Reinforcement Learning for $\mathrm{LTL}_f/\mathrm{LDL}_f$ Goals

SAPIENZA
UNIVERSITÀ DI ROMA

Marco Favorito

M.Sc. in
Engineering in Computer Science
at Sapienza, University of Rome

A.Y. 2018/2019

RL for $\text{LTL}_f/\text{LDL}_f$ goals: What is it? (1)

- A joint work with:
  - Giuseppe De Giacomo
  - Luca Iocchi
  - Fabio Patrizi
- The main topic of my M.Sc. thesis
- Publication:
  - De Giacomo, Giuseppe, et al. "Reinforcement Learning for LTLf/LDLf Goals." arXiv preprint arXiv:1807.06333 (2018).

RL for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals: What is it? (2)

It is a **Reinforcement Learning framework** that:

- Allows to specify temporal goals (in LTL_f or LDL_f)
- Allows the RL agent to learn them

Advantages:

- The agent does not need to know the fluents
- We can rely on off-the-shelf RL algorithms (Q-Learning, Sarsa, ...)
- **Theorem**: the agent learns to do "the best" (in terms of rewards), given the $\mathrm{LTL}_f/\mathrm{LDL}_f$ constraints

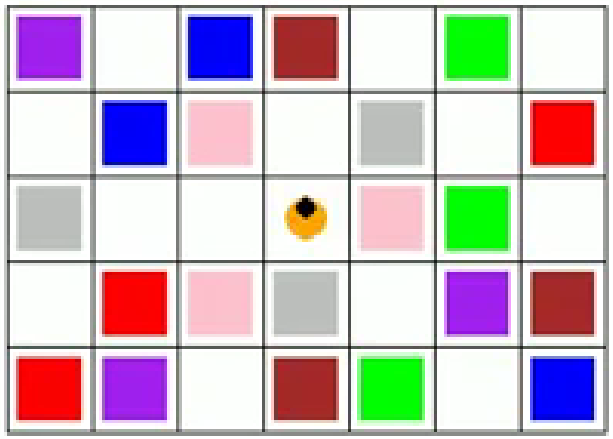
A Toy Example: SAPIENTINO

The robot **state space** $S = \{(x_1, y_1), (x_2, y_1), \dots\}$

Fluents $\mathcal{F} = \{red, green, blue, pink, \dots\}$

Goal: visit colors in a given order, e.g. $\langle red, blue, green \rangle$.

LTL_f formula $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$



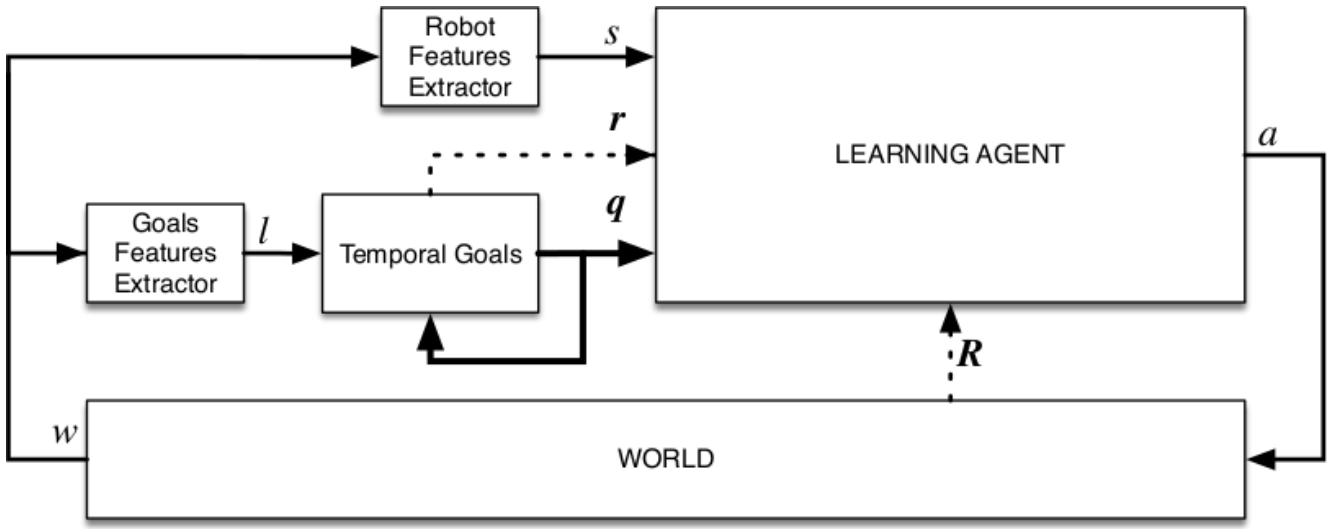
Notice that φ is a **non-Markovian** goal/reward, i.e. **depends on a sequence of transitions**

RL for $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals: a new problem

Two-fold representation of the world $\mathcal{W}$:

- An agent learning an MDP with **low-level features** S, trying to optimize reward R
- $\mathrm{LTL}_f/\mathrm{LDL}_f$ goals $\{(\varphi_i, r_i)_{i=1}^m\}$ over a set of **high-level features** $\mathcal{F}$, yielding a set of fluents configurations $\mathcal{L} = 2^{\mathcal{F}}$

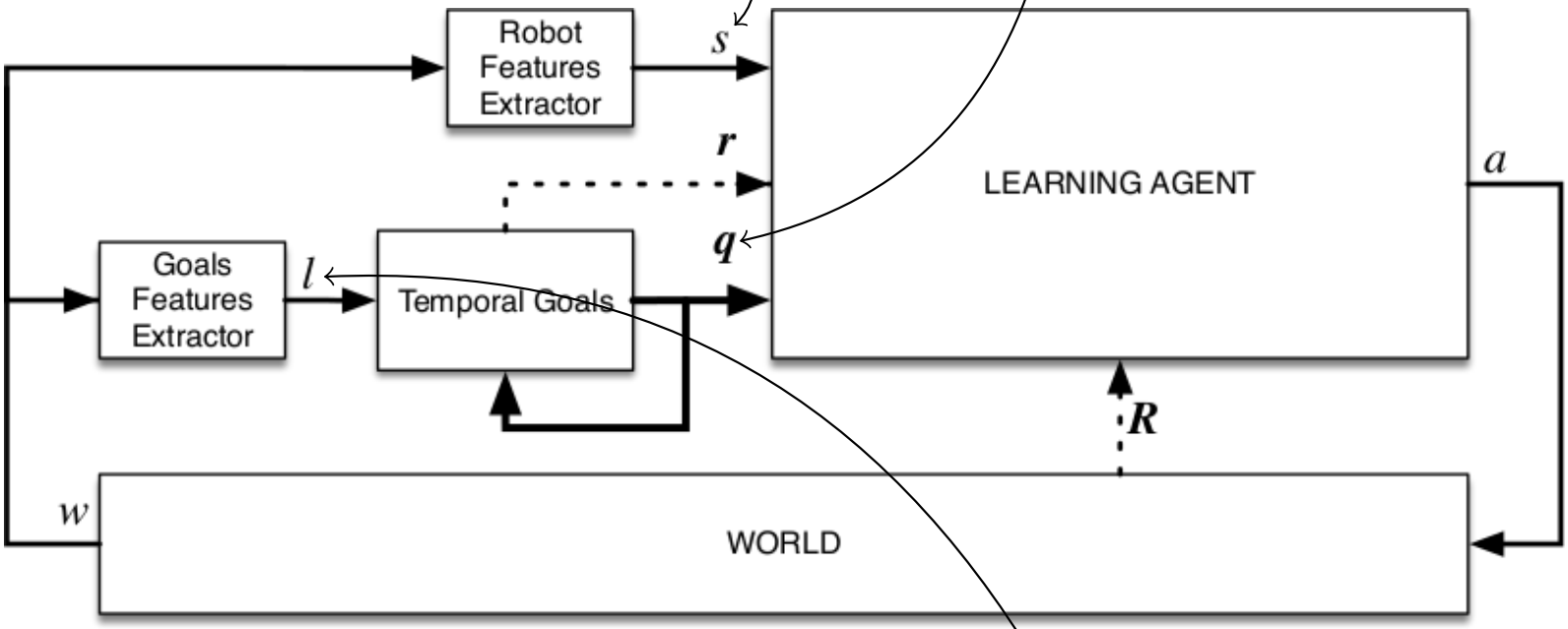
Solution: a non-Markovian policy $\rho : S^* \rightarrow A$ that is optimal wrt rewards r_i and R.



