Shielded Reinforcement Learning

Outline

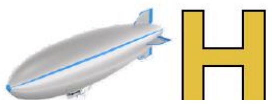
- **Shielding for Safety**
  - Integration of a shield in RL
  - Symbolic Models
  - Shields with Absolute Safety Guarantees
  - Shields with Probabilistic Guarantees

Reinforcement Learning

- Decision Making under Uncertainty
- Environment modeled as Markov Decision Process



Uncertainty caused by sensor imprecision, wind gusts, and limited view

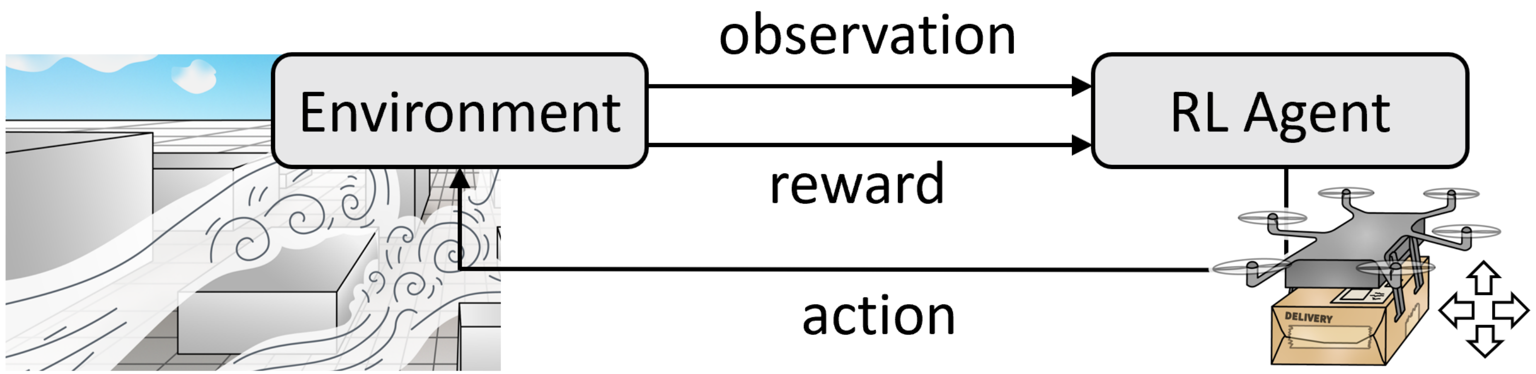


Complex **task specification**

Setting from Nils Jansen

Reinforcement Learning

- RL agent learns optimal policy via trial and error



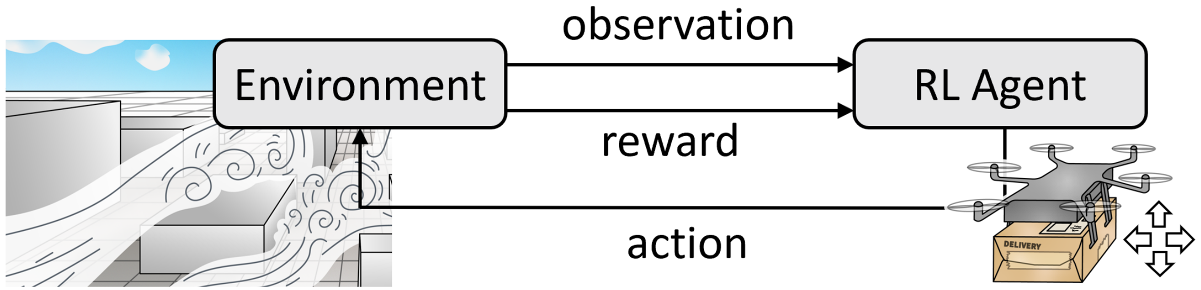
Find a policy π^* that maximixes $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$

with the discount factor $0 \leq \gamma \leq 1$ and reward R_t at time t

Reinforcement Learning

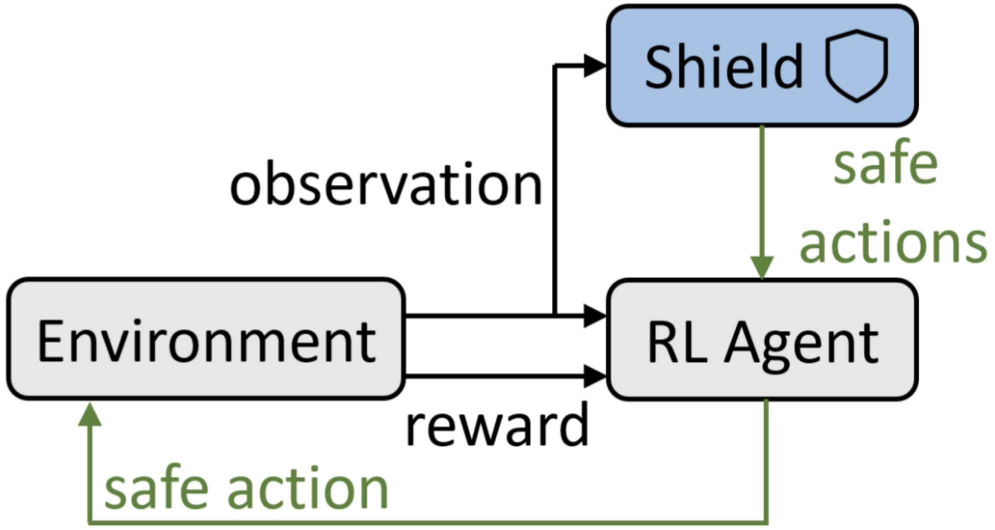
Limitations

- **Safety violations** (during exploration)
- RL is data-hungry
- Rewards cannot capture sophisticated task specifications



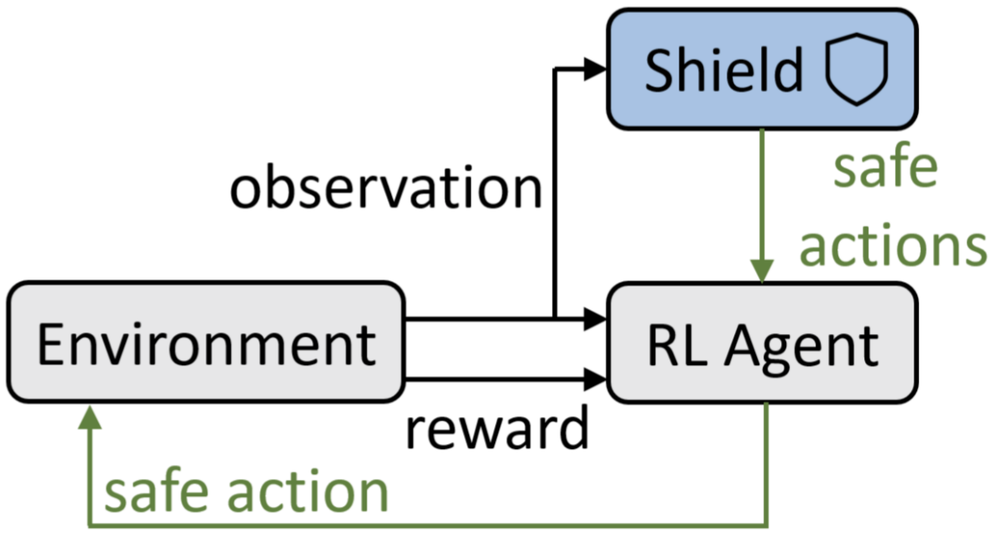
Integration of a Shield in RL

Pre-Shielding

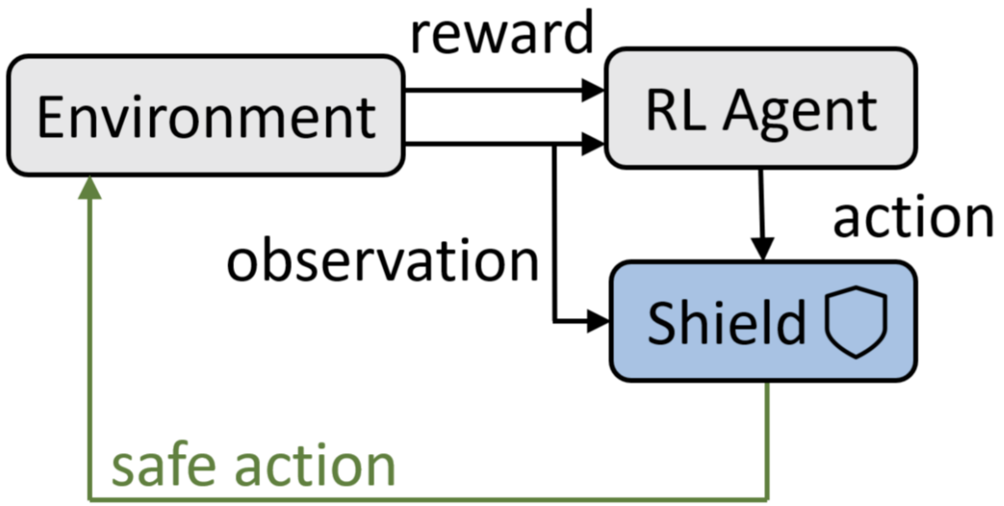


Integration of a Shield in RL

Pre-Shielding



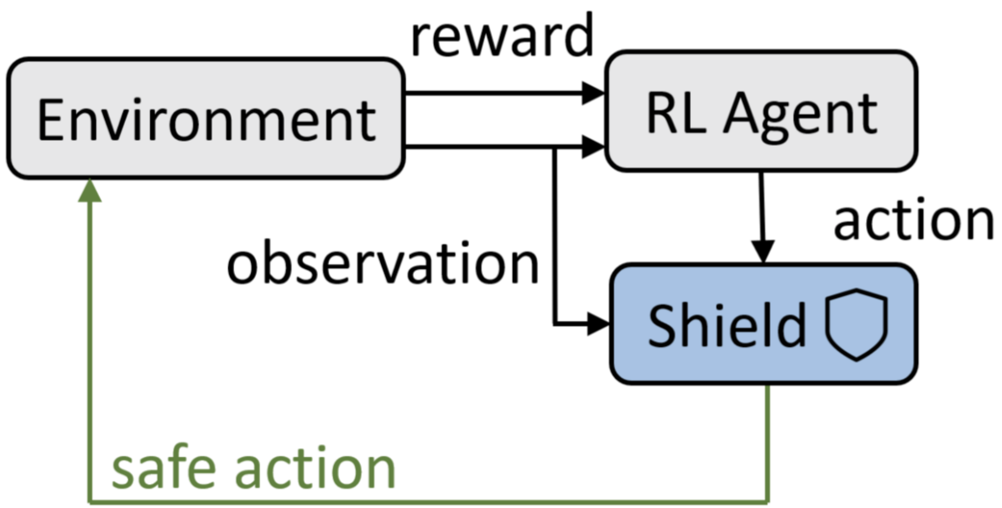
Post-Shielding



Pros/Cons of Post-Shielding

- **Advantages**
  - **Safety** during training/deployment

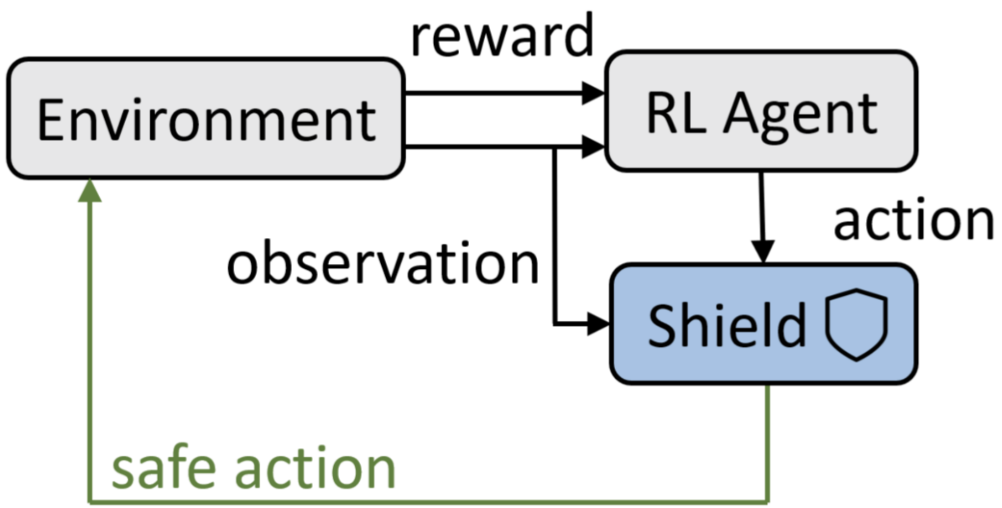
Post-Shielding

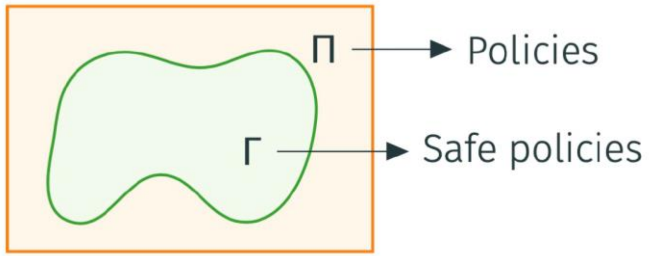


Pros/Cons of Post-Shielding

- **Advantages**
  - **Safety** during training/deployment
  - Can improve the **learning performance** of RL
  - A **shield** injects **domain knowledge** to reduce search space

Post-Shielding

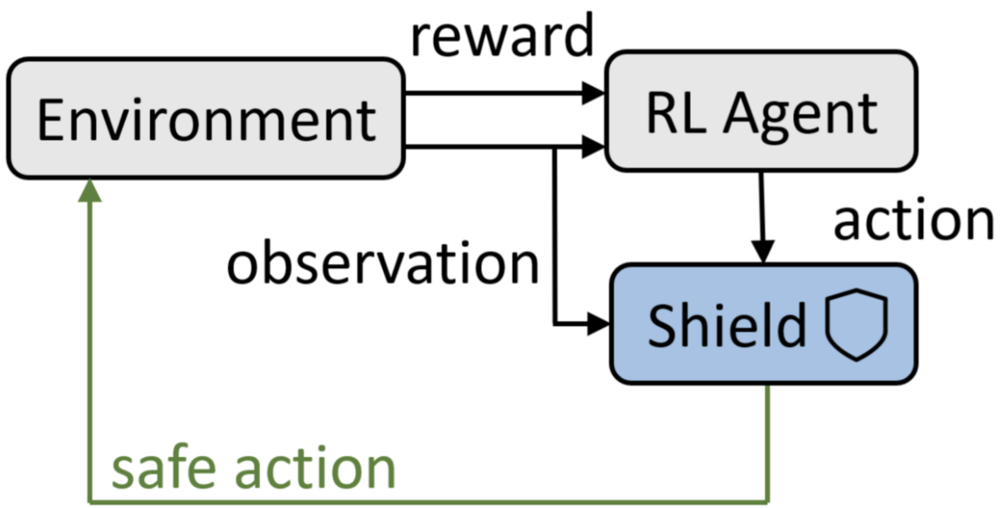




Pros/Cons of Post-Shielding

- **Disadvantages**
  - **Shielding Assumptions**
    - Symbolic **model is correct** and captures everything safety critical
    - **Observations** are correct

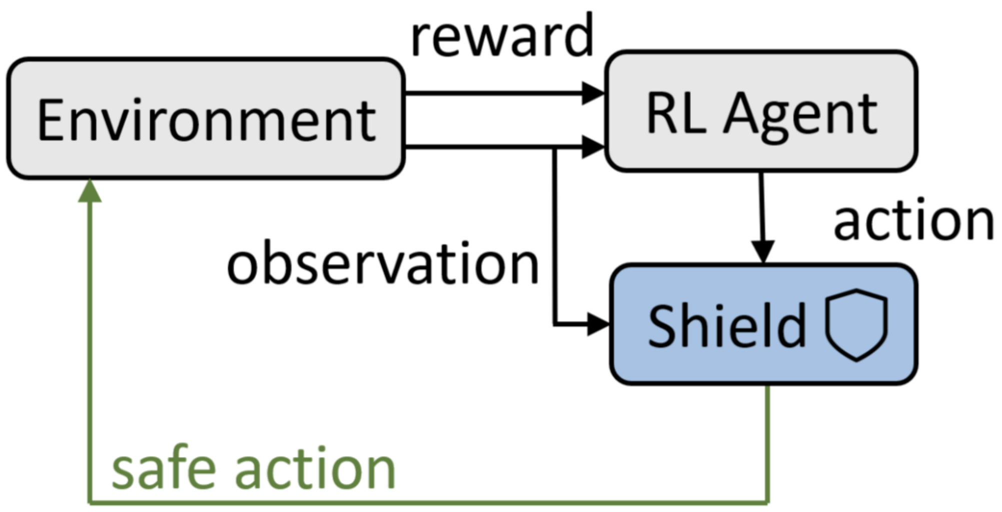
Post-Shielding



Pros/Cons of Post-Shielding

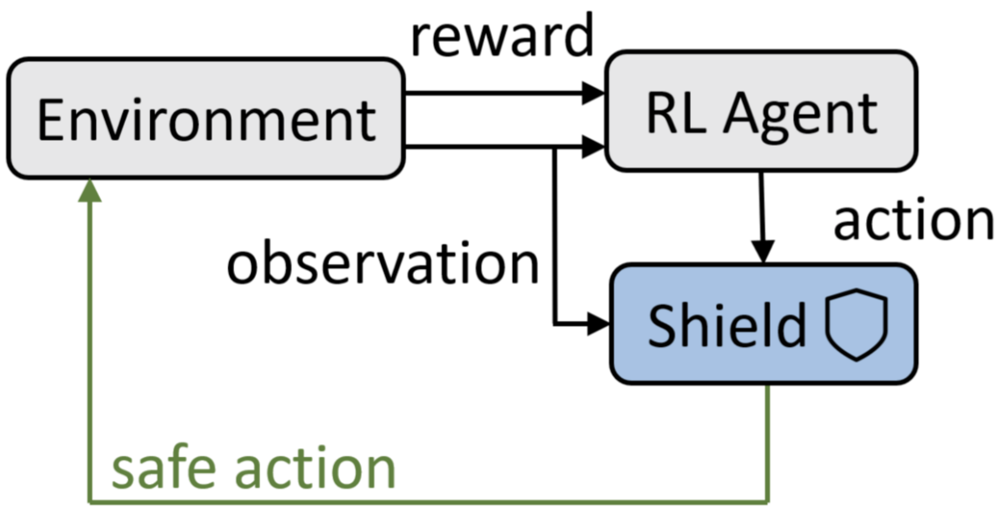
- **Disadvantages**
  - **Shielding Assumptions**
    - Symbolic **model is correct** and captures everything safety critical
    - **Observations** are correct
  - Naive integration can destroy association between executed **action and reward**.