Our approach:

- Transform each φ_i into DFA $\mathcal{A}_{\varphi_i}$
- Do RL over an MDP $\mathcal{M}'$ with a transformed state space:

$$S' = S \times Q_1 \times \cdots \times Q_m$$

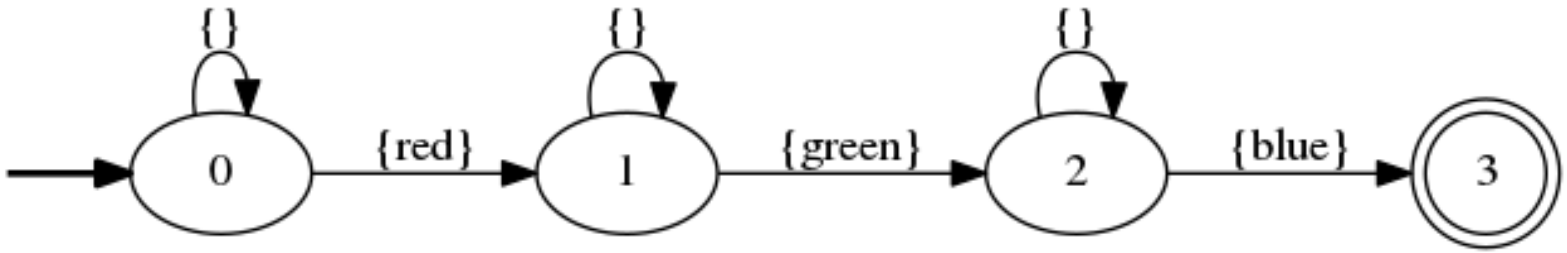


Notice: **the agent ignores the fluents** $\mathcal{L}$!

The actual RL relies on standard RL algorithms (e.g. Sarsa(λ))

An optimal policy for SAPIENTINO

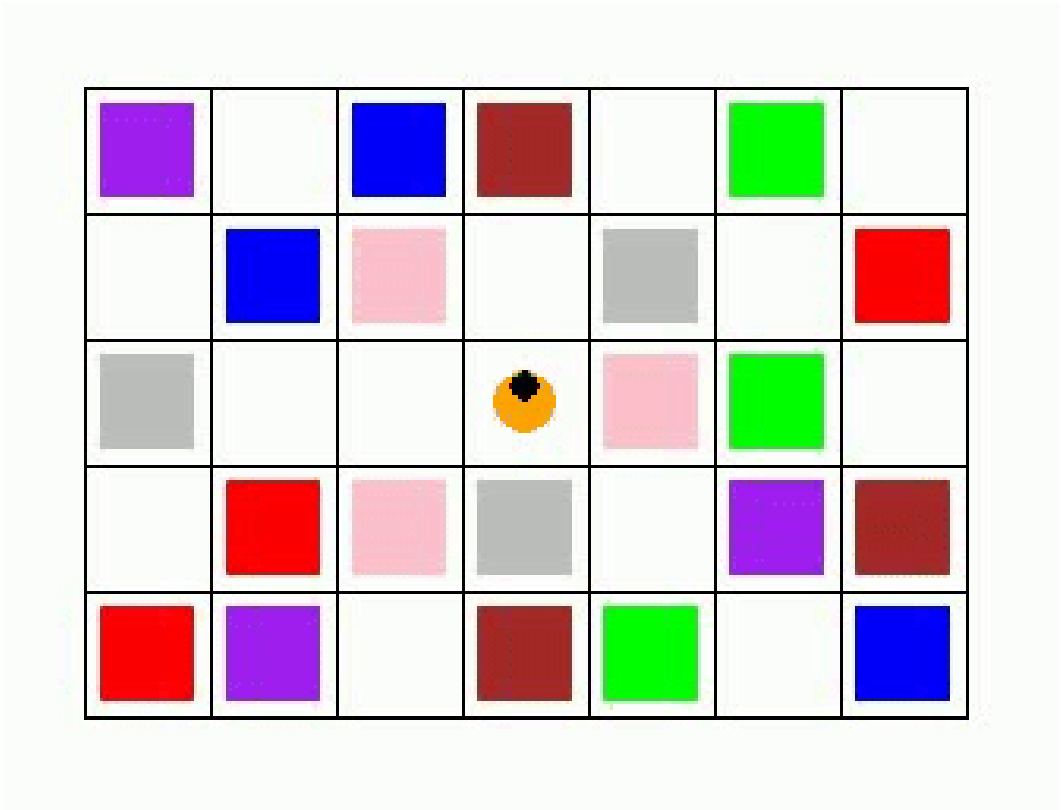
LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$
The equivalent DFA is (roughly):

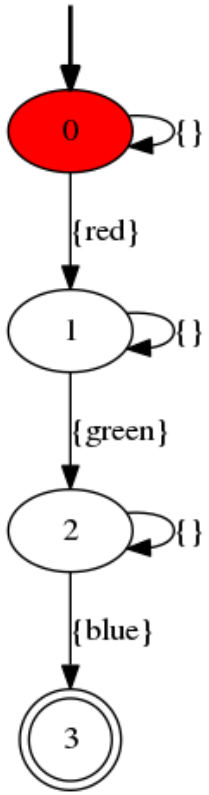


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_4, y_3, 0)$

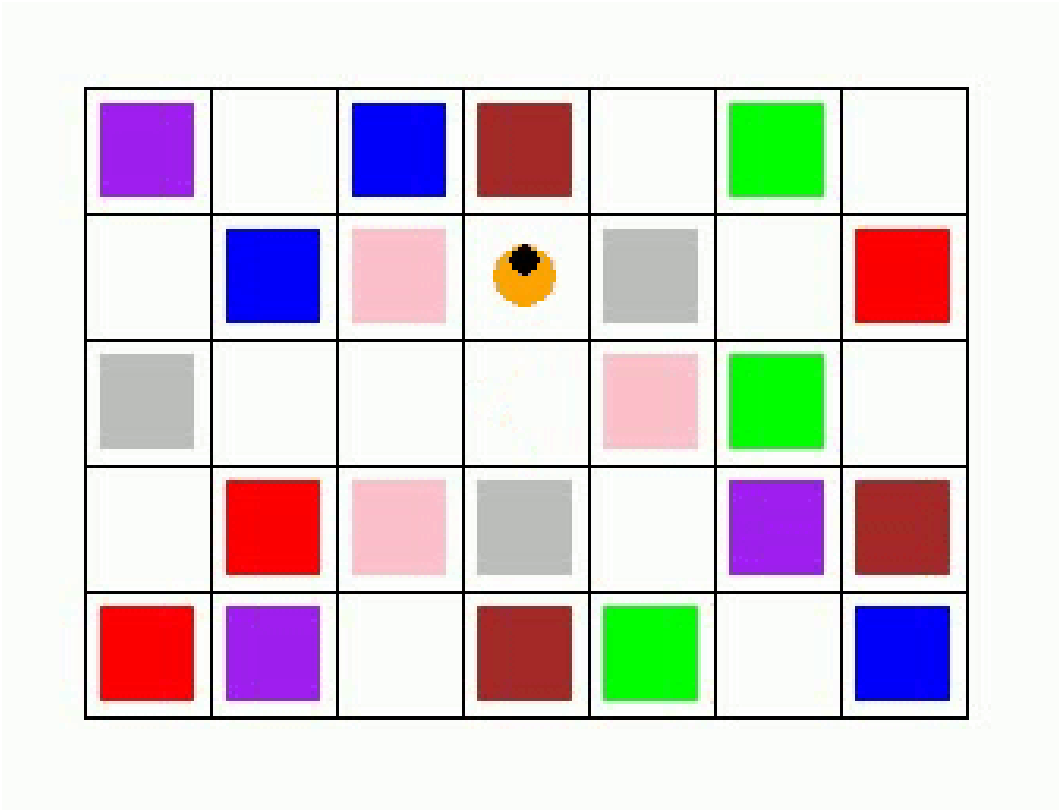


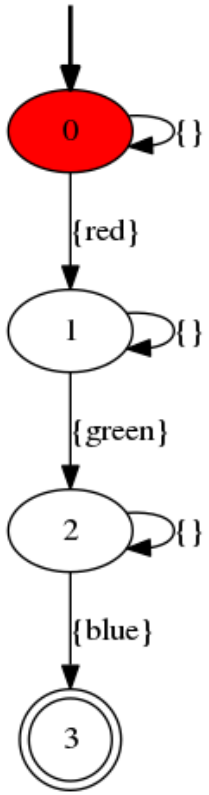


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_4, y_2, 0)$

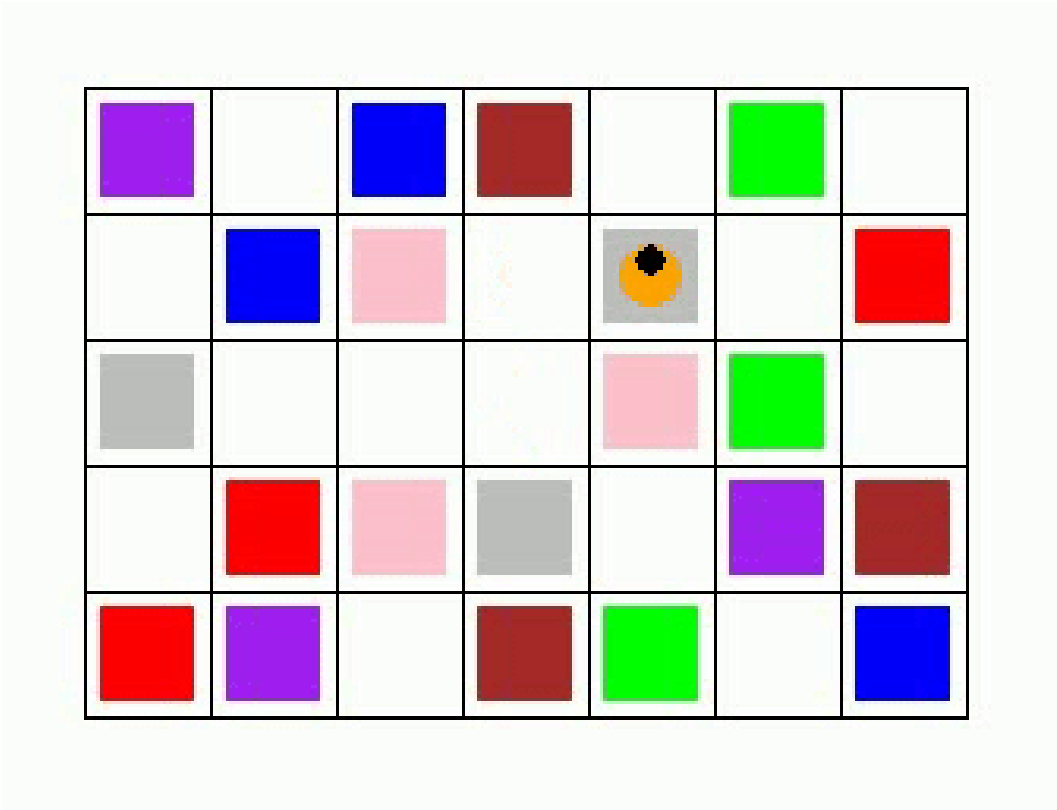