Post-Shielding



Pros/Cons of Post-Shielding

Post-Shielding

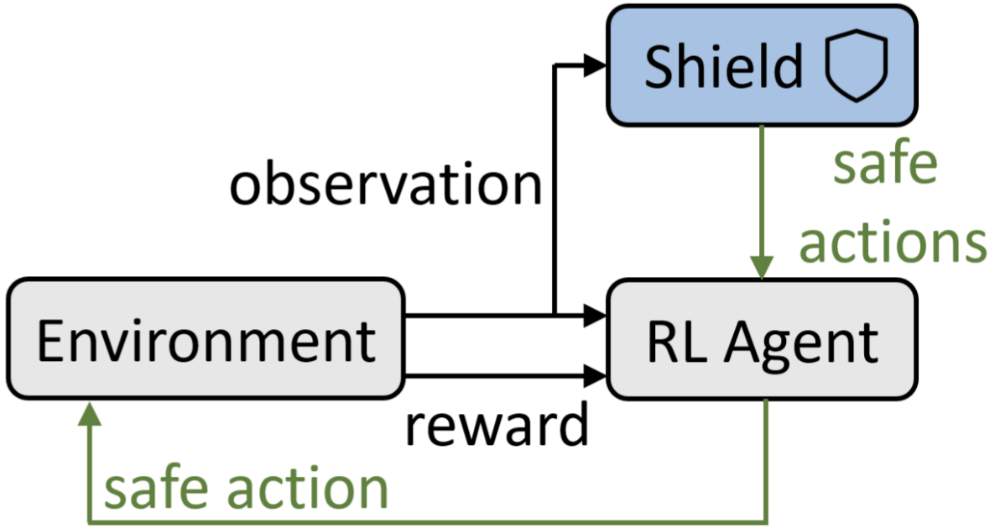


- **Disadvantages**
  - **Shielding Assumptions**
    - Symbolic **model is correct** and captures everything safety critical
    - **Observations** are correct
  - Naive integration can destroy association between executed **action and reward**.
  - Shield may hinder agent to **explore environment.**

Pros/Cons of Pre-Shielding

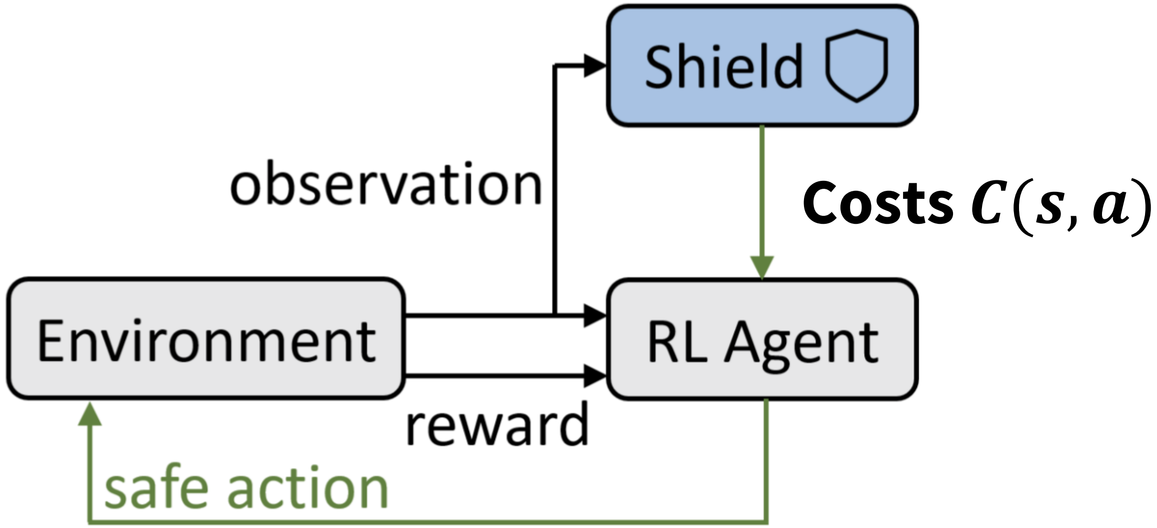
- **Advantages**
  - **Easy integration** for **maskable RL algorithms**
  - Final decision about which **action to explore** remains with RL agent
- **Disadvantages**
  - Integration difficult for non-maskable RL algorithms
- Others as before

Pre-Shielding



Shield Integration via Constrainted RL

- Agent should learn to behave safely



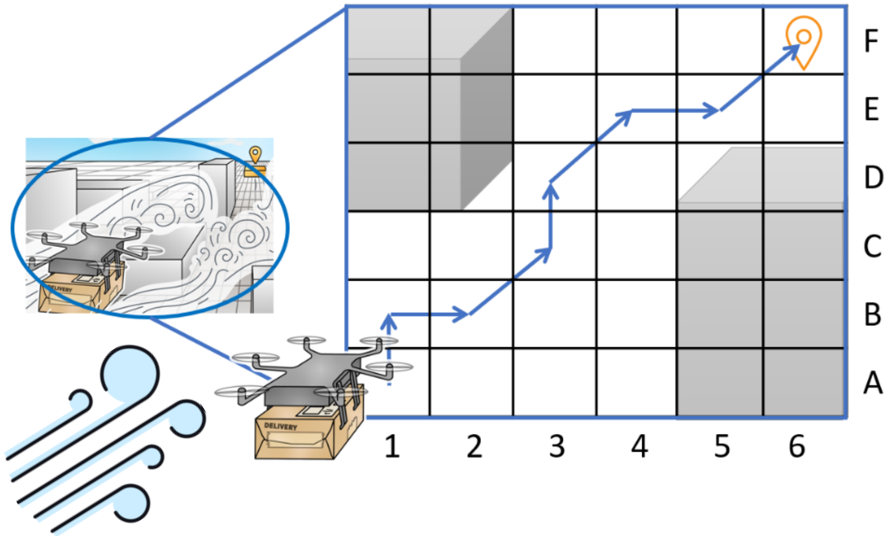
Find policy $\max_{\theta} J_R^{\pi_\theta} = \mathbb{E}_{\tau \sim \pi_\theta} [\sum_{t=0}^{\infty} \gamma^t R_t(s_t, a_t, s_{t+1})] \quad s.t. \quad J_C^{\pi_\theta} \leq \epsilon.$

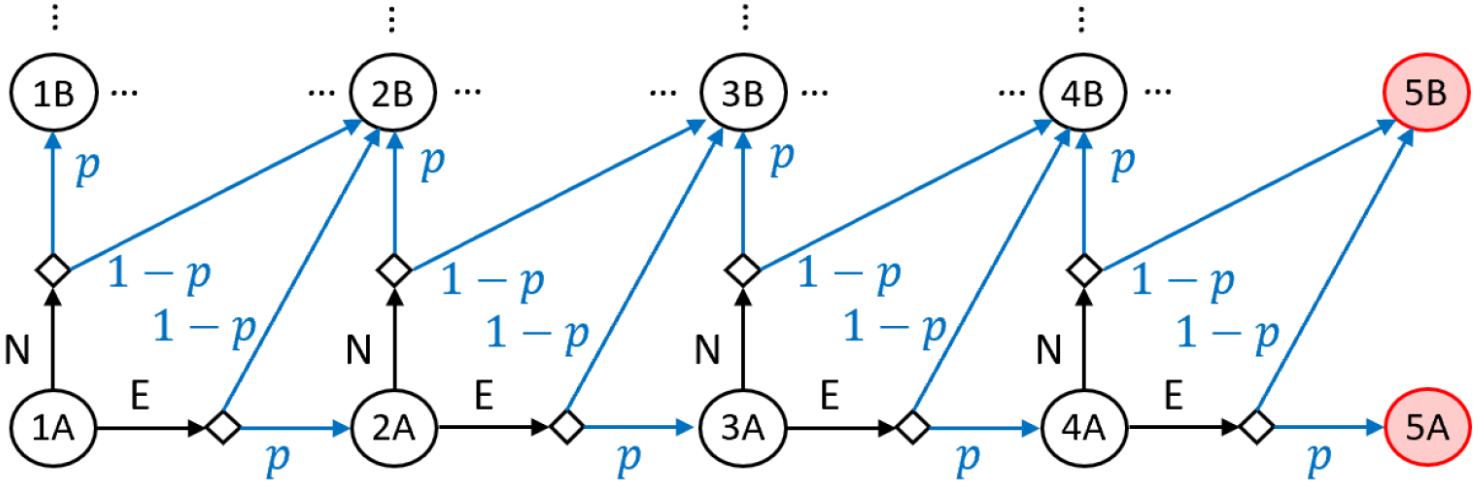
Outline

- **Shielding for Safety**
  - Integration of a shield in RL
  - Symbolic Models
  - Shields with Absolute Safety Guarantees
  - Shields with Probabilistic Guarantees

Symbolic World Model

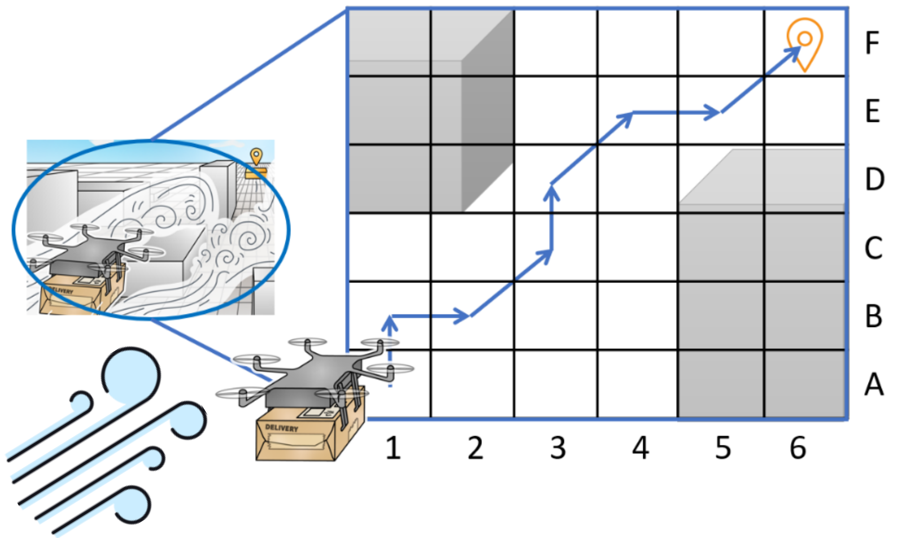
- Assumption: Environment has finite number of states, time is discrete
- → model as Markov Decision Process M

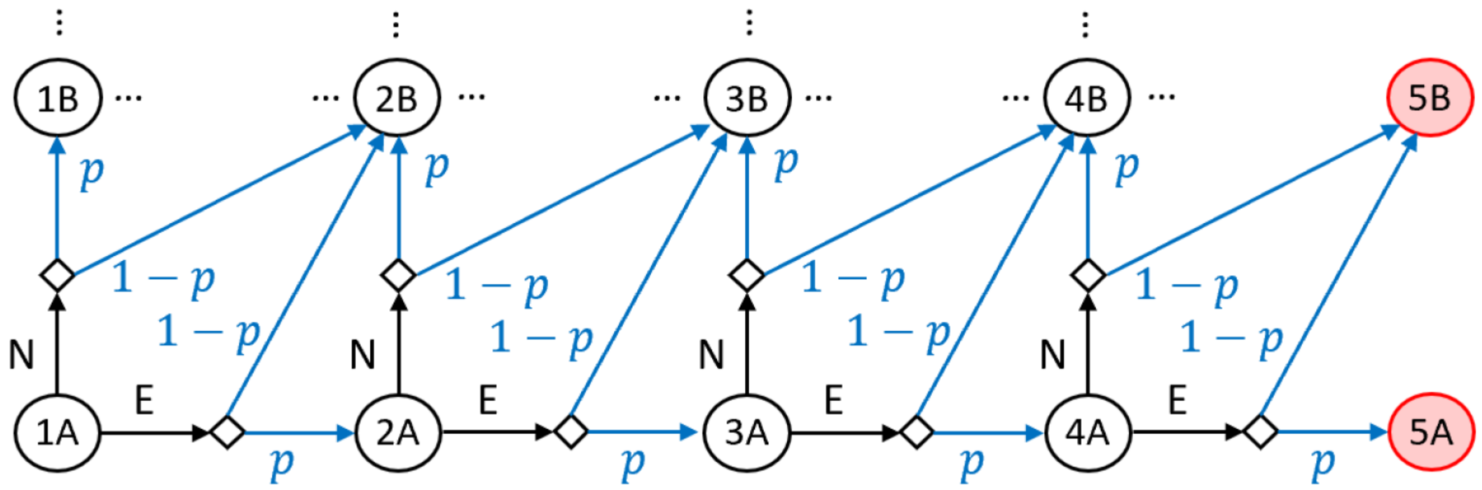




Symbolic World Model

- Assumption: Environment has finite number of states, time is discrete
- → model as Markov Decision Process M
- φ is a safety specification in temporal logic
  - Defines unsafe states in M
- **Shield prevents/limits probability of reaching an unsafe state in M**





Scalability of Shielded Learning

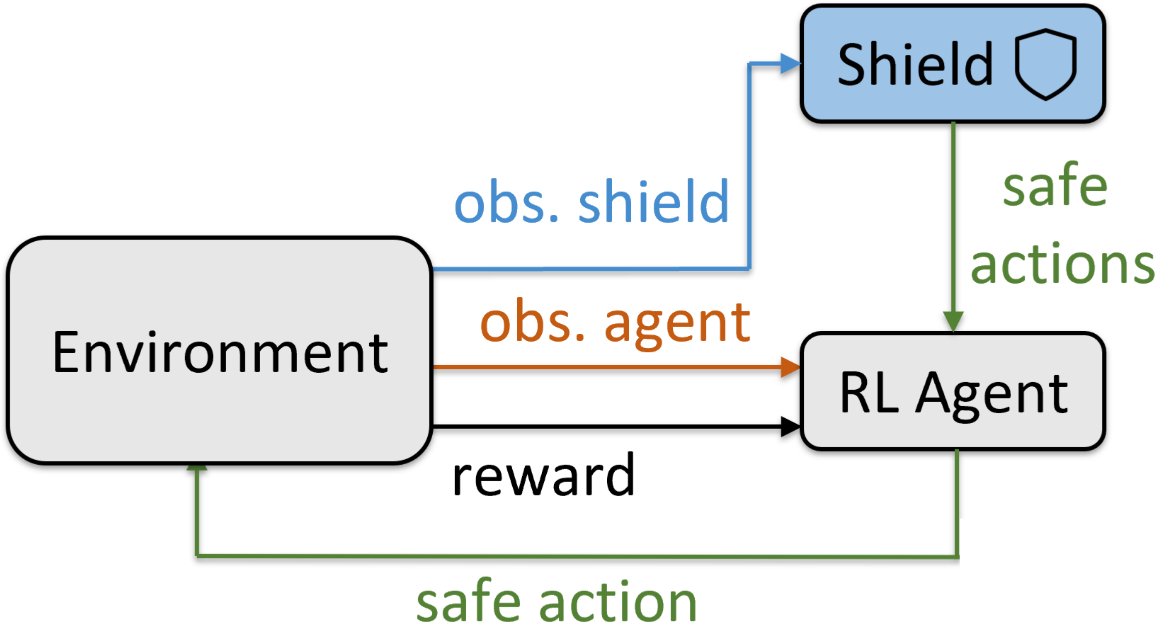
- Shielding is less scalable as RL
  - Shielding can handle MDPs with Millions of states

- Shield computed on safety-relevant MDP
  - RL works on original MDP
  - Shield works with MDP with reduced feature space

Scalability of Shielded Learning

- Shielding is less scalable as RL
  - Shielding can handle MDPs with Millions of states

- Shield computed on safety-relevant MDP
  - RL works on original MDP
  - Shield works with MDP with reduced feature space

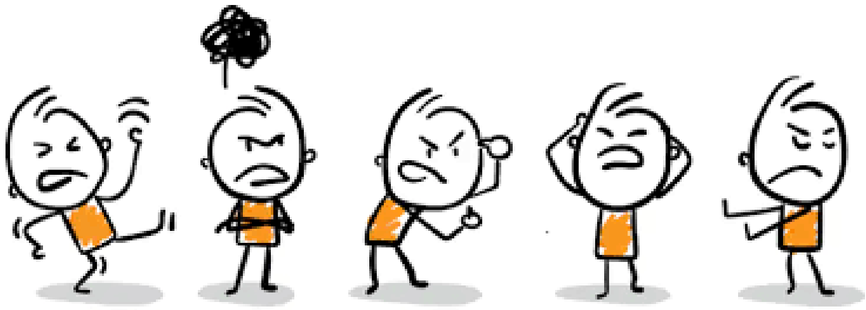


$\langle x, y, z, v \rangle$

$\langle x, y, z, v, \text{goal position}, \text{temperatur} \ldots \rangle$

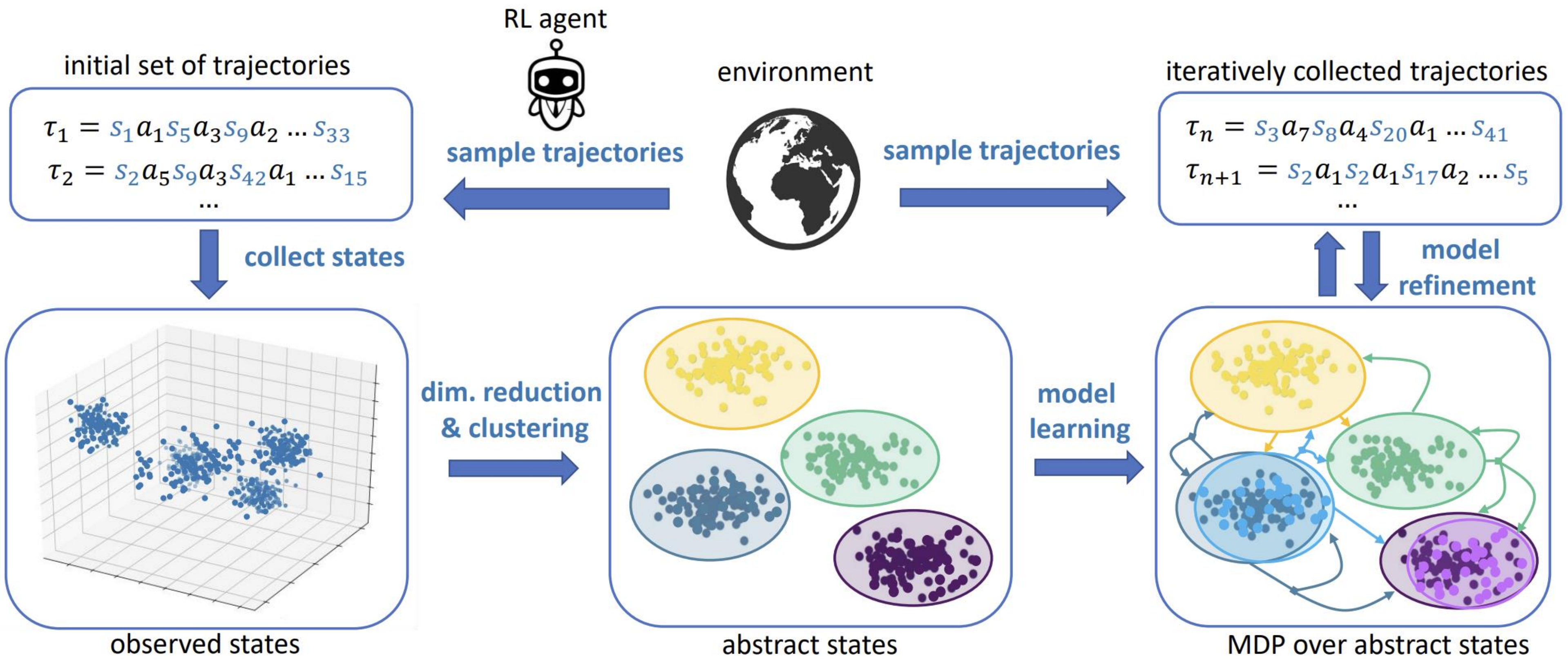
How to get the Model?