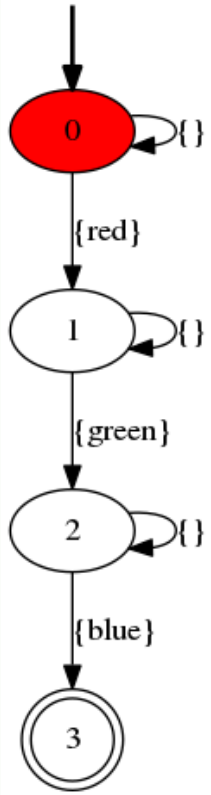


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_5, y_2, 0)$

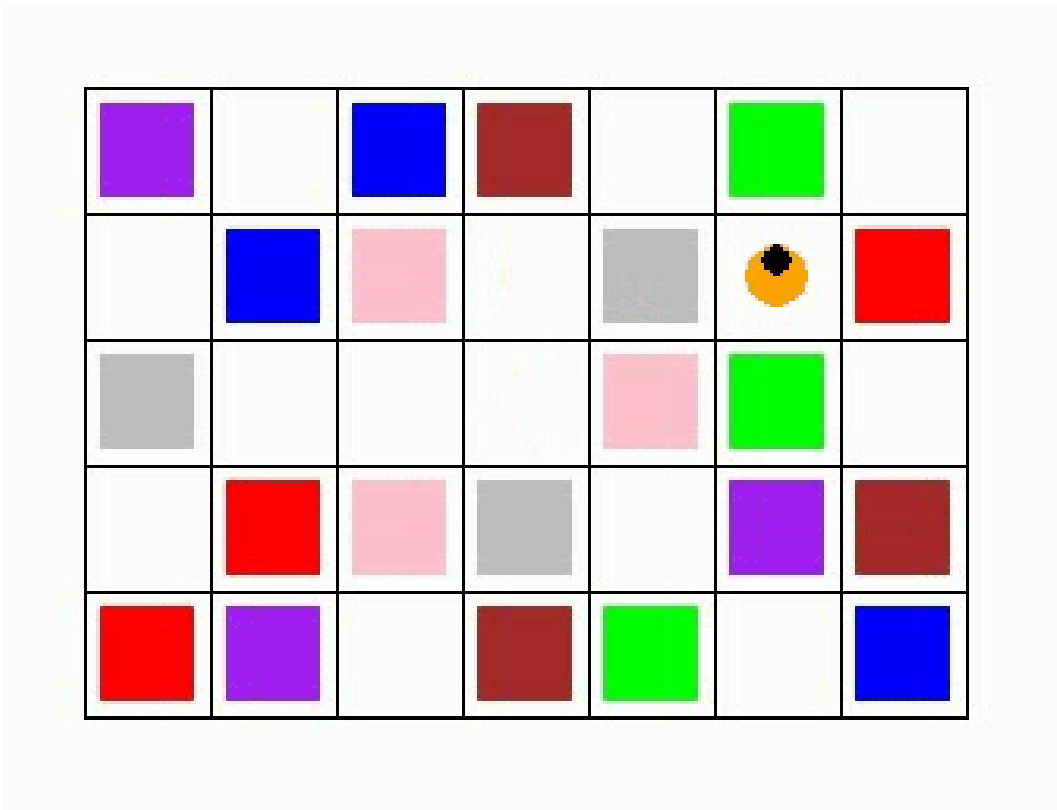


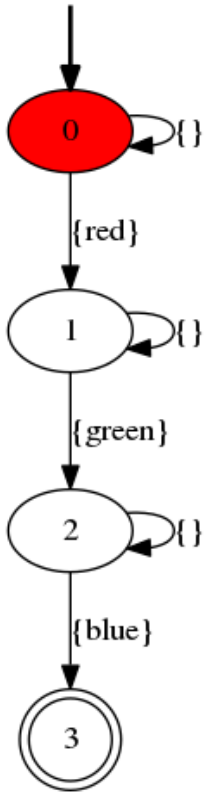


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_6, y_2, 0)$

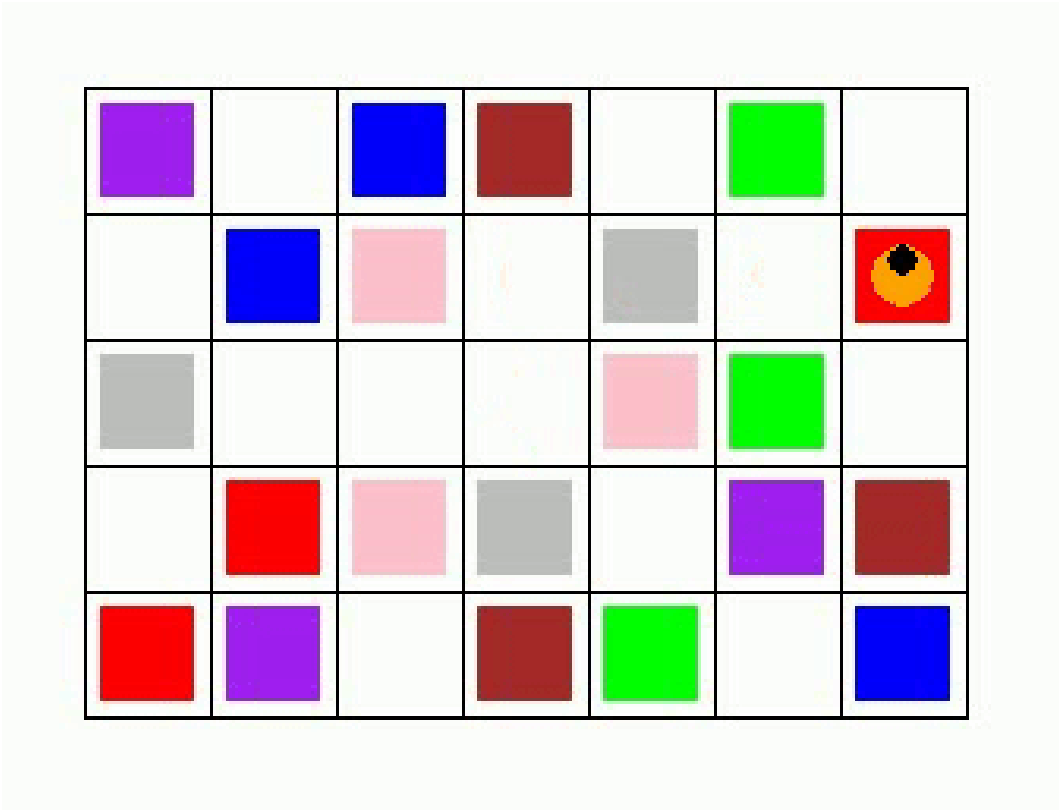