- Most of the time, no models are available!



How to get the Model?

- Most of the time, no models are available!
- → Use **automata learning** to learn world model



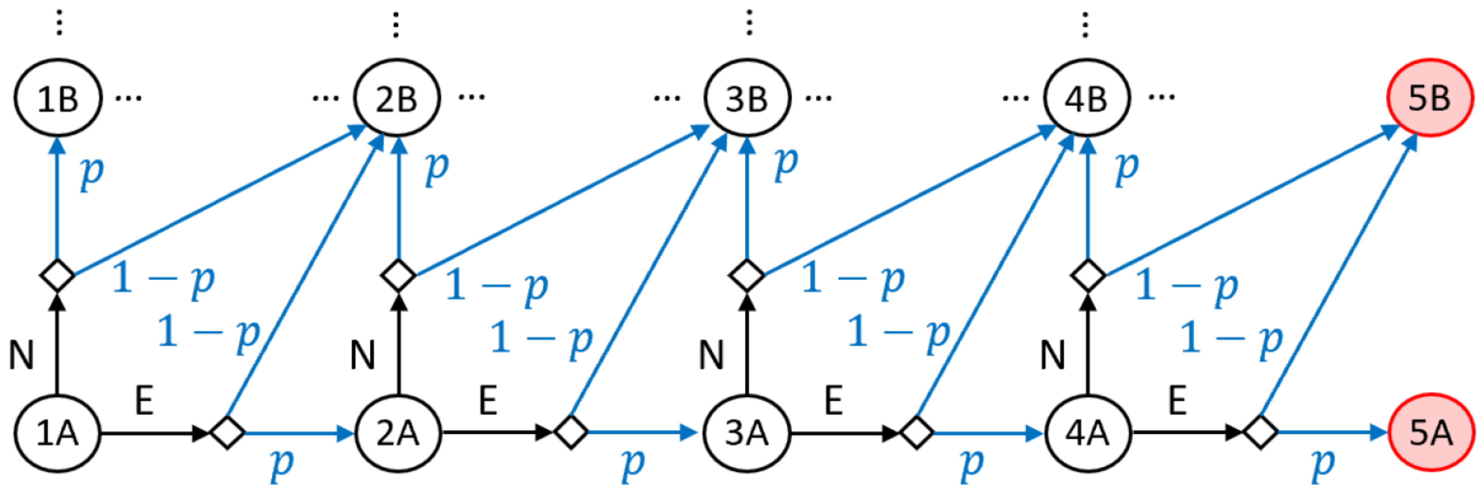
📖 M. Tappler, E. Muskardin, B. Aichernig, B. Könighofer:
Learning Environment Models with Continuous Stochastic Dynamics. ICST 2024

Outline

- **Shielding for Safety**
  - Integration of a shield in RL
  - Symbolic Models
  - Shields with Absolute Safety Guarantees
  - Shields with Probabilistic Guarantees

Shield with Absolute Safety Guarantees

- *Given*: MDP M, safety spec φ defines set of unsafe states in M
- Shield provides **absolute safety guarantees**
  - **Unsafe states are never visited!**
  - **In LTL: $G(safe)$**

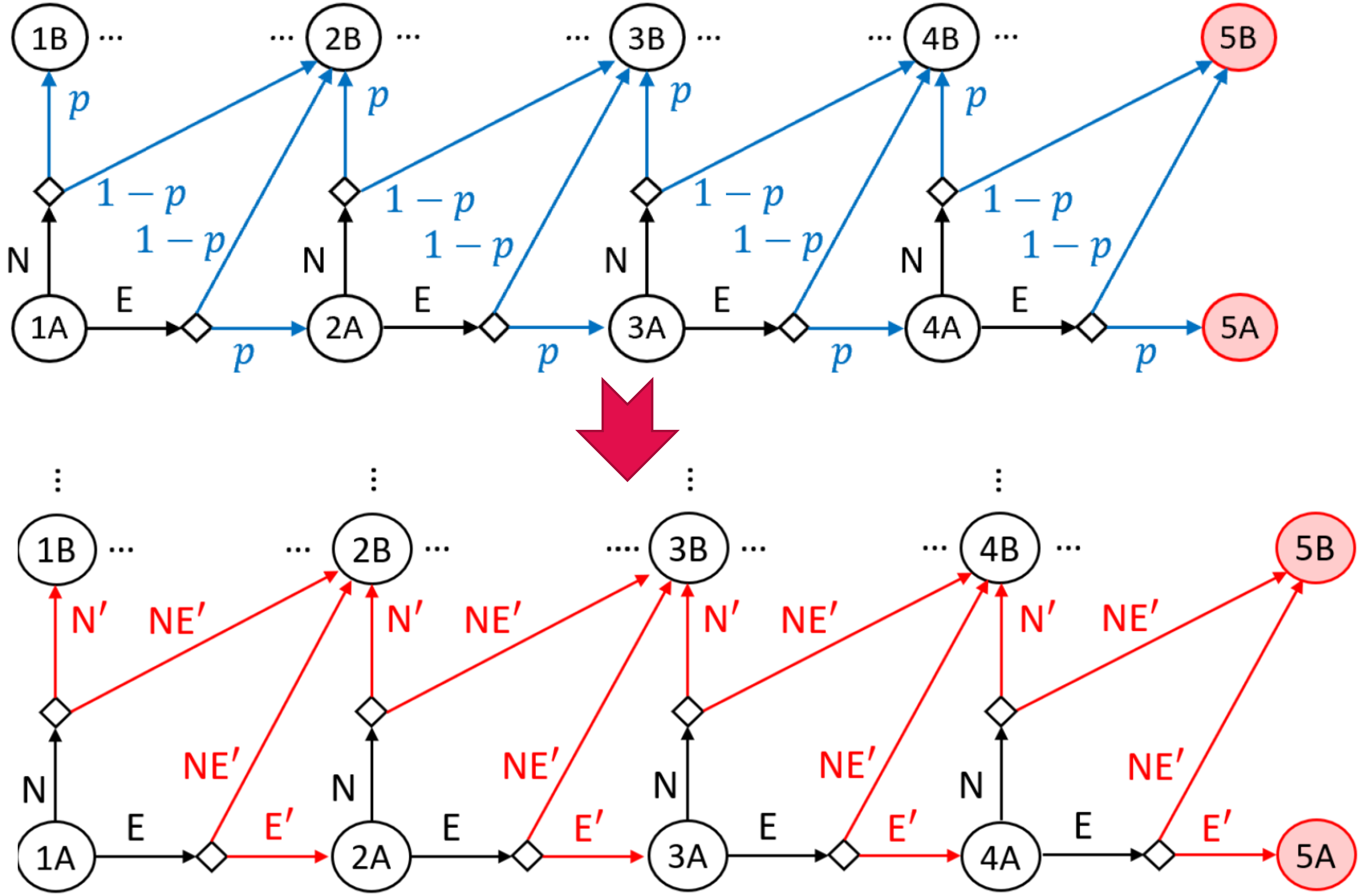


Shields with Absolute Safety Guarantees

- Shield Computation: Transform MDP to 2-Player Game
  - Replace probabilities by choices of environment

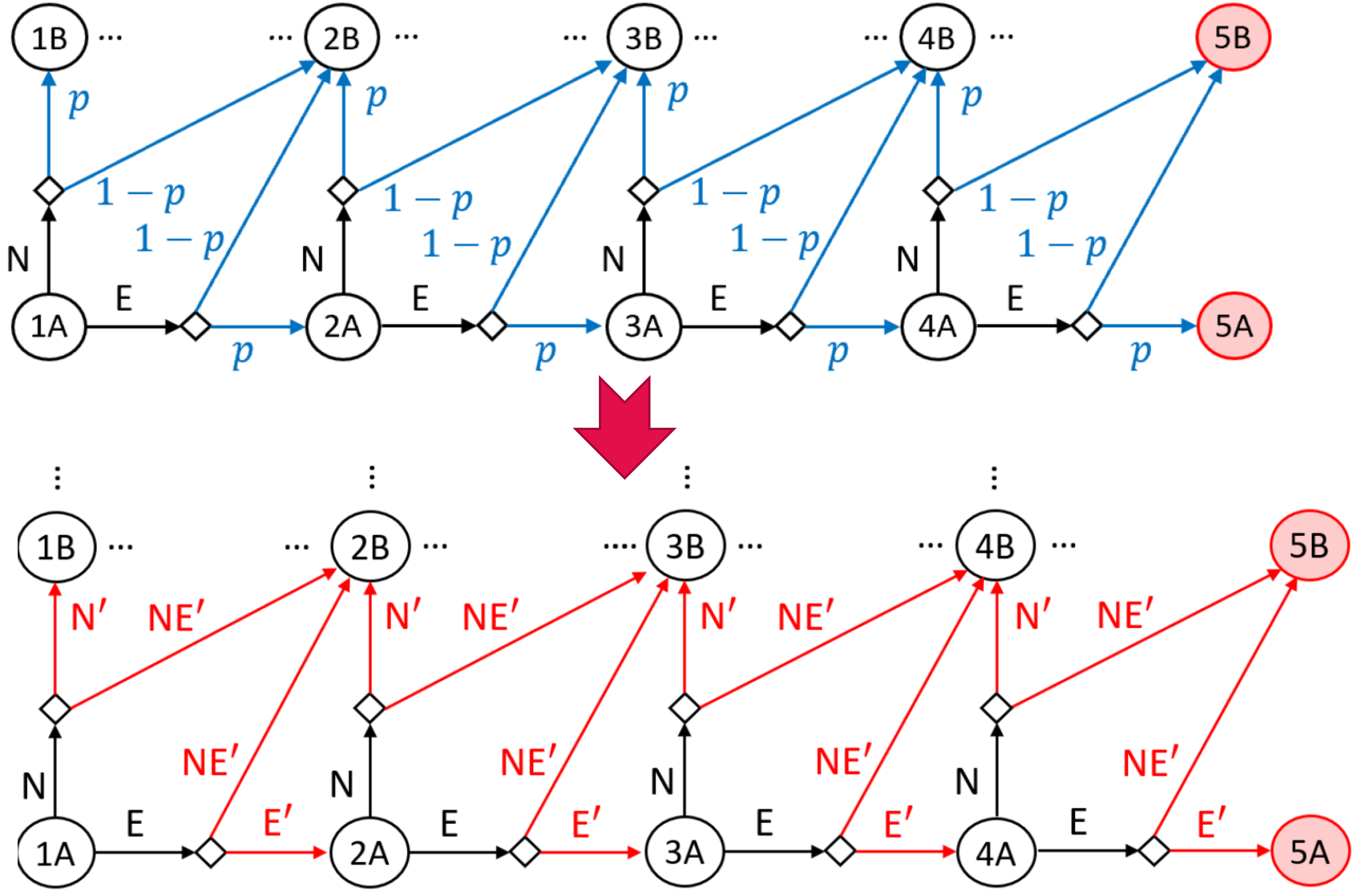
Shields with Absolute Safety Guarantees

- Shield Computation: Transform MDP to 2-Player Game
  - Replace probabilities by choices of environment



Shields with Absolute Safety Guarantees

- Shield Computation: Transform MDP to 2-Player Game
  - Replace probabilities by choices of environment



MDP = 1 ½ Player
→ Player 1: RL Agent
→ Probabilistic ½ Player

2 Player Game
→ Player 1: RL Agent
→ Plyer 2: Environment

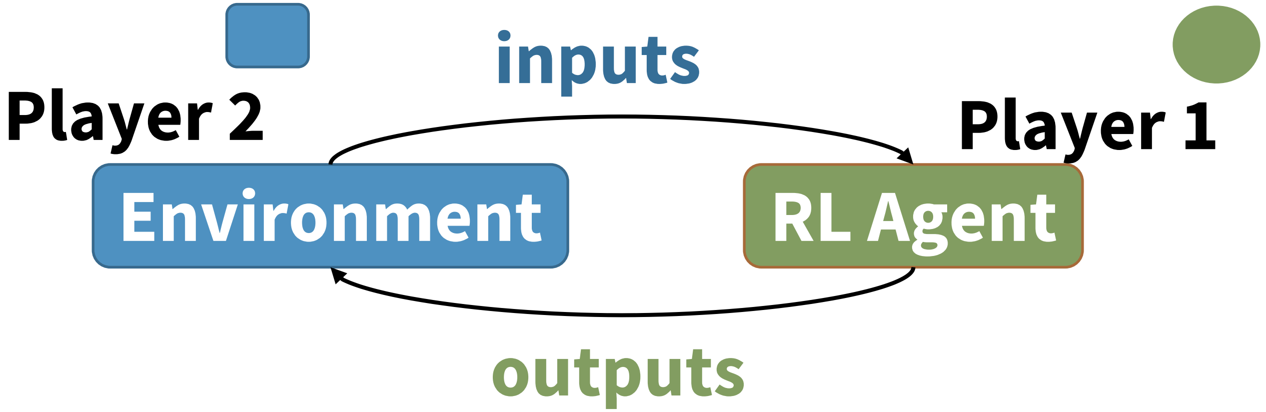
Shields with Absolute Safety Guarantees

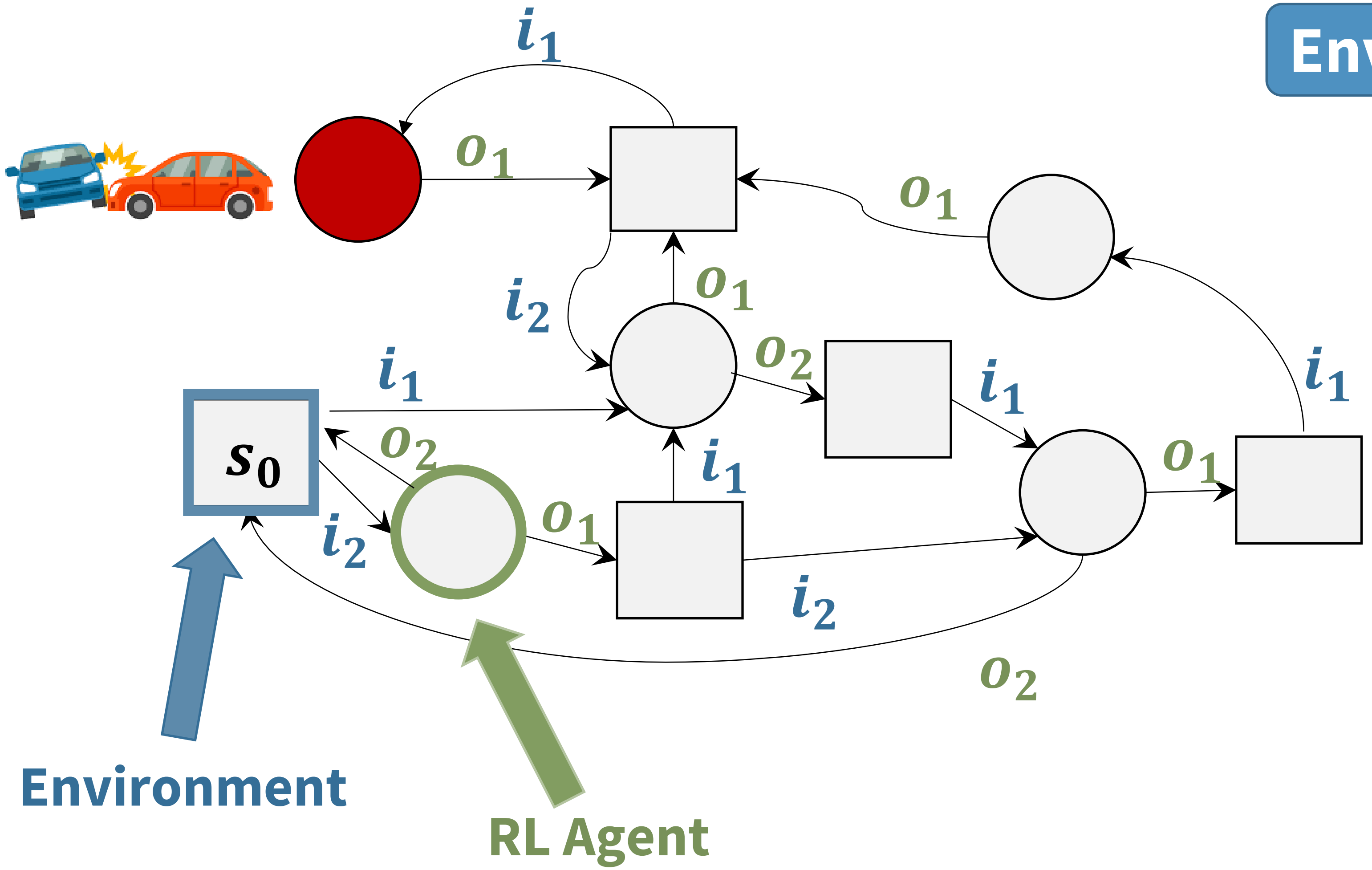
- Shield computation = Solve Safety Game
  - Agent: **Good player:** wins if only safe states are visited
  - Environment: **Evil player**: wins if an unsafe state is visited

Shields with Absolute Safety Guarantees

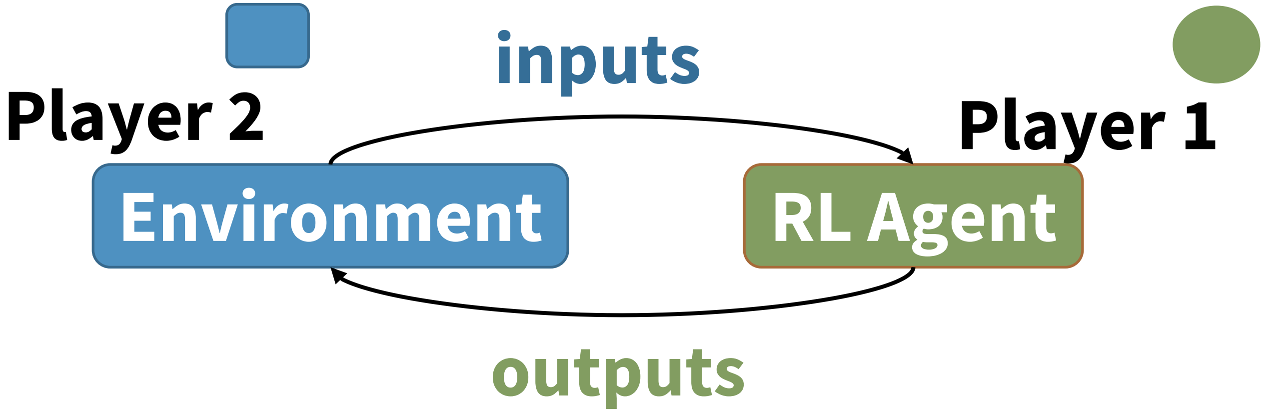
- Shield computation = Solve Safety Game
  - Agent: **Good player:** wins if only safe states are visited
  - Environment: **Evil player**: wins if an unsafe state is visited
  - Solve safety game
    - Fixpoint computation
    - Linear time in size of graph

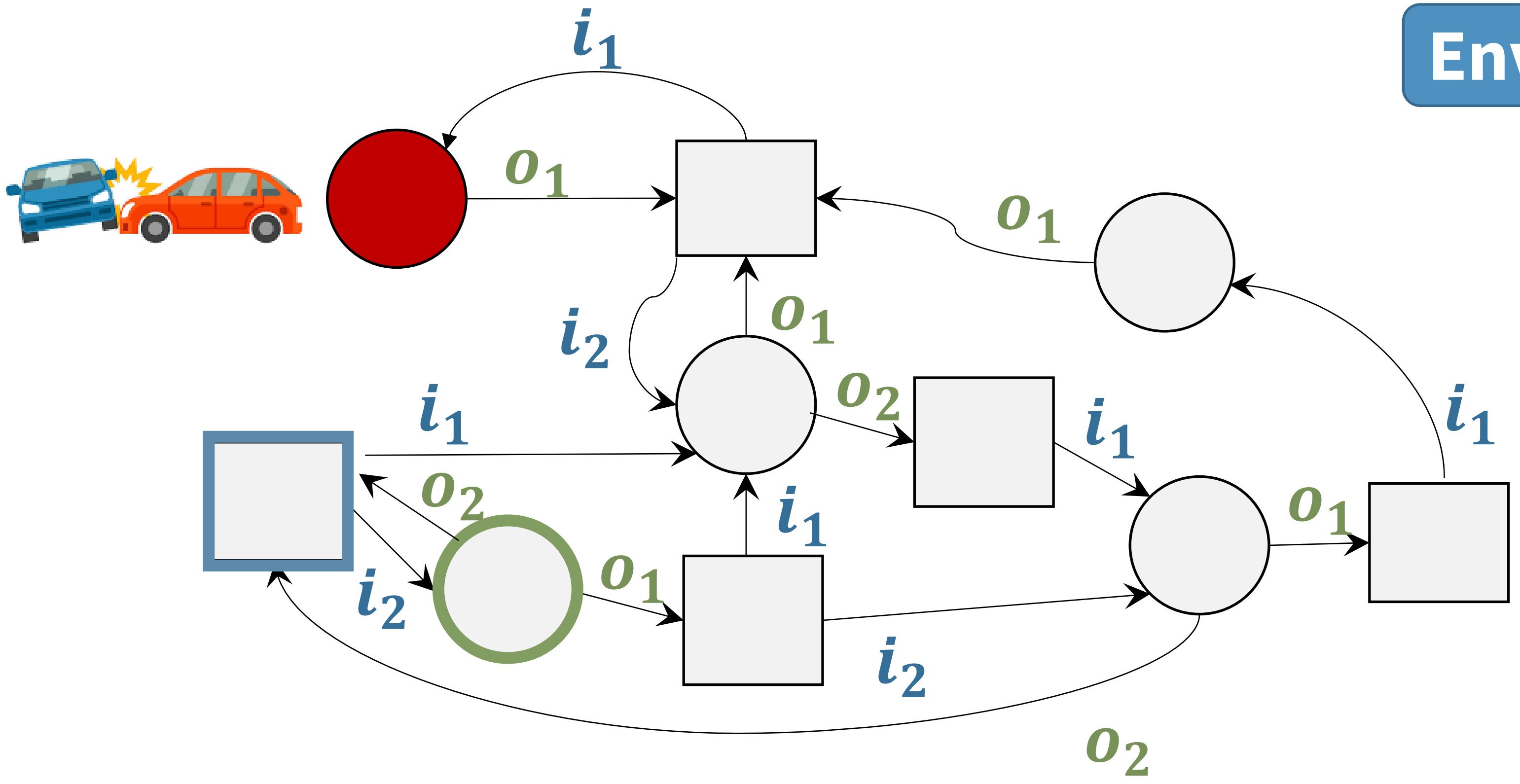
Example: Shield Construction = Solve Safety Game





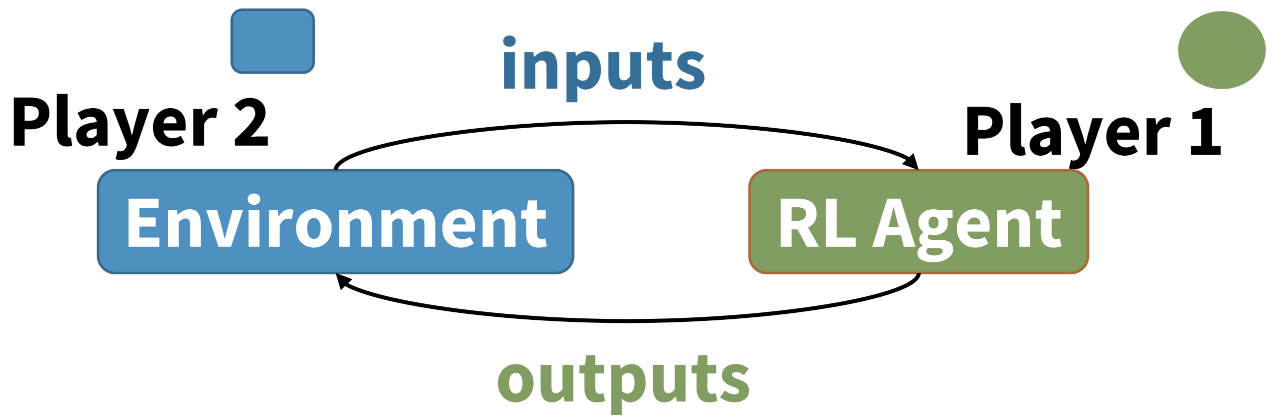
Example: Shield Construction = Solve Safety Game

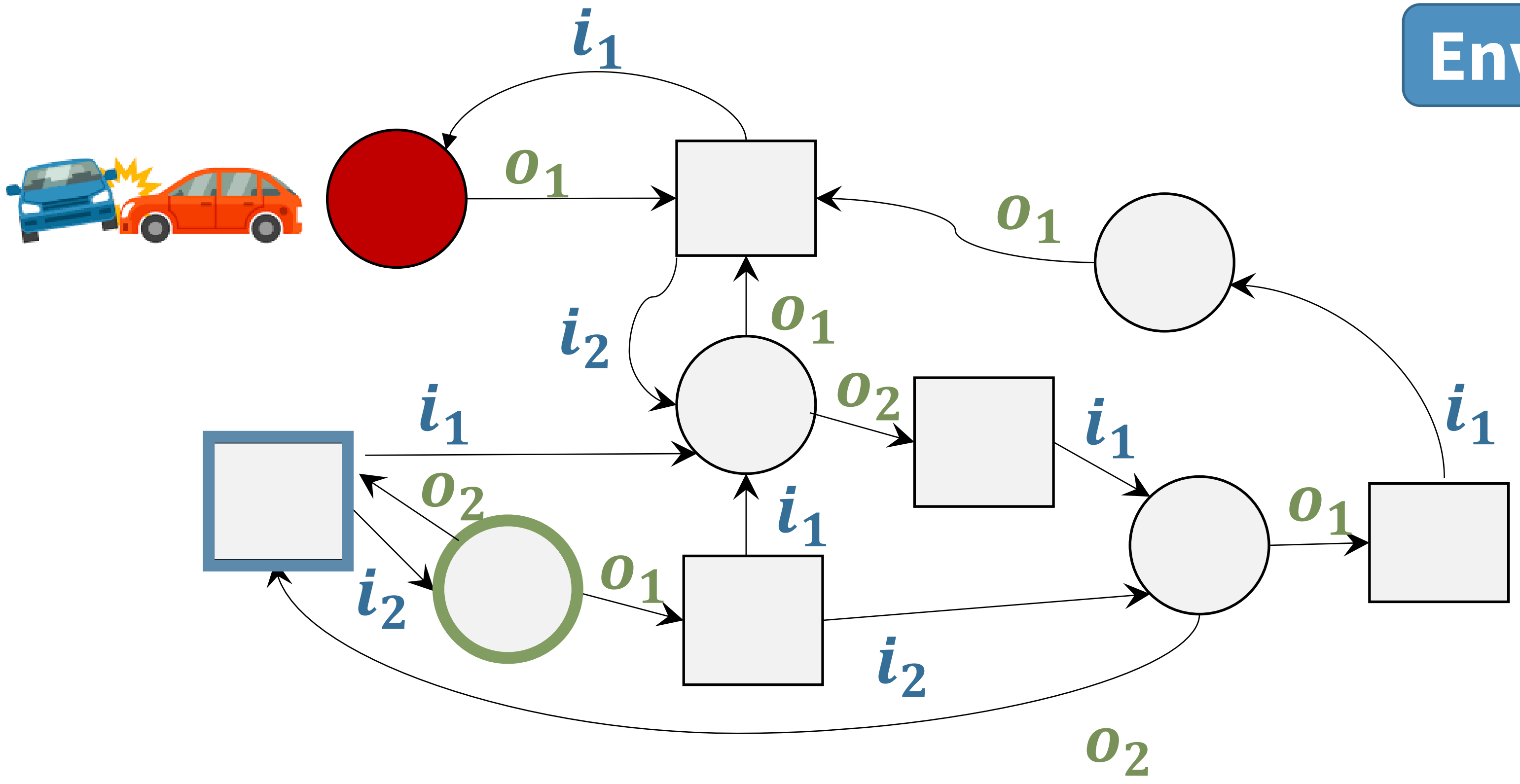




Player 1 wins,
if is **never** visited

Example: Shield Construction = Solve Safety Game

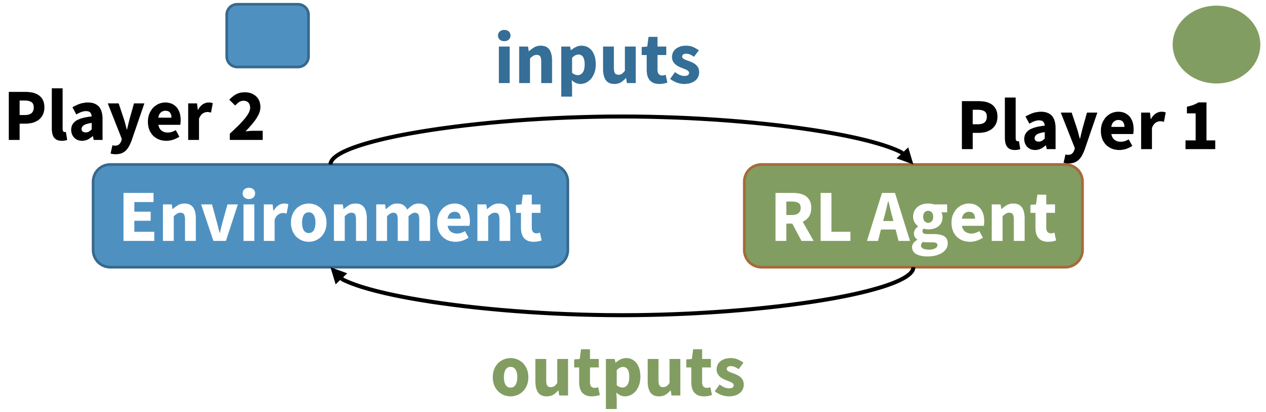


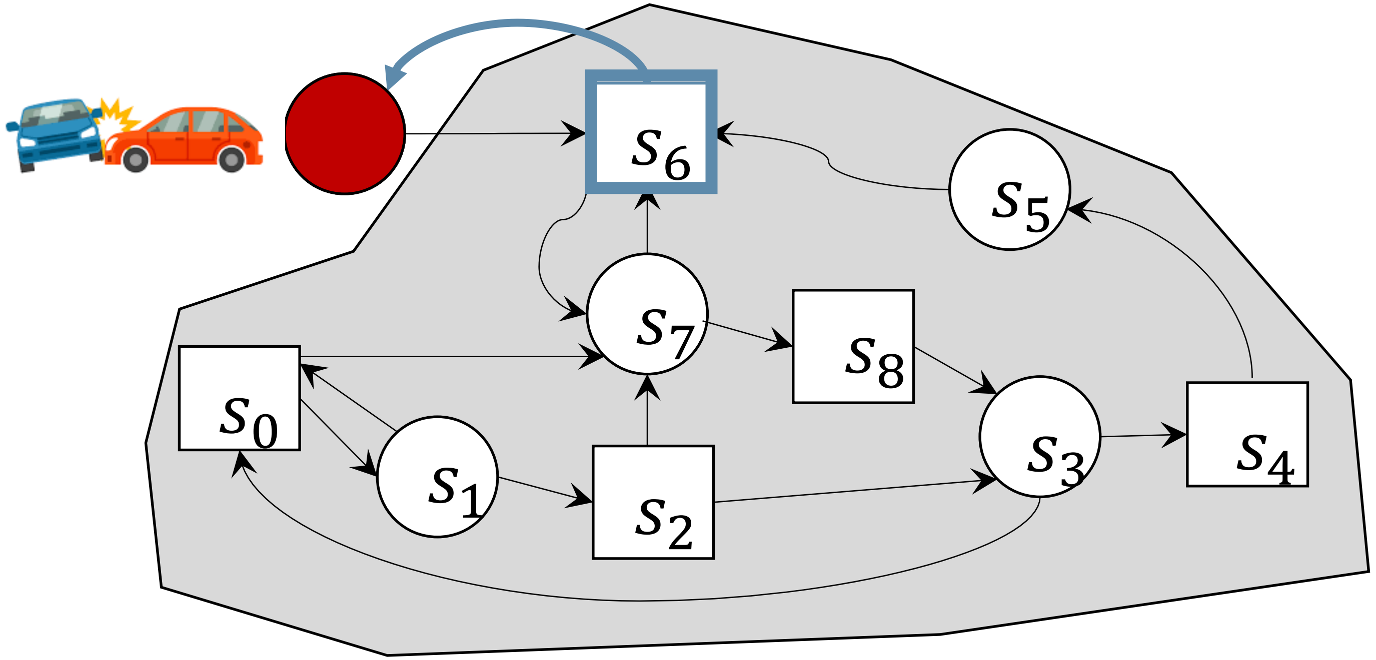


Player 1 wins,
if ● is **never** visited

Winning Region: States from which Player 1 can enforce that ● is **never** visited