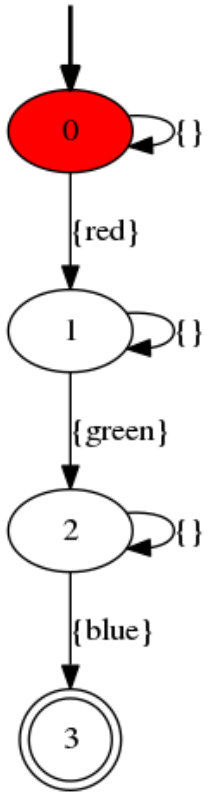


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_7, y_2, 0)$

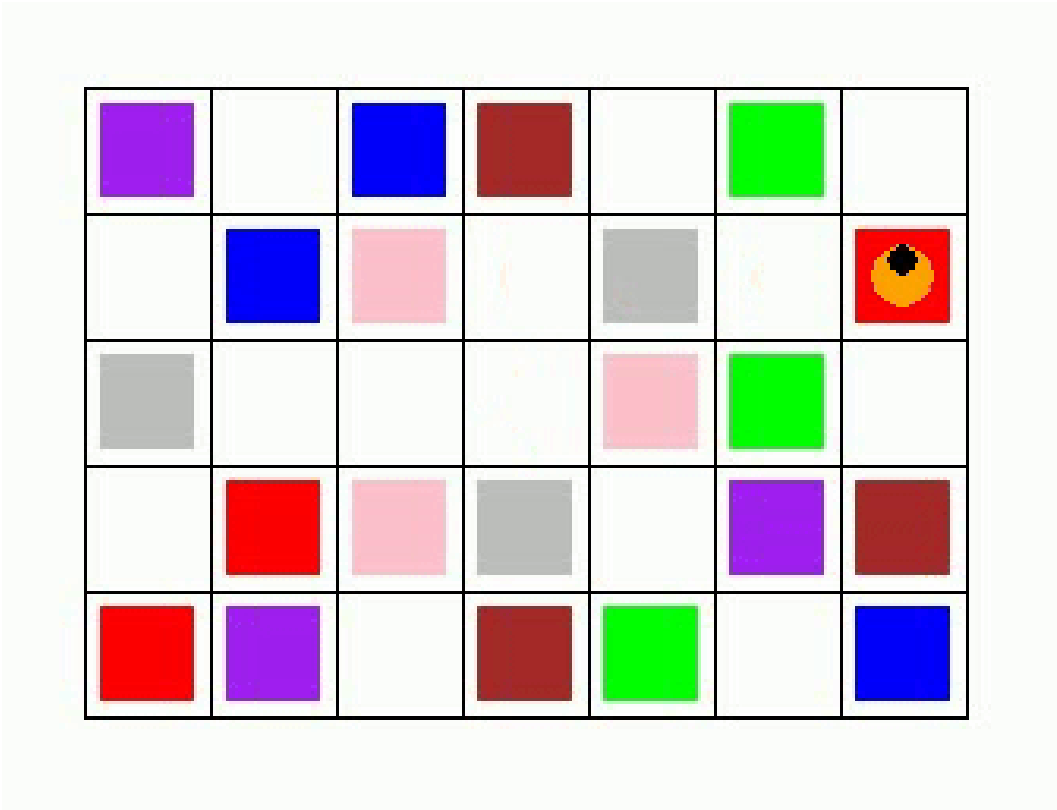


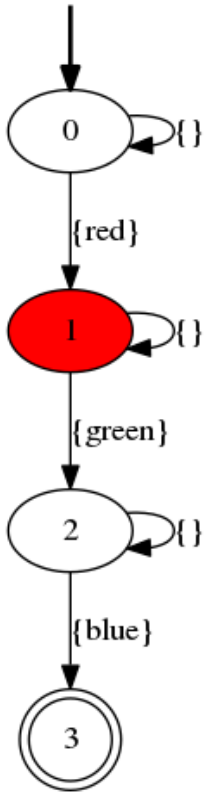


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_7, y_2, 1)$, reward $= +100$ (reward shaping)