Example: Shield Construction = Solve Safety Game

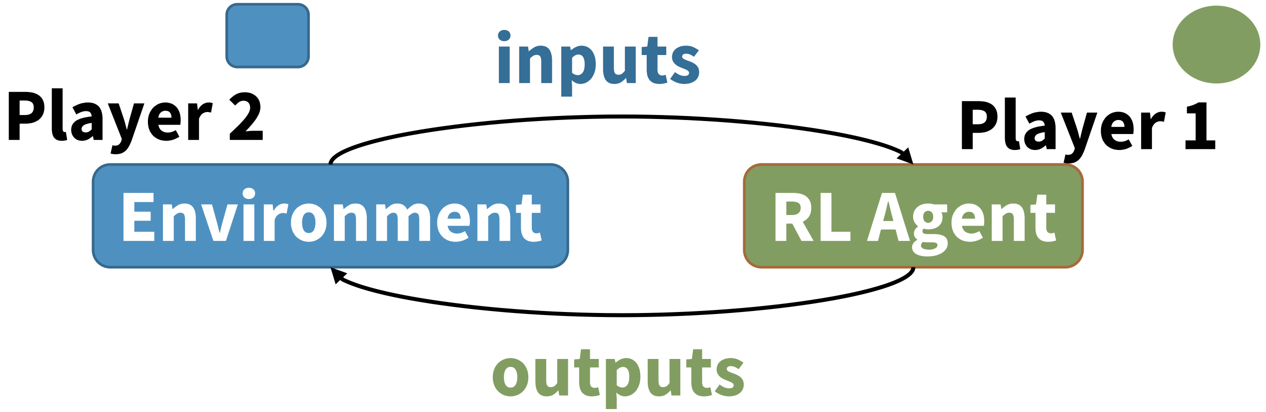


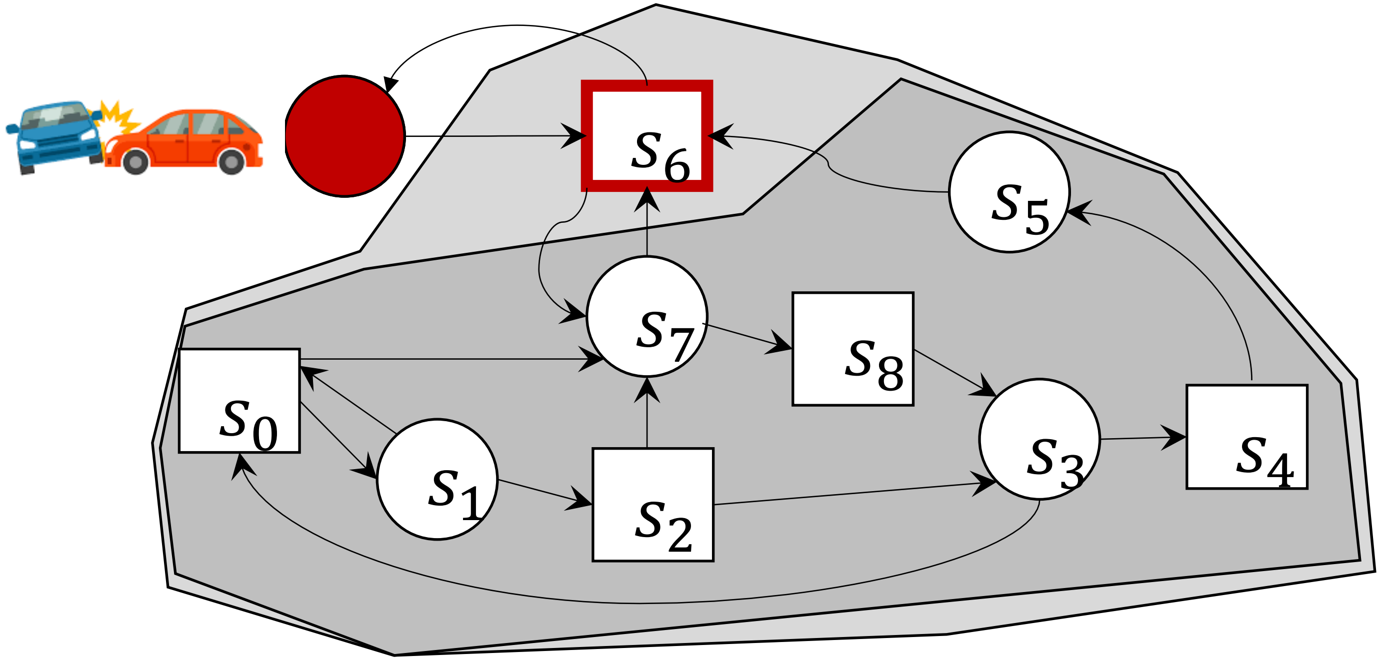


Player 1 wins,
if is **never** visited

Winning Region: States from which Player 1 can enforce that is **never** visited

Example: Shield Construction = Solve Safety Game

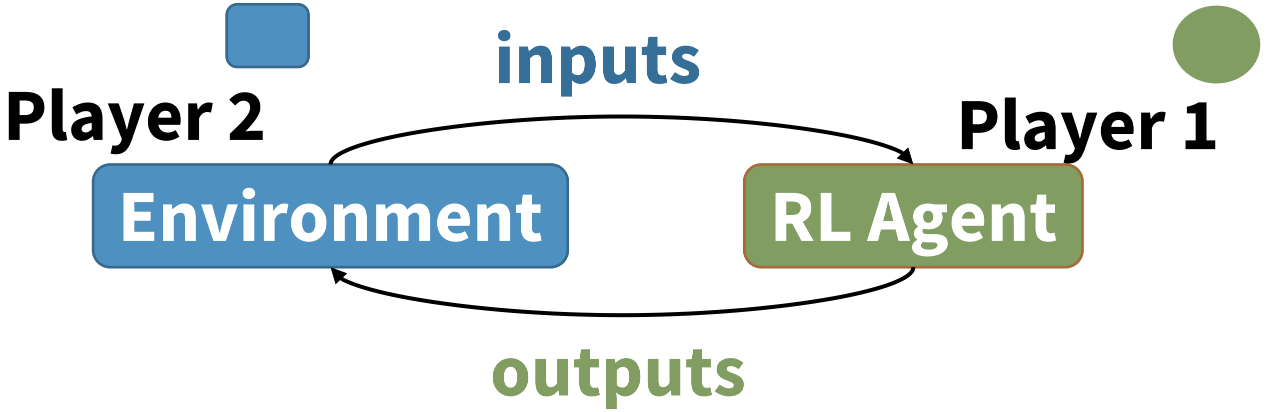


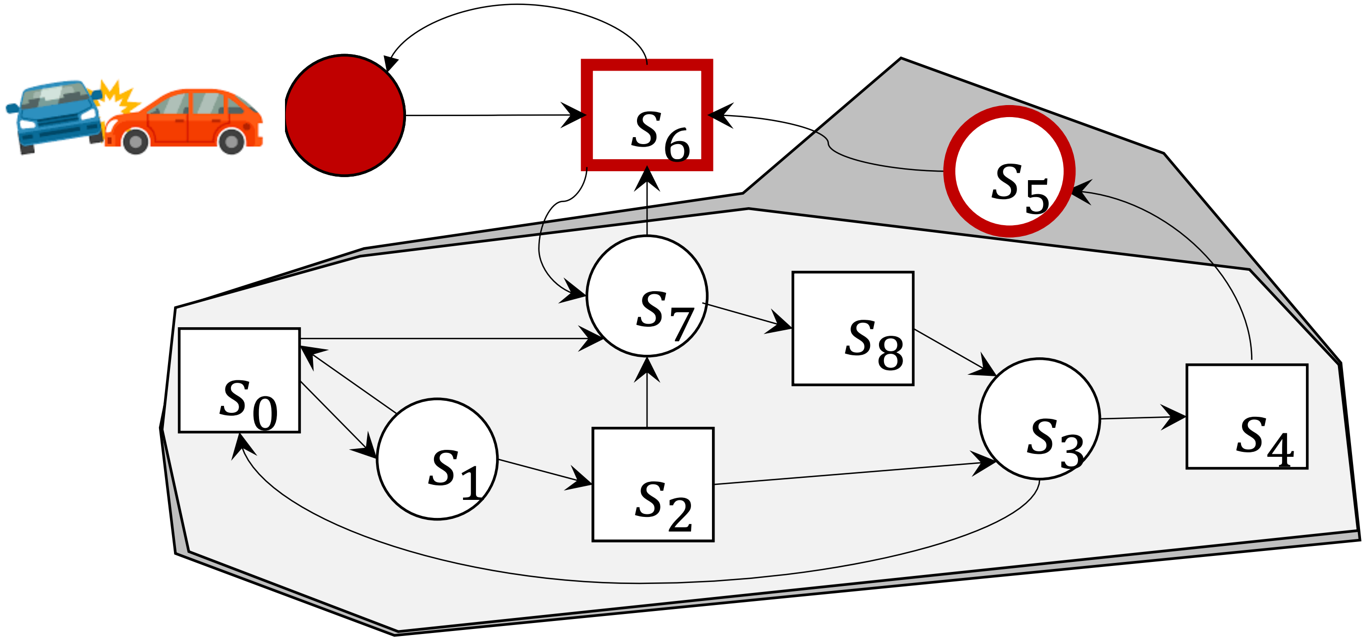


Player 1 wins,
if is **never** visited

Winning Region: States from which Player 1 can enforce that is **never** visited

Example: Shield Construction = Solve Safety Game

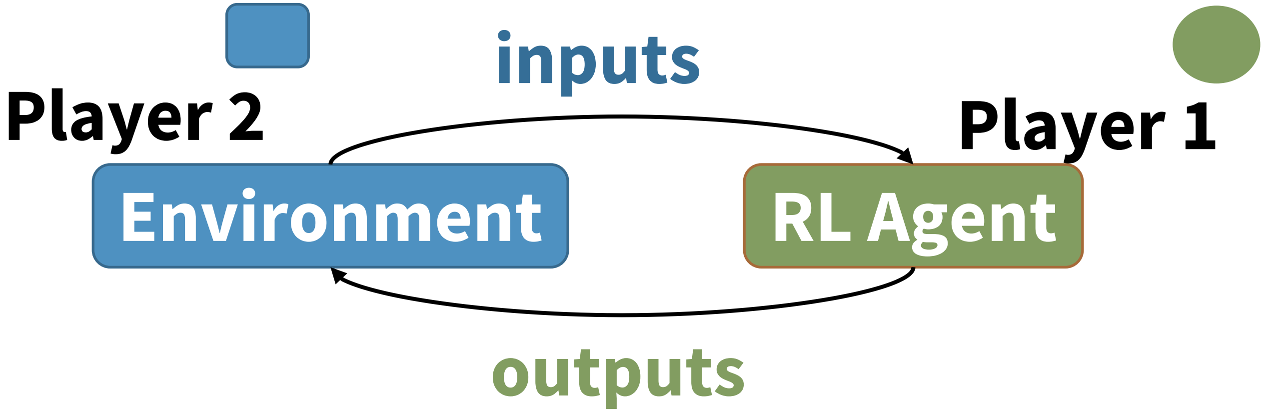


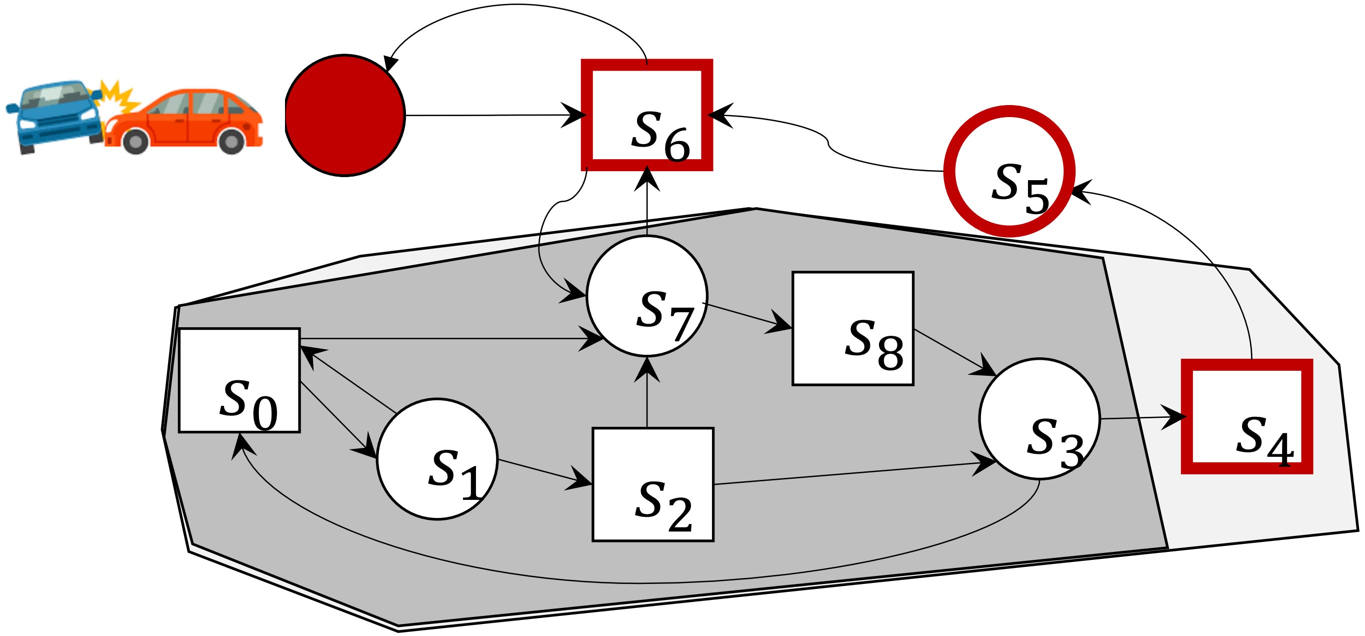


Player 1 wins,
if ● is **never** visited

Winning Region: States from which Player 1 can enforce that ● is **never** visited

Example: Shield Construction = Solve Safety Game

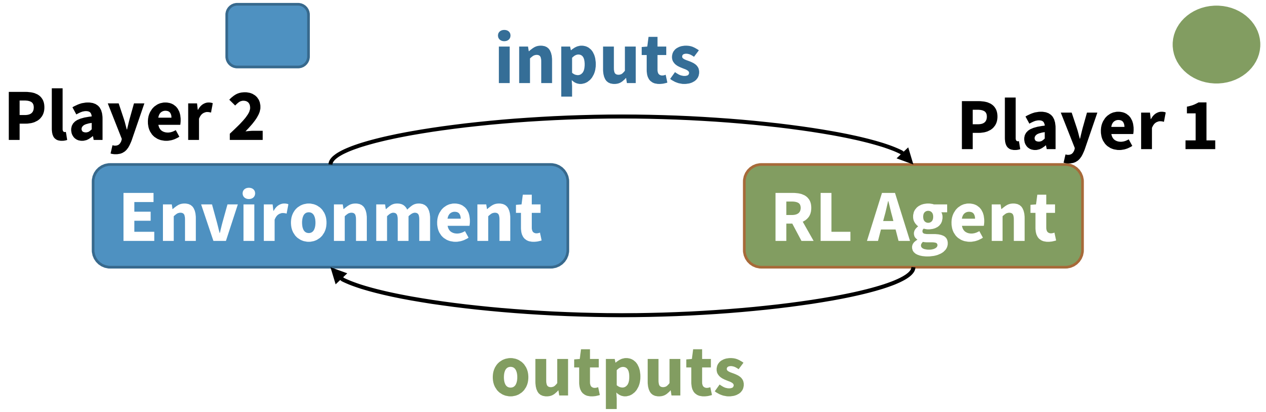


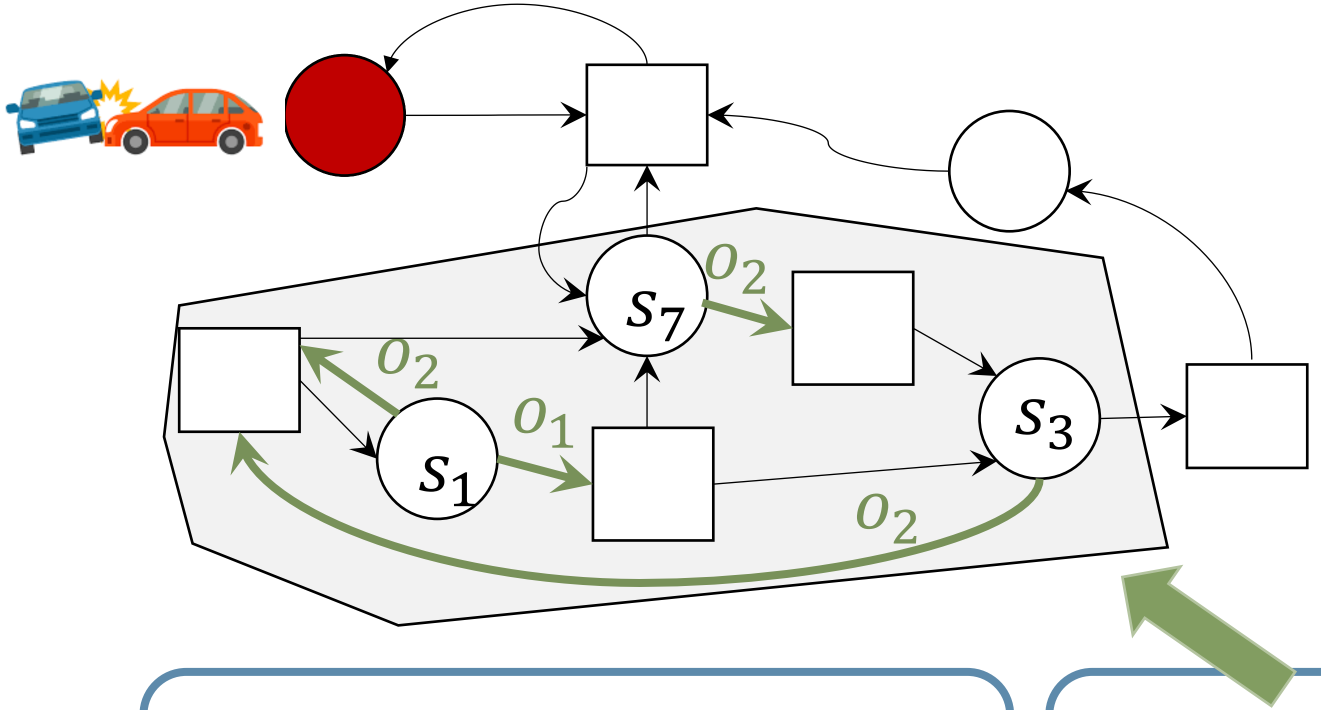


Player 1 wins,
if (red circle) is **never** visited

Winning Region: States from which Player 1 can enforce that (red circle) is **never** visited

Example: Shield Construction = Solve Safety Game

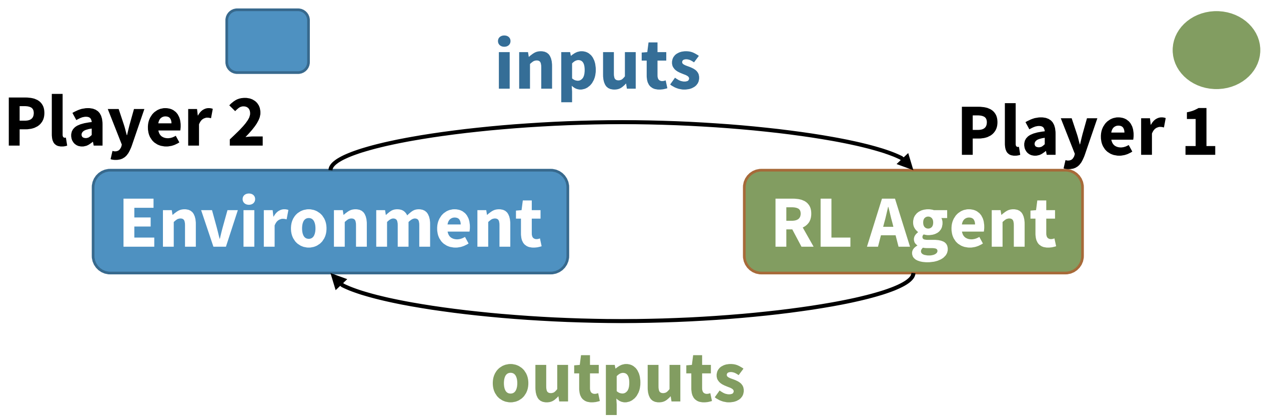


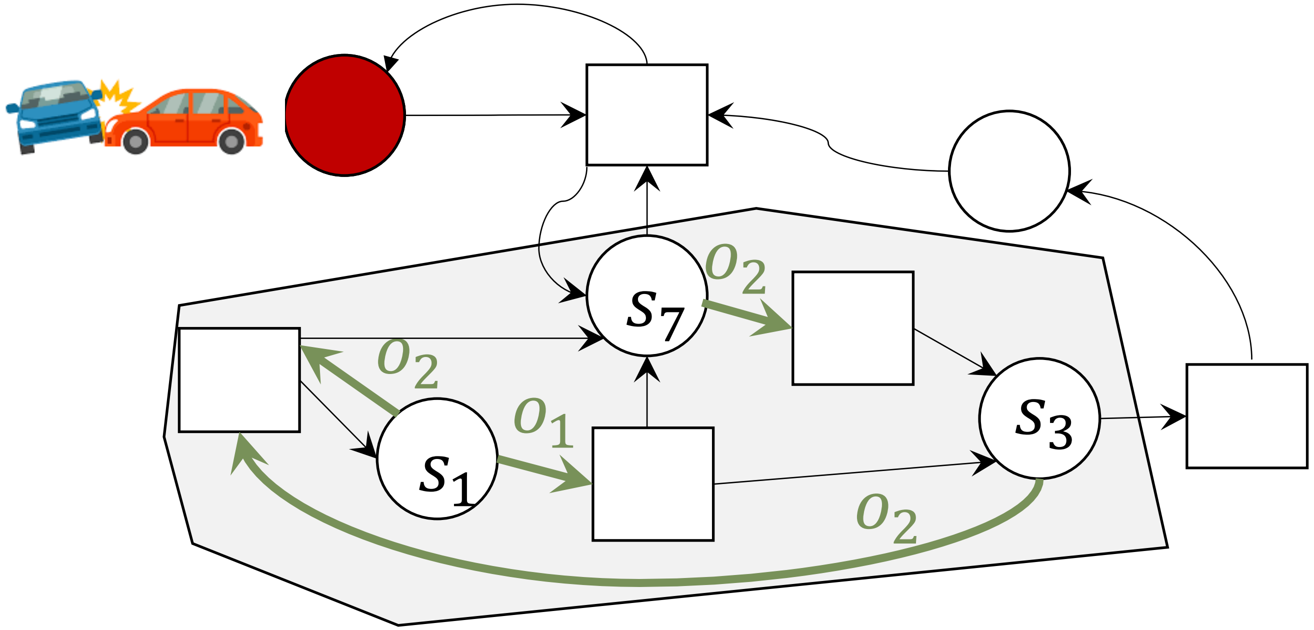


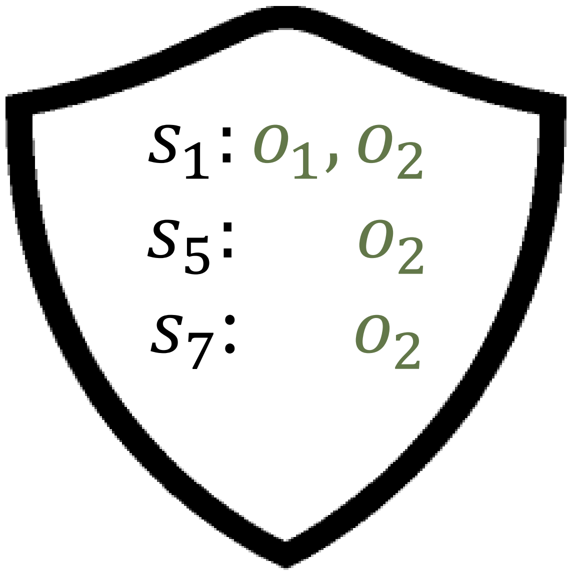
Player 1 wins,
if 🔴 is **never** visited

Winning Region: States from which Player 1 can enforce that 🔴 is **never** visited

Example: Shield Construction = Solve Safety Game

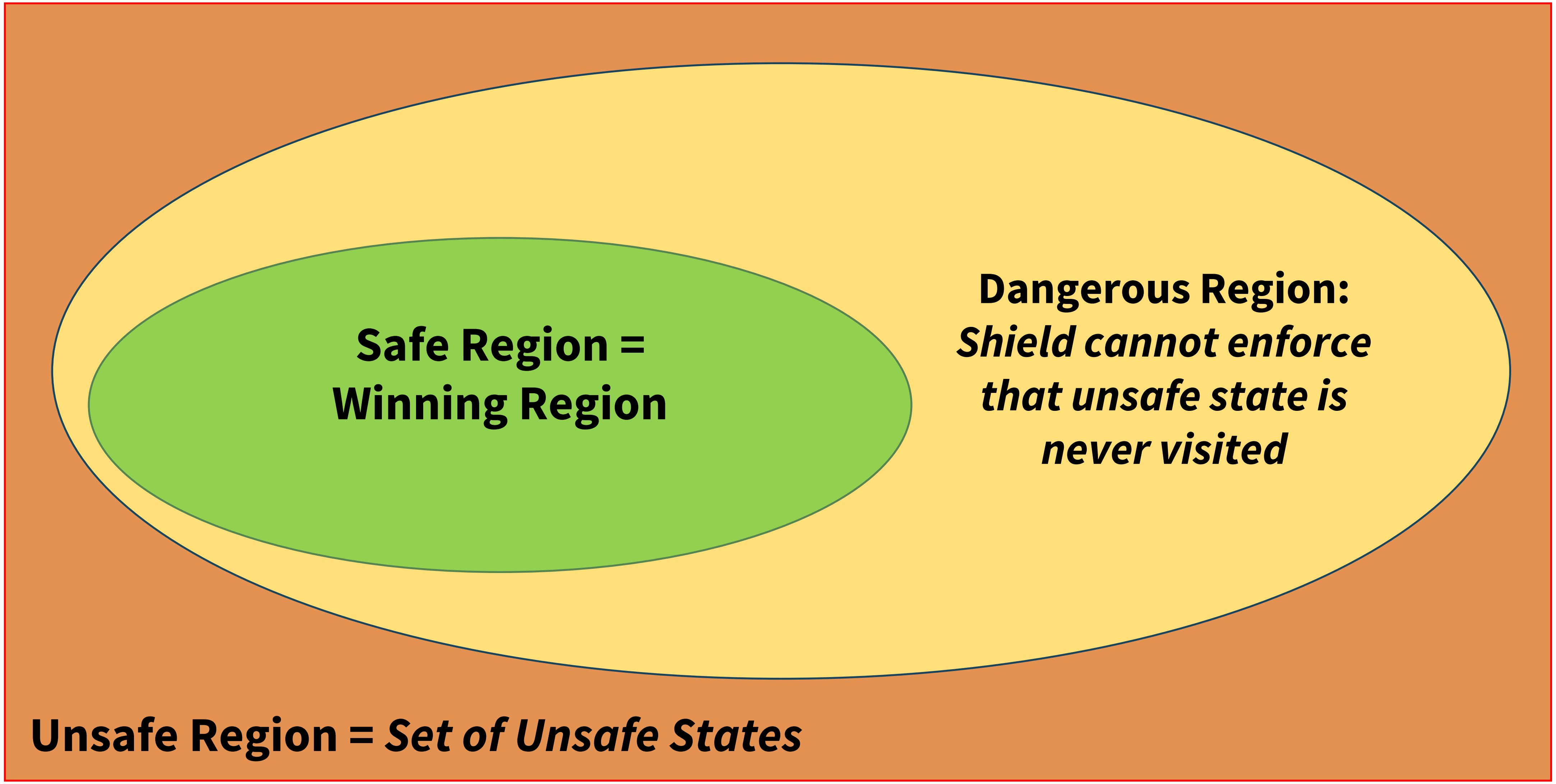




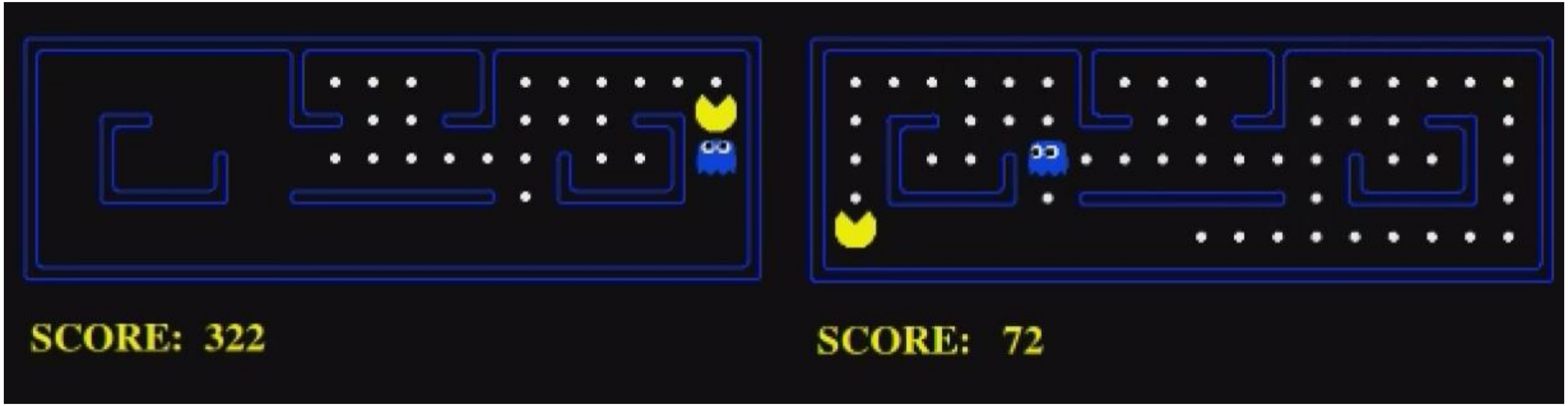


Player 1 wins,
if ● is **never** visited

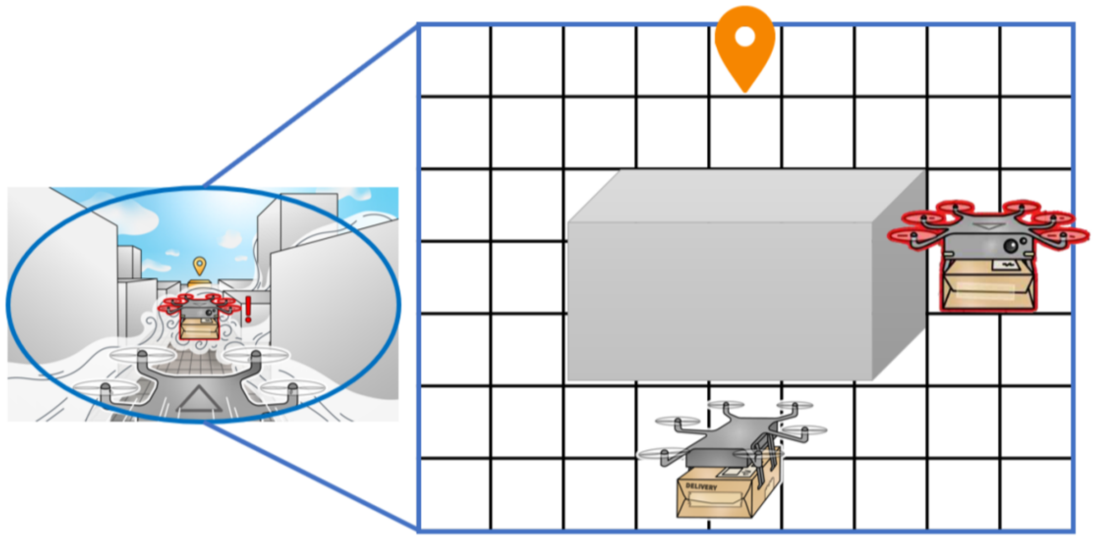
Safe Region / Dangerous Region / Unsafe Region

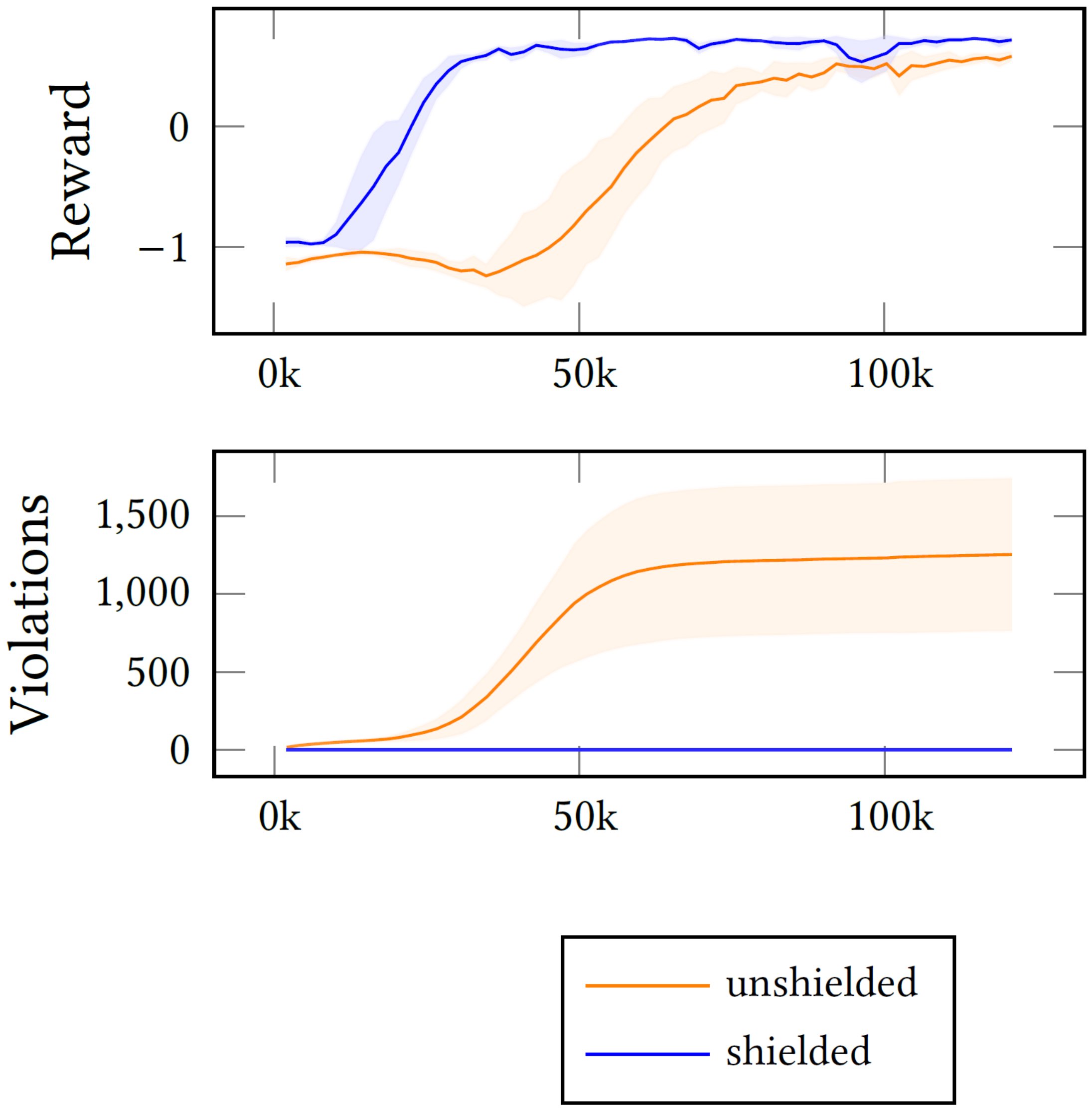


Video Pac-Man



Demo





Reward
0
−1
0k
50k
100k
Violations
1,500
1,000
500
0
0k
50k
100k
unshielded
shielded

Outline

- **Shielding for Safety**
  - Integration of a shield in RL
  - Symbolic Models
  - Shields with Absolute Safety Guarantees
  - Shields with Probabilistic Guarantees

Probabilistic Worldview for RL

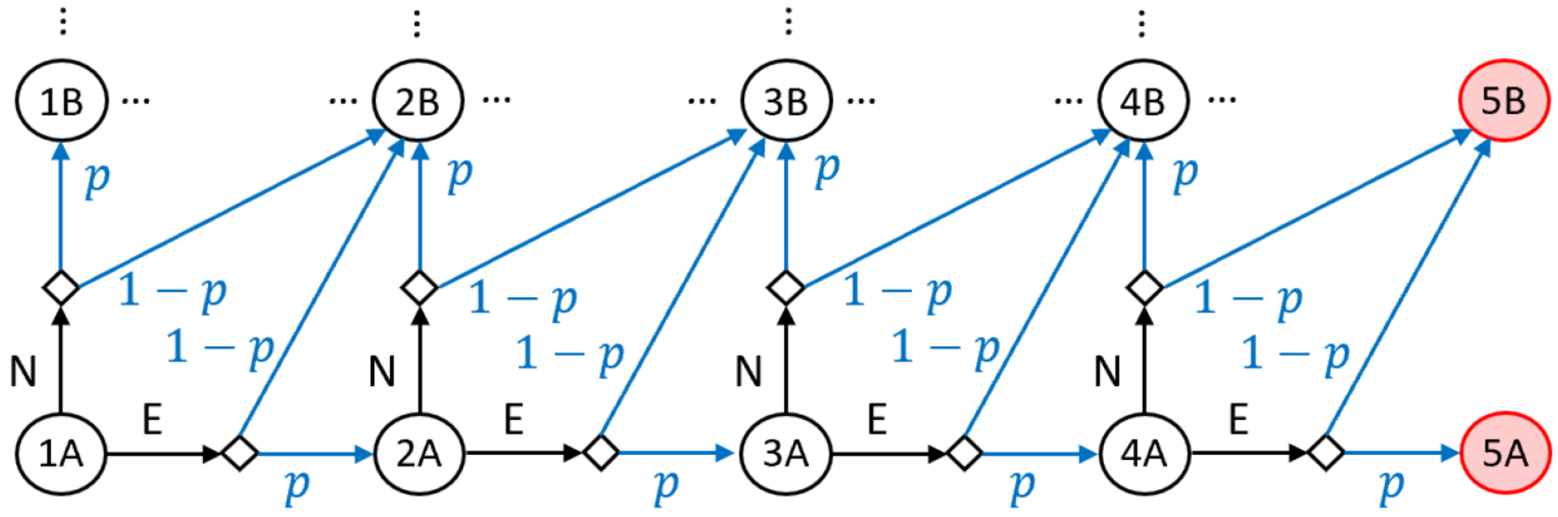
- Worst-case assumptions are too pessimistic
  - E.g. Sensors:
    - Assuming that any sensor always fails does not yield to useful results

- Consider finite horizon
  - A bad event that can happen with low probability at each step , will eventually occur with probability 1.