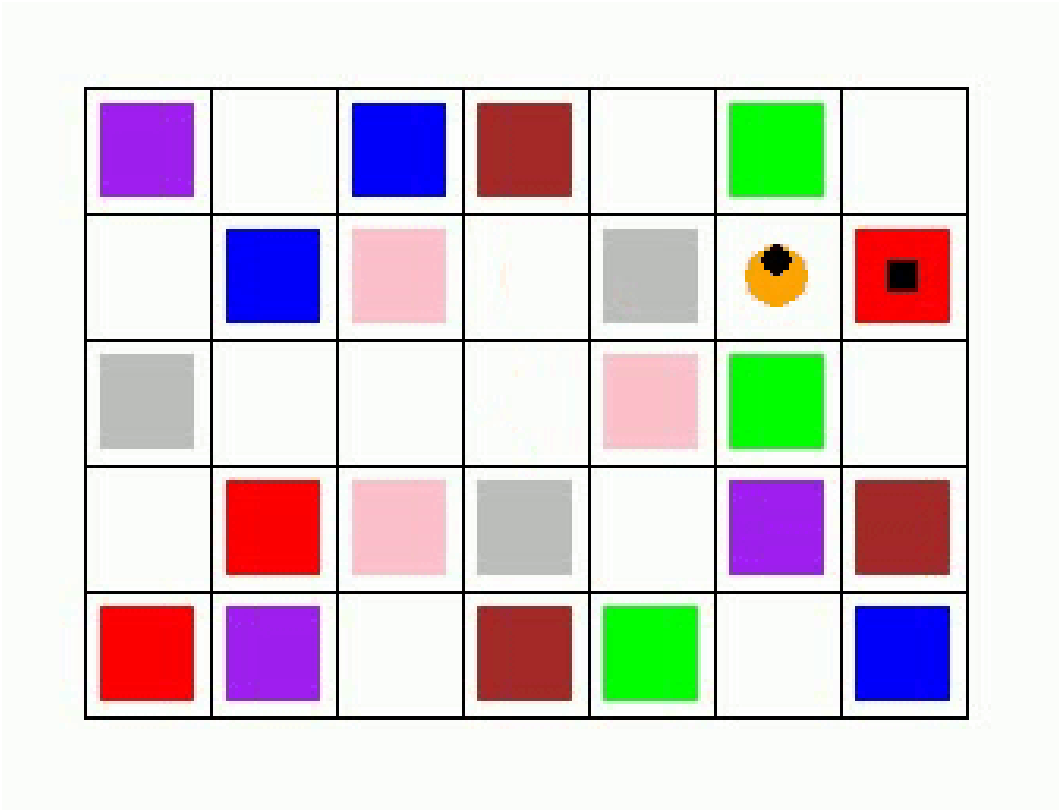


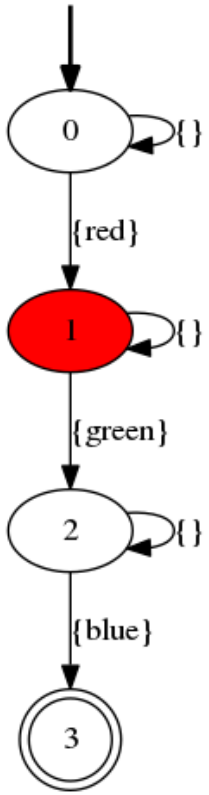


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_6, y_2, 1)$

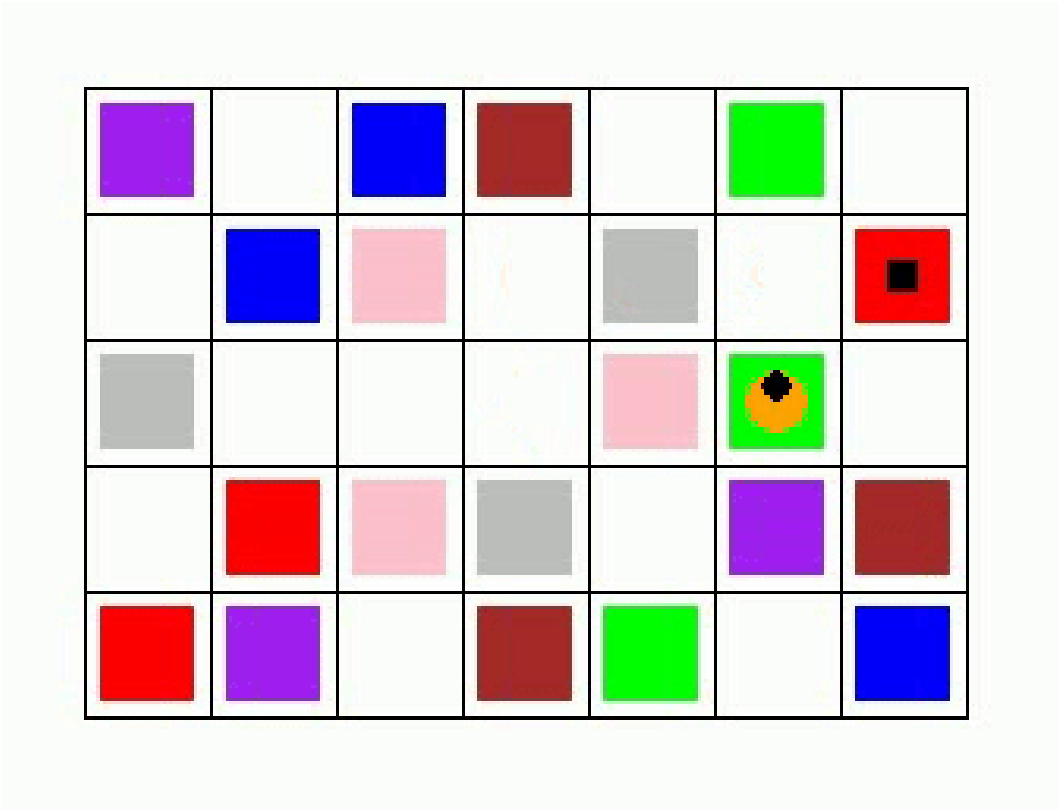


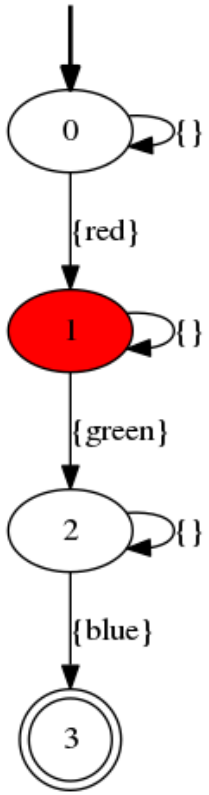


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_6, y_3, 1)$

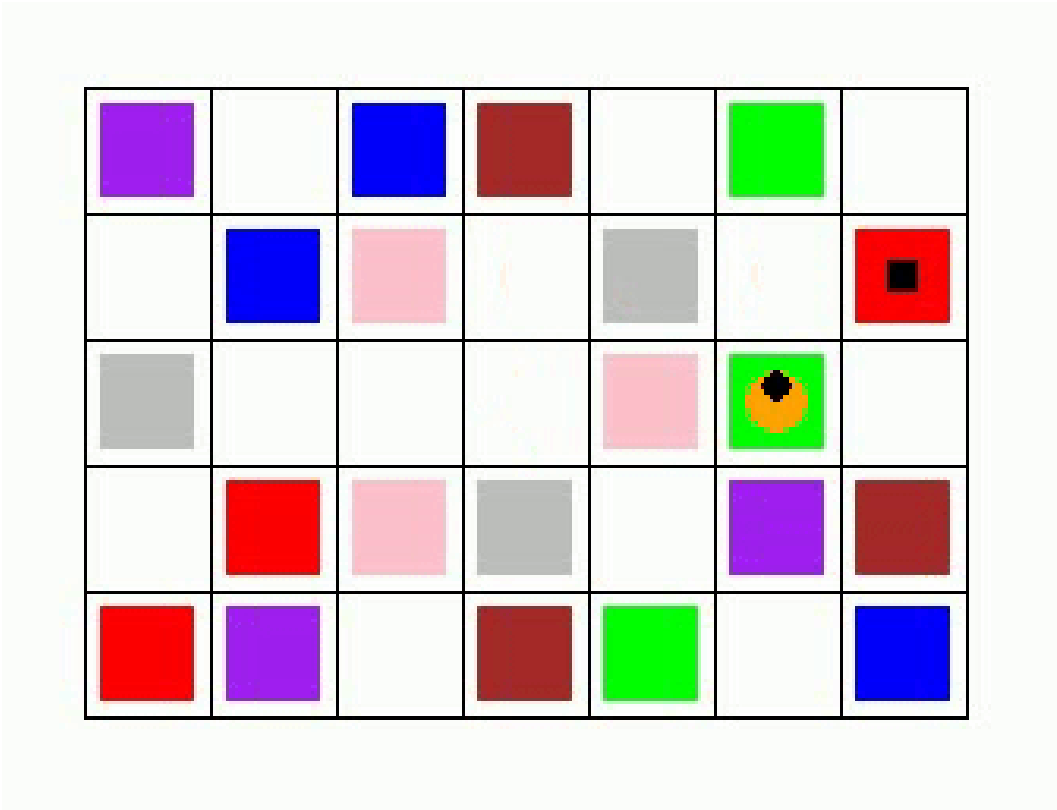


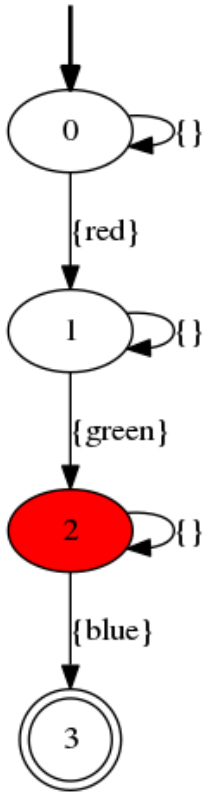


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_6, y_3, 2)$, reward $= +100$ (reward shaping)