  - Choose finite horizon of h steps
    - E.g. $h \; =$ mission time, or expected battery life…

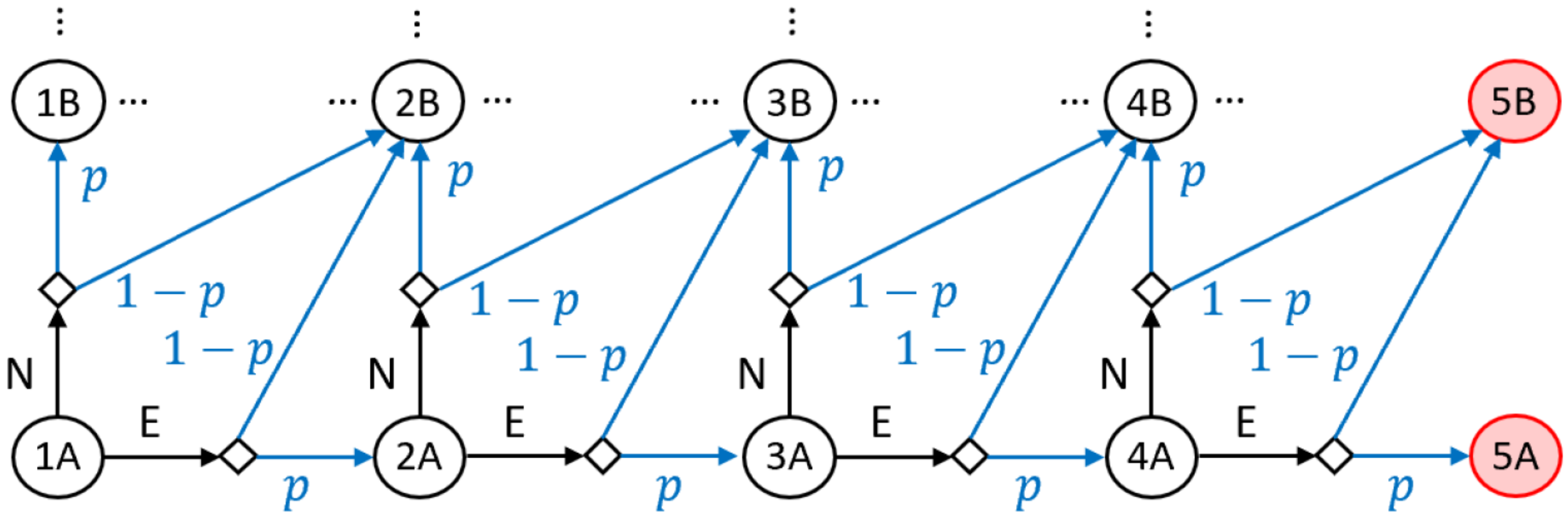
Shield with Probabilistic Safety Guarantees

- *Given*: MDP M, safety spec φ defines set of unsafe states in M
- **Shield: Limits probability of visiting an unsafe state.**



Probabilistic Model Checking

- $M = (S, s_0, A, P)$ … Markov Decision Process (MDP)
- $\pi\colon\ S \to A$ … policy
- $M^\pi = (S, s_0, P)$ induced Markov Chain by applying π to M



Probabilistic Model Checking

$\varphi = \mathrm{G(safe)}$, policy π, MDP M

Model Checking:

- $\mathbb{P}_{M^\pi,\varphi}: S \times N \rightarrow [0,1]$ … expected probability to satisfy φ from a state s within h steps in the MC M^π
- $\mathbb{P}^{max}_{M,\varphi}(s,h) = \max_{\pi\in\Pi} \mathbb{P}_{M^\pi,\varphi}(s,h)$ … **maximal** expected probability over all policies to satisfy φ from a state s within h steps.
- $\boldsymbol{\mathbb{P}^{max}_{M,\varphi}(s,a,h) = \sum_{s'\in S} P(s,a,s') \cdot \mathbb{P}^{max}_{M,\varphi}(s',h-1)}$

… **maximal** expected probability over all policies to satisfy φ from a state s when **taking action** $\boldsymbol{a}$ within h steps.

Simple Shield for Quantitative Safety

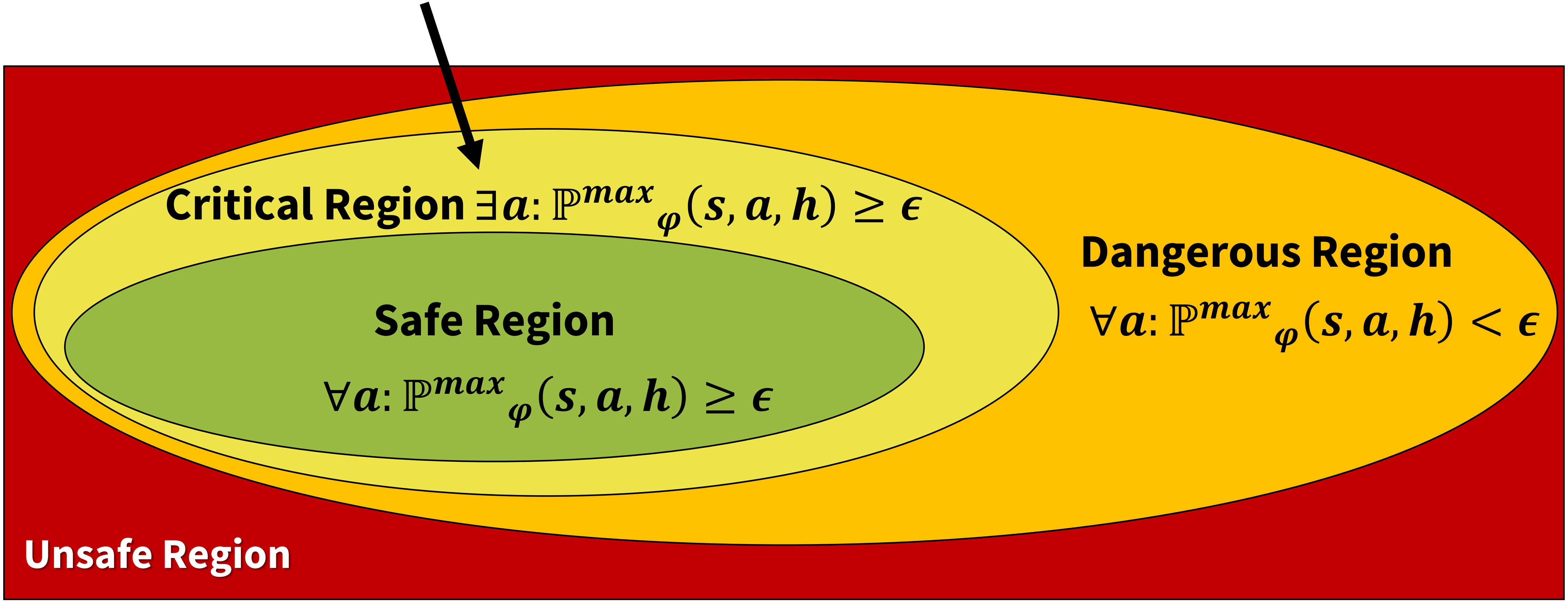
- **Shielding Objective** $\langle \varphi, h, \epsilon \rangle$
  - $\varphi = \mathrm{G}(\text{safe})$
  - h … finite horizon
  - ϵ … safety threshold

$\forall s \forall a$: if $\mathbb{P}^{max}{}_{\varphi}(s, a, h) < \epsilon$ then ***a*** **is shielded in** ***s***

- **Idea of using** $\mathbb{P}^{max}$**:**
  - $\mathbb{P}^{max}_{M,\varphi}(s, a, h) = \sum_{s' \in S} P(s, a, \mathbf{s}') \cdot \mathbb{P}^{max}_{M,\varphi}(\mathbf{s}', h - 1)$
  - Shield interferes, if after executing a, the **safest policy** is too risky

Simple Shield for Quantitative Safety

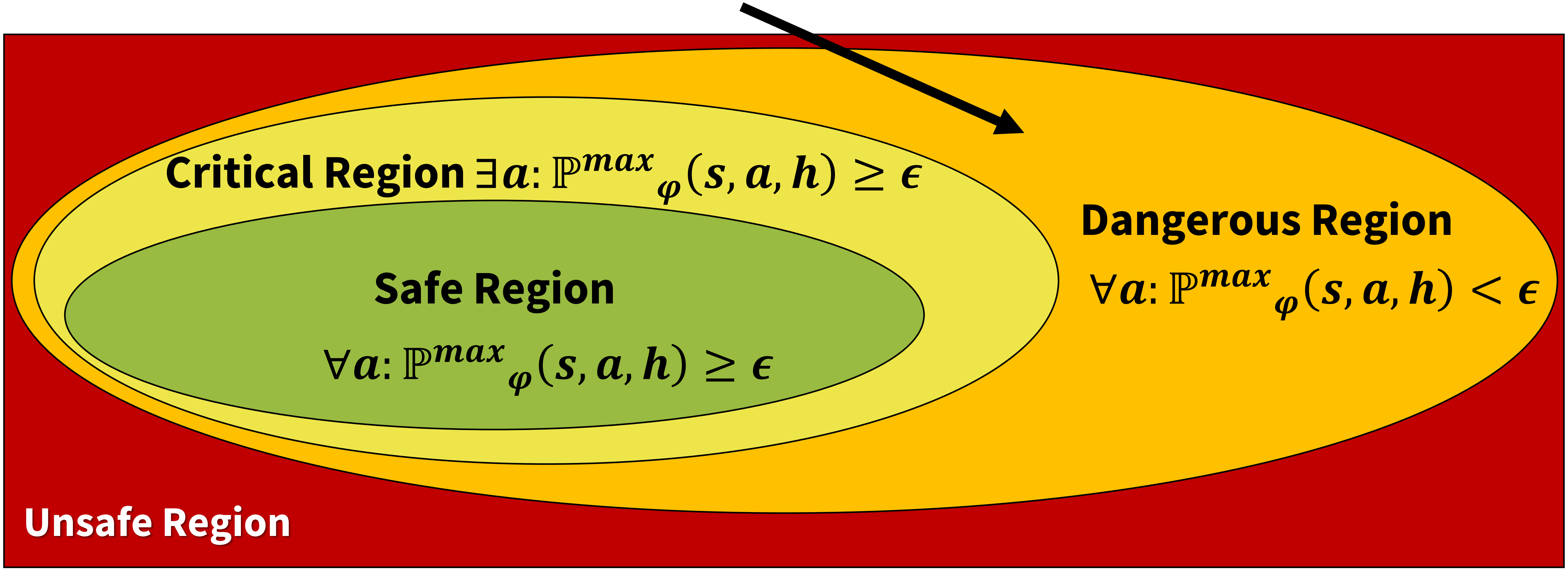
Shield: blocks actions with $\mathbb{P}^{max}{}_{\varphi}(s, a, h) < \epsilon$



Simple Shield for Quantitative Safety

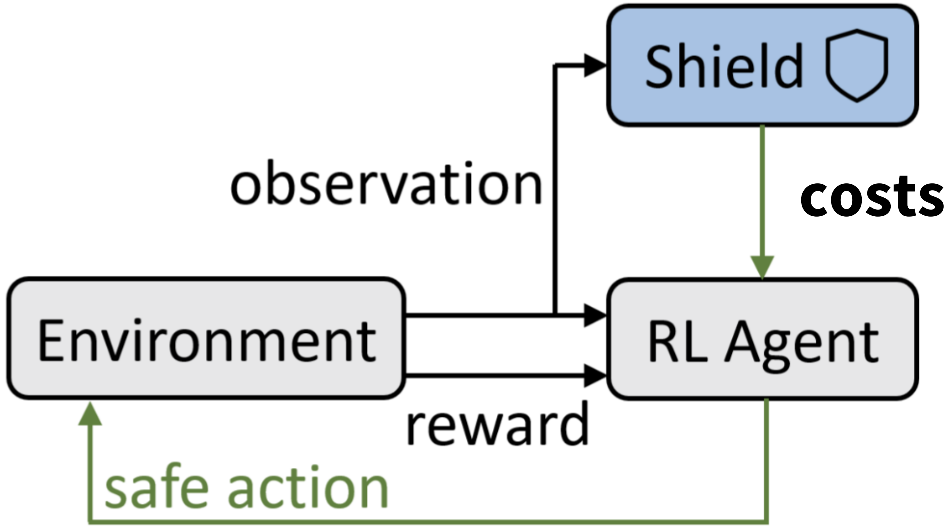
Shield: Domain specific solution

- Allow only the safest action
- Pre-defined fallback action (breaking), hand over control to human…



Shield Integration via Constrainted RL

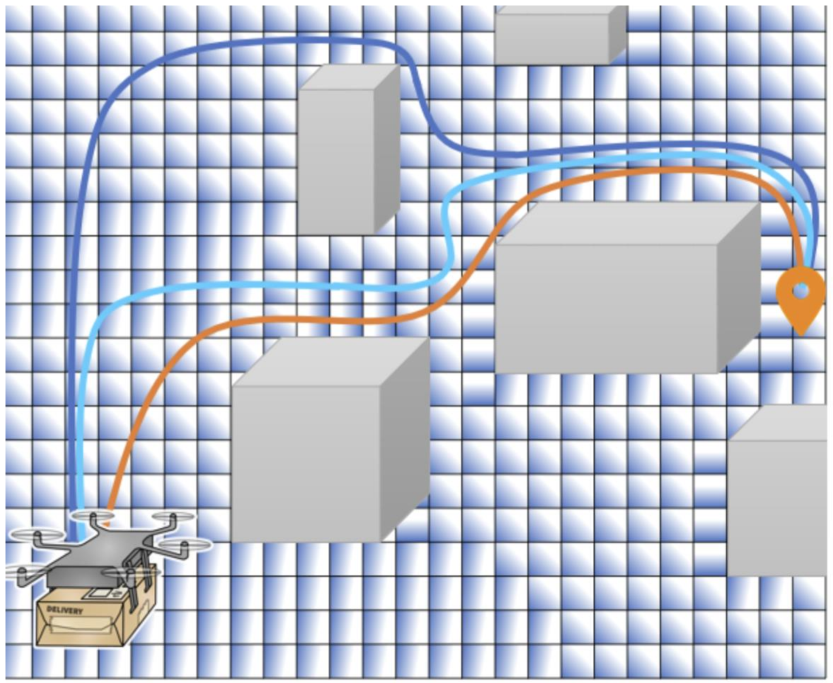
- Agent should learn to behave safely

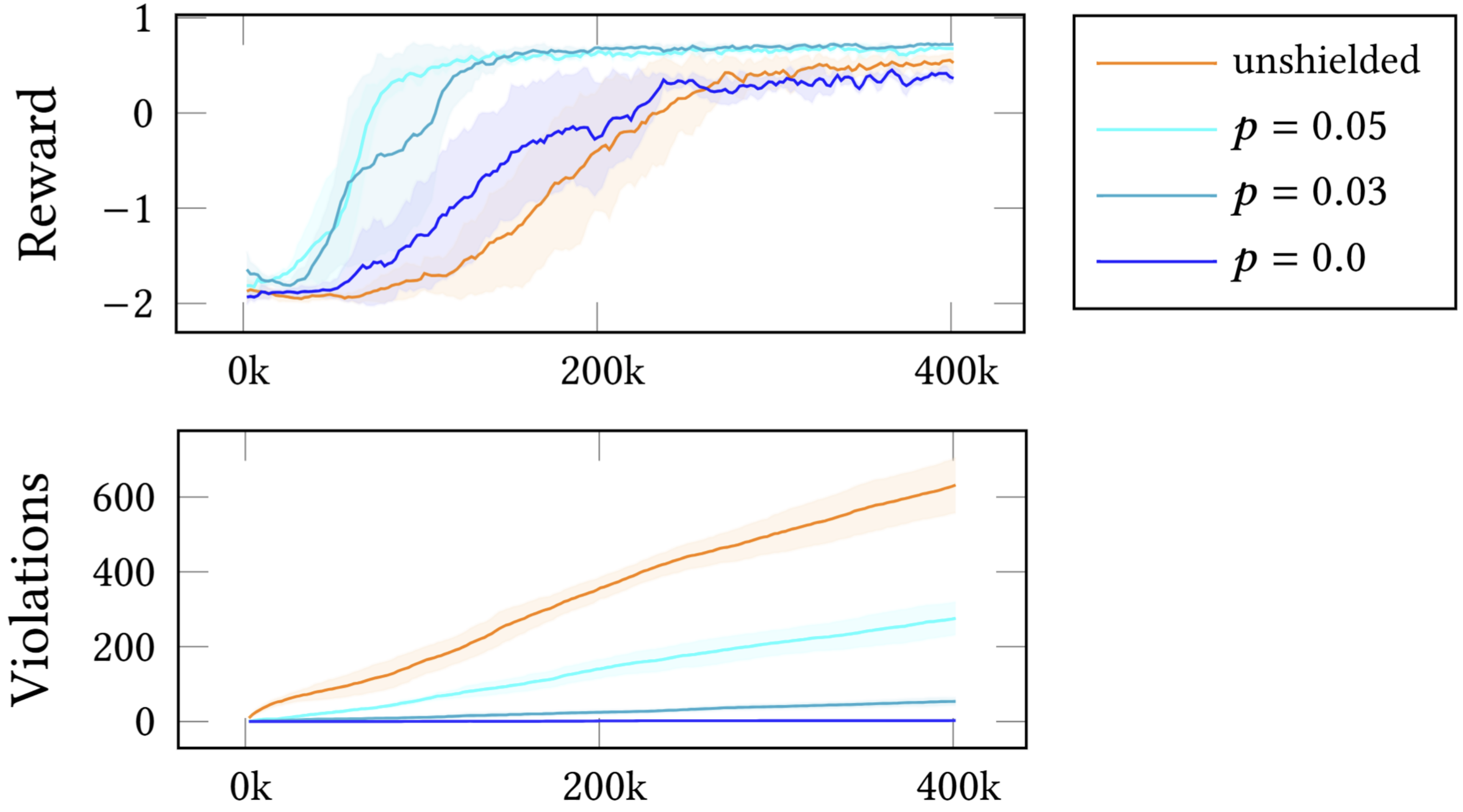


$$\boldsymbol{C(s,a)} = \mathbb{P}^{\boldsymbol{max}}_{\boldsymbol{M,\varphi}}(\boldsymbol{s},\boldsymbol{a})$$

- Find policy $\max_{\theta} J_R^{\pi_\theta} \; s.t. \; J_C^{\pi_\theta} = \mathbb{P}[\sum_t C\,(s_t, a_t) \geq \eta] \leq \epsilon$

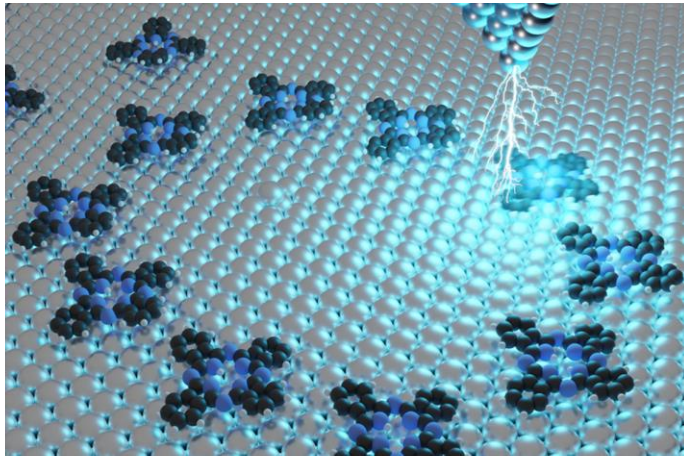
Demonstration

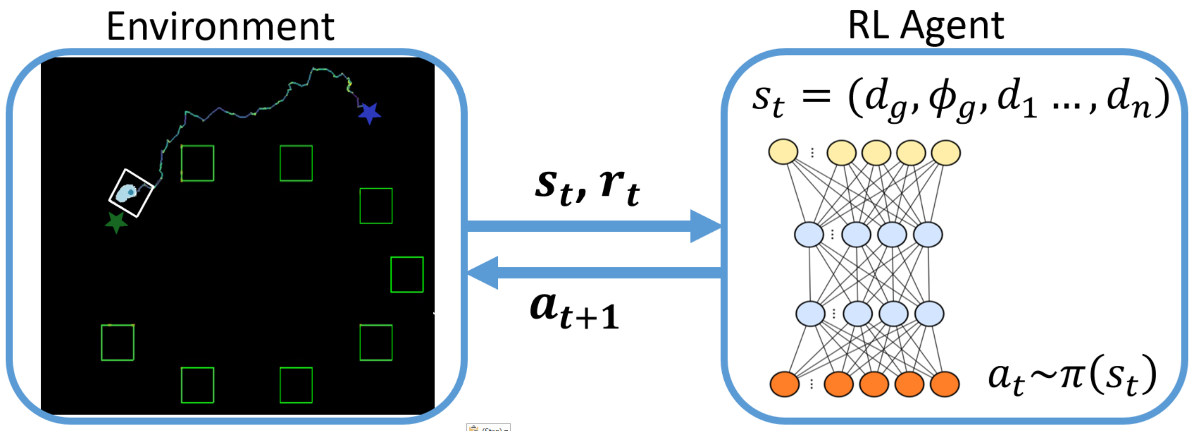


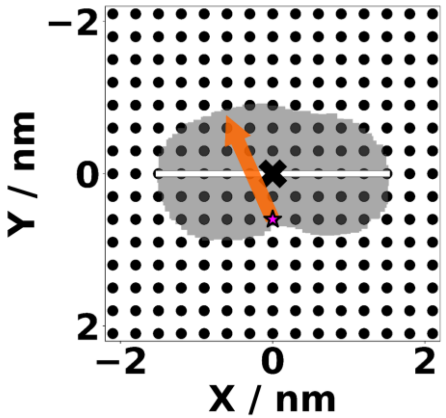


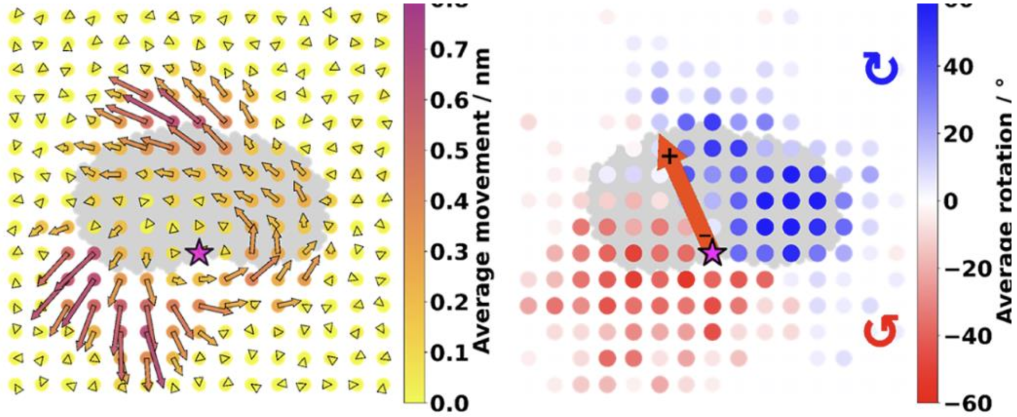
- Training: maskable Proximal Policy Optimization, via Stable-Baseline3, default parameters
- 5% prob. that wind displaces UAV
- Shield enforces that the **minimal probability** of reaching an unsafe state within $\mathbf{20}$ steps is at most $\boldsymbol{p} \in \{\mathbf{0.0}, \mathbf{0.03}, \mathbf{0.05}\}$

Demo Molecular Assembly

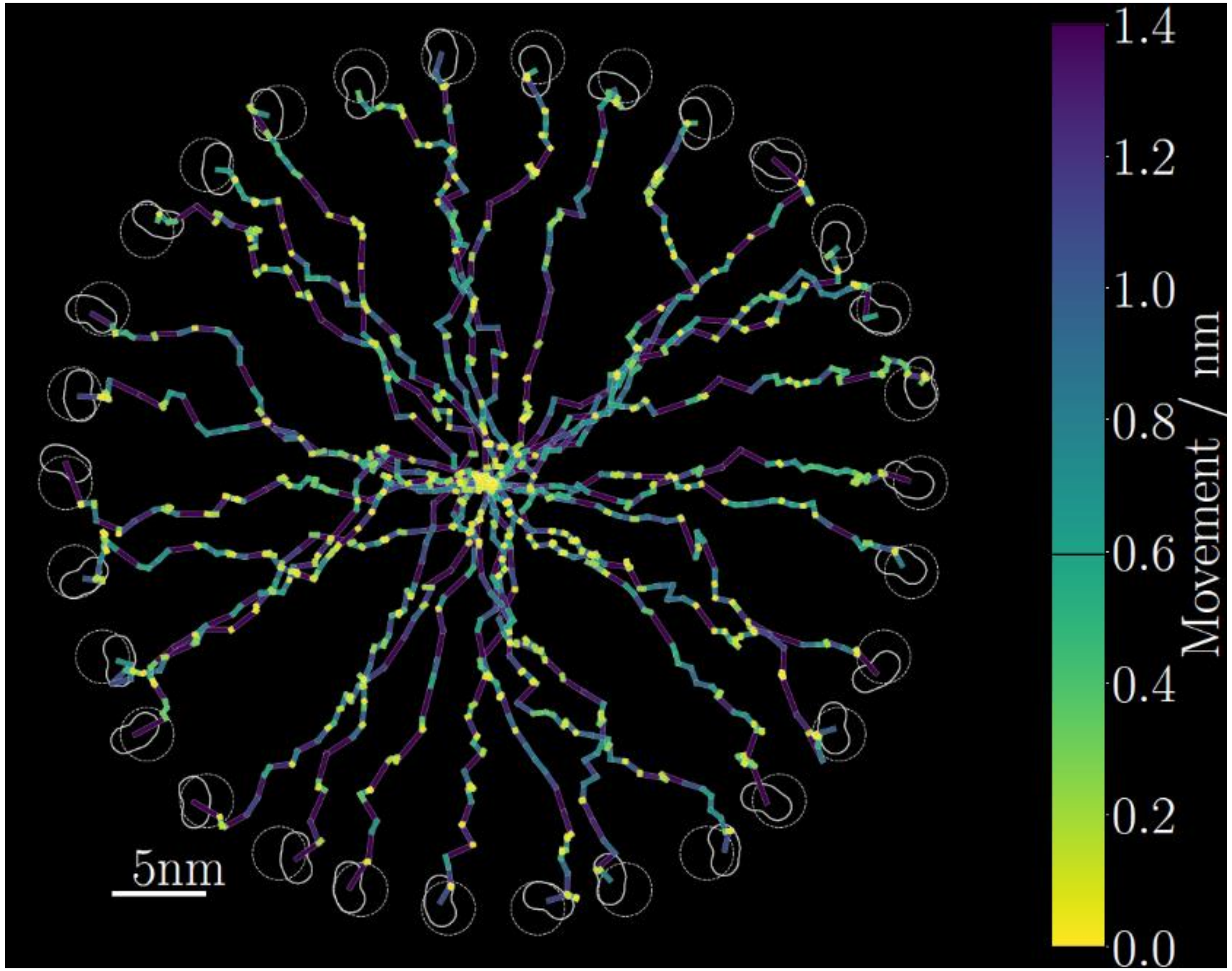






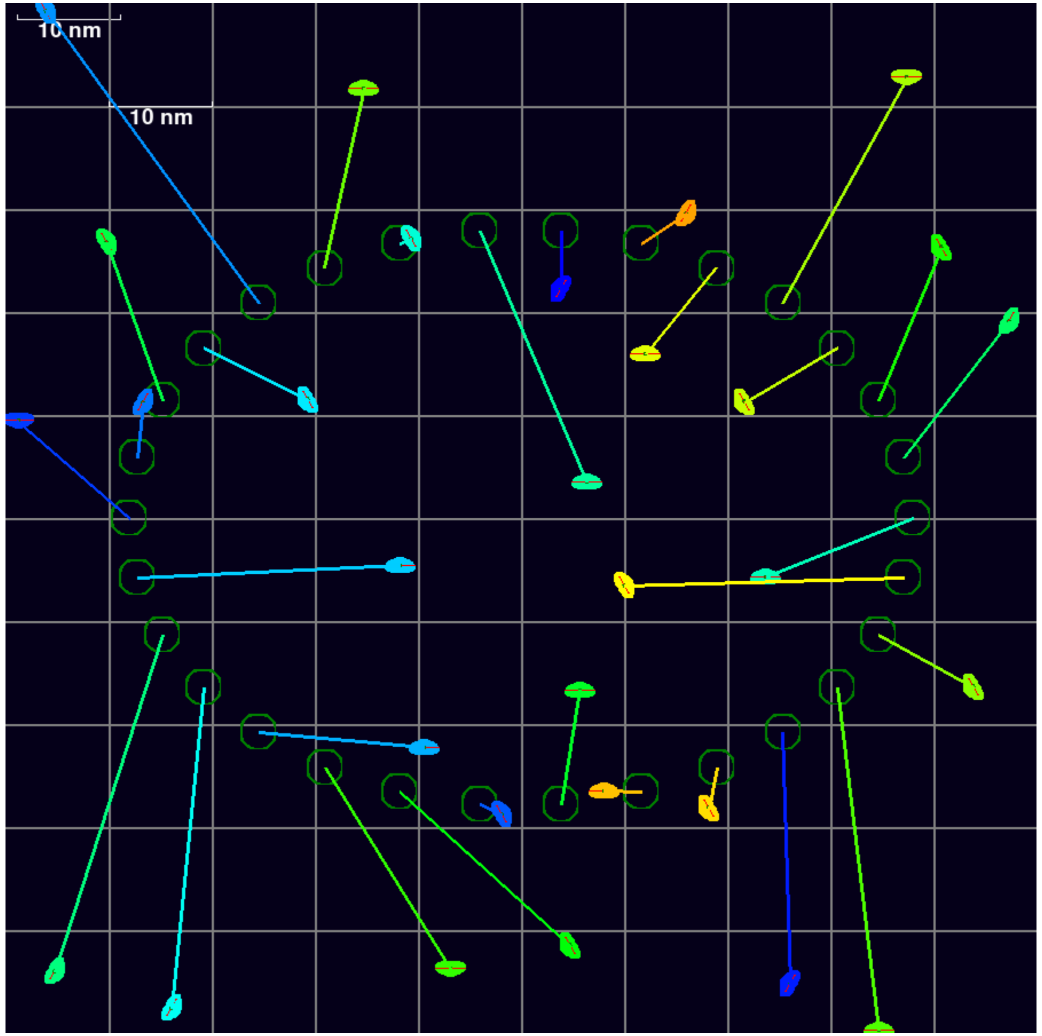


Demo Molecular Assembly

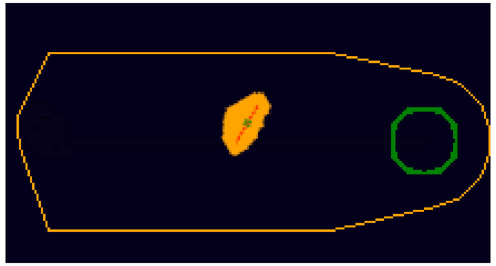


Demo Molecular Assembly

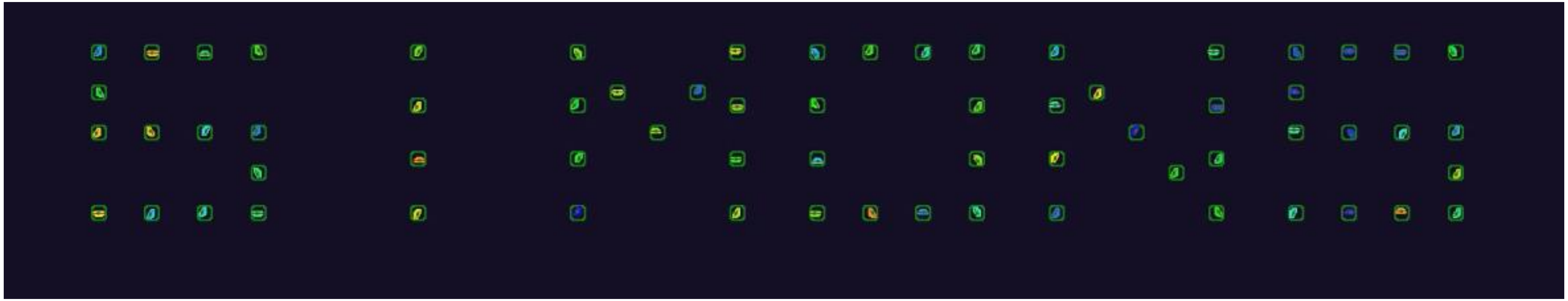
SAT-based matching and scheduling



Shield: Forces molecule to stay in corridor

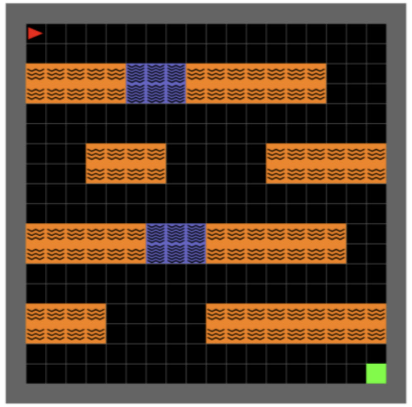


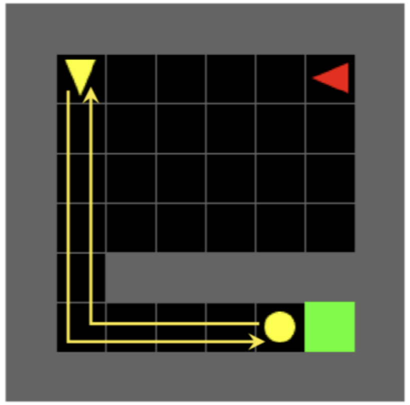
Demo Molecular Assembly

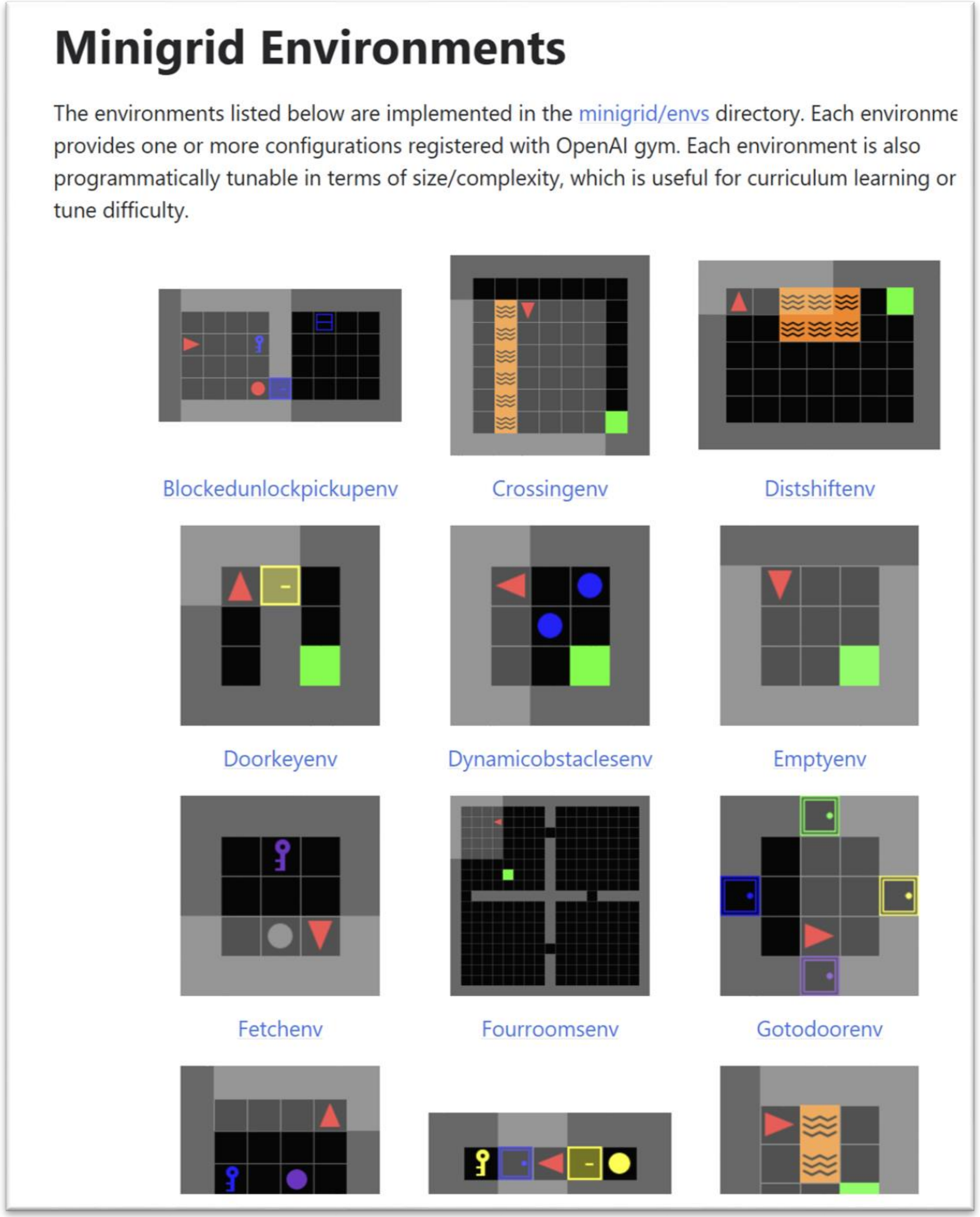


Playground for Shielding

- MinigridSafe
- TEMPEST
  - Integrates Tempest directly in the Gymnasium API







Minigrid Environments

The environments listed below are implemented in the minigrid/envs directory. Each environme provides one or more configurations registered with OpenAI gym. Each environment is also programmatically tunable in terms of size/complexity, which is useful for curriculum learning or tune difficulty.

Shields are great...

...if you have an accurate world model.