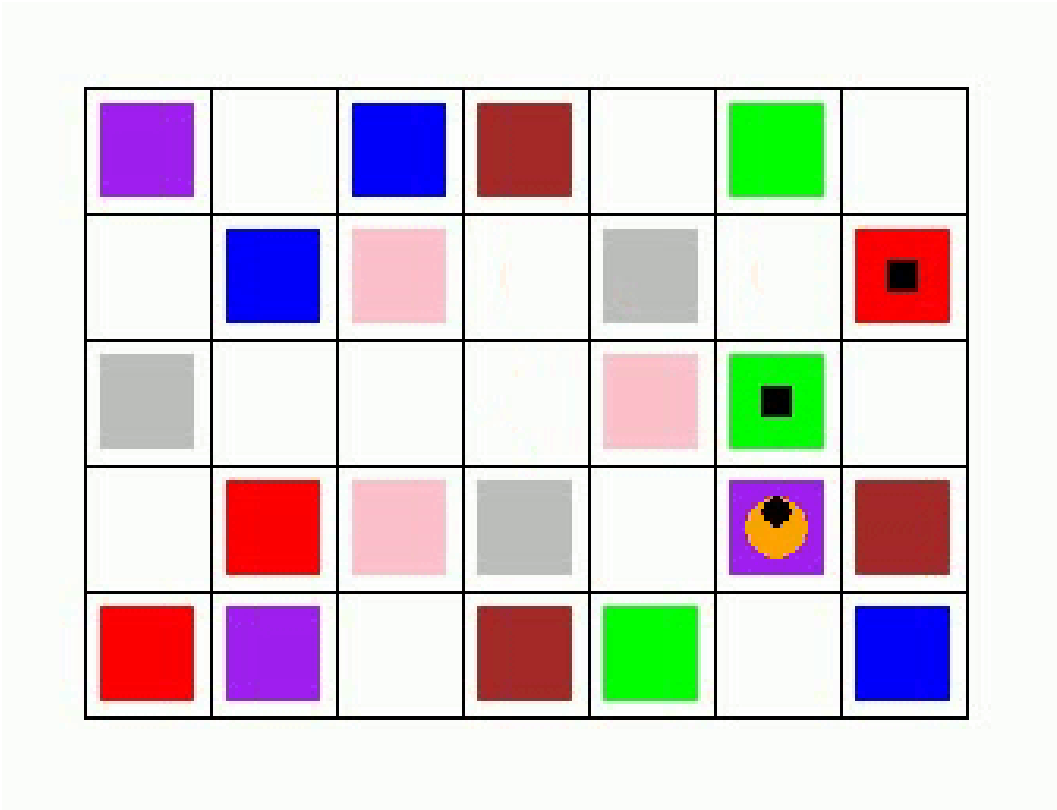


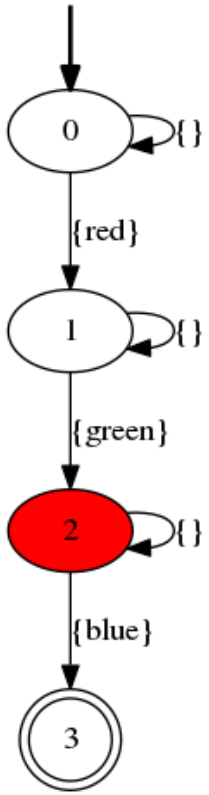


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_6, y_4, 2)$

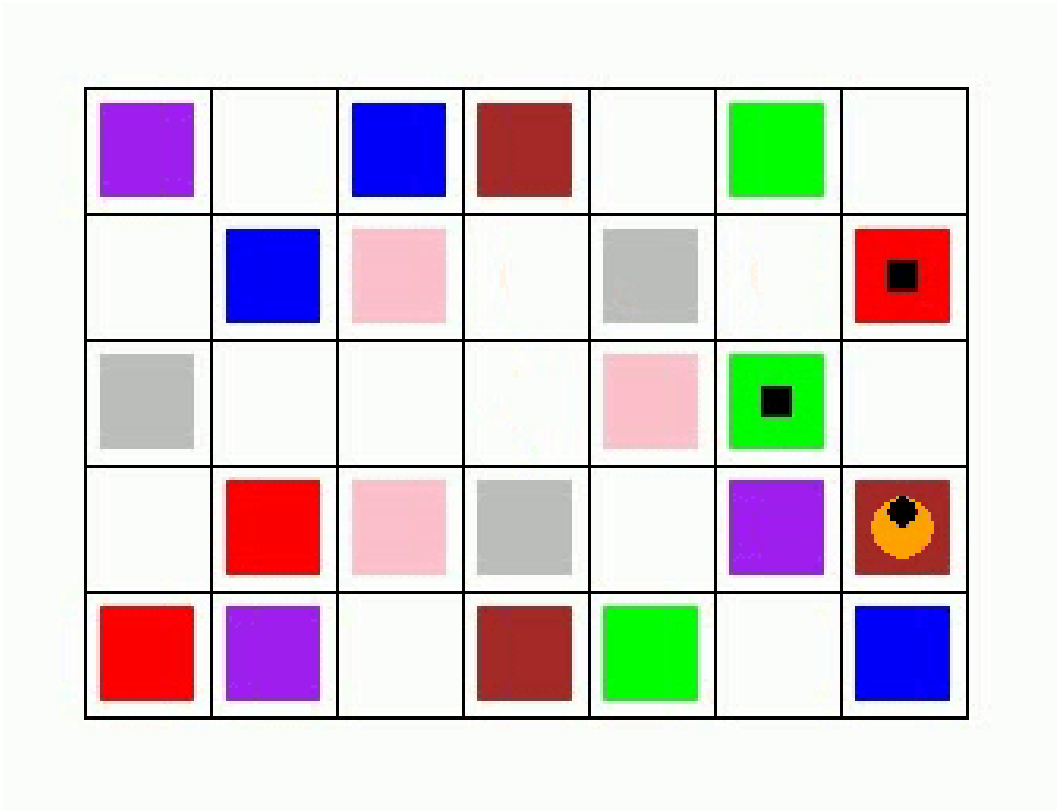


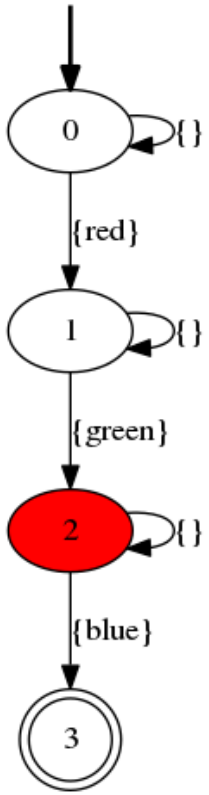


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_7, y_4, 2)$

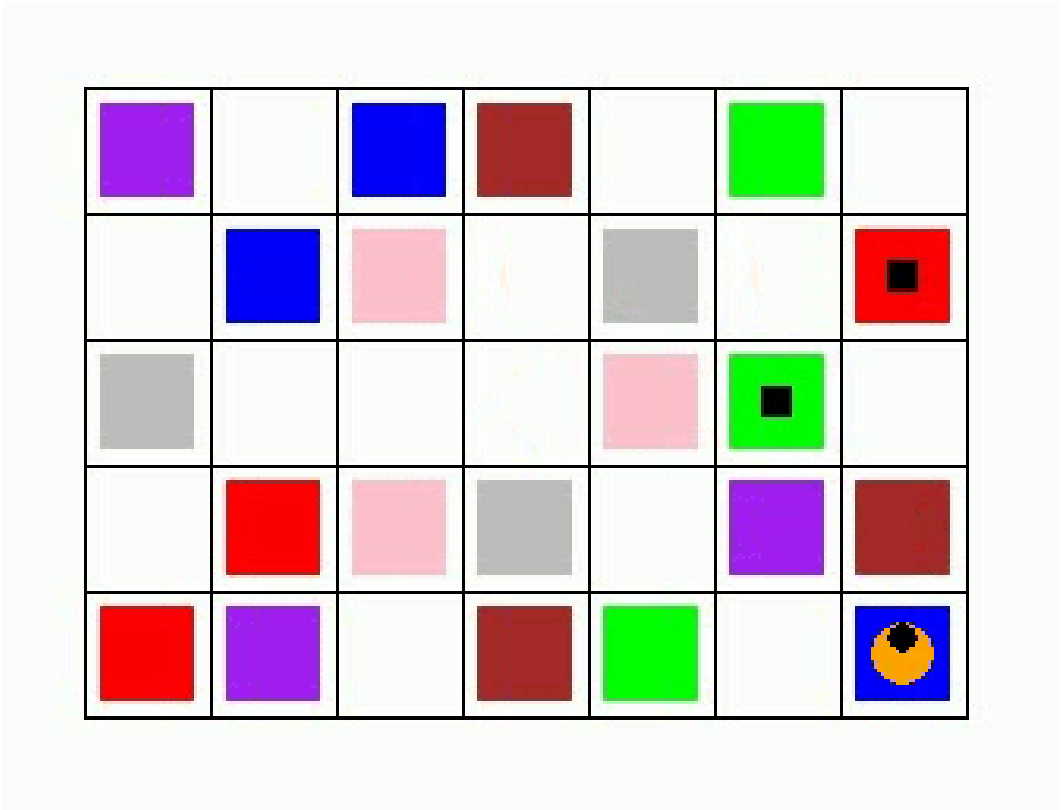


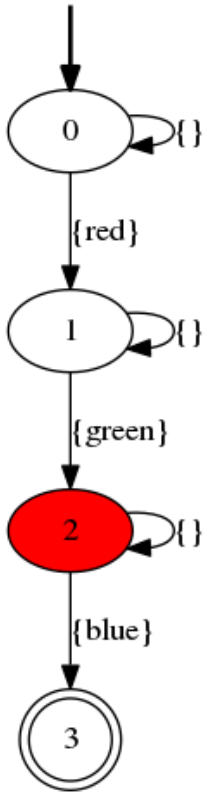


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_7, y_5, 2)$

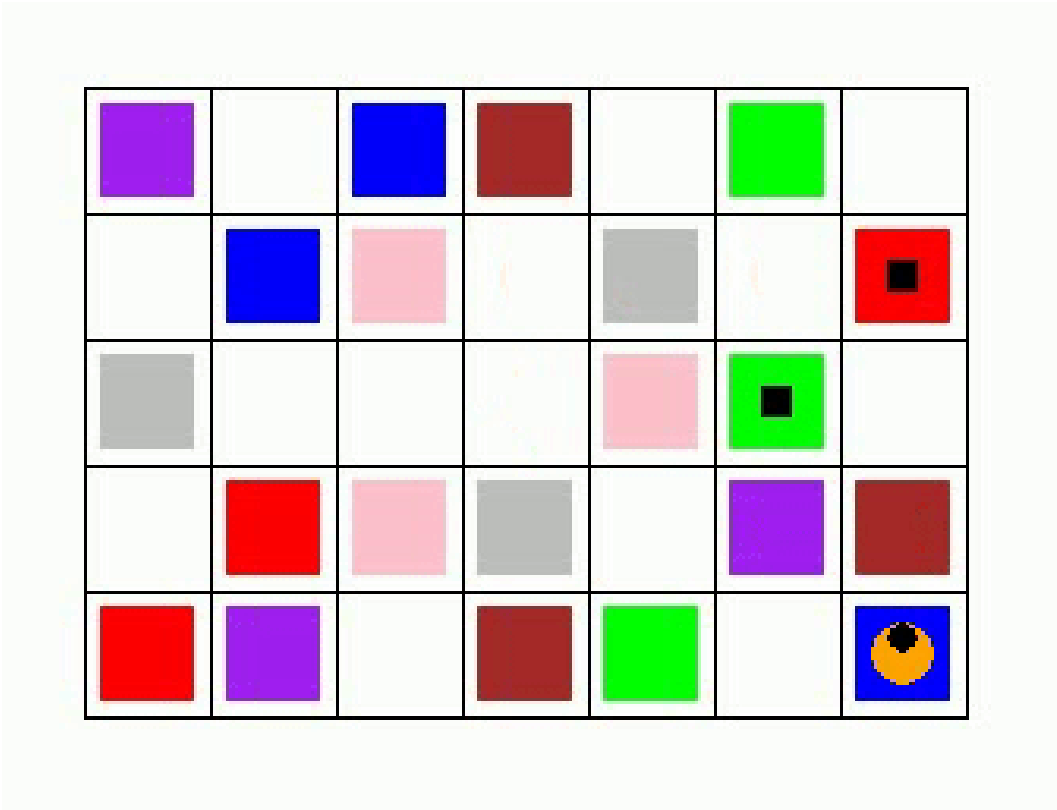


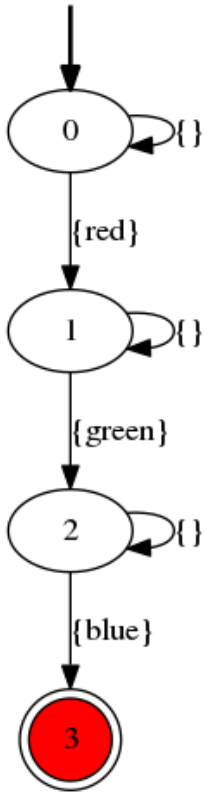


An optimal policy for SAPIENTINO

LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$, reward $r = 300$

Current state $s' = (x_7, y_5, 3)$, reward $= r - 200 = +100$





Discussion about SAPIENTINO example

Crucial remark: the agent doesn't know **anything** about fluents!

- He is aware that "something changes" (automaton transitions)
- Thanks to *reward shaping*, he knows if it is a *good/bad* transition

High expressive power:

- Many different goals and constraints, e.g. exactly two visit per color in a given order, no bad visits in the meanwhile...
- We can use also LDL_f, expressive as MSO (i.e. regular expressions)

About the two-fold representation

Notice: it seems that the agent **implicitly** learns where are the *colors*, by knowing the *coordinates*.
However, the correlation between these representations does not need to be formalized.

Denoting with $\mathcal{F}$ and $\mathcal{S}$ the set of **high-level features** and **low-level features**, respectively:

Question 1

Is $\mathcal{S}$ minimal (or "expressive enough") wrt $\mathcal{F}$?

Question 2

What is the relationship b/w $\mathcal{S}$ and $\mathcal{F}$ to be satisfied, in order to allow the agent to accomplish the high-level tasks?

About the two-fold representation

Answer: our approach makes no assumption about how the world works, hence we cannot give a general answer to these questions.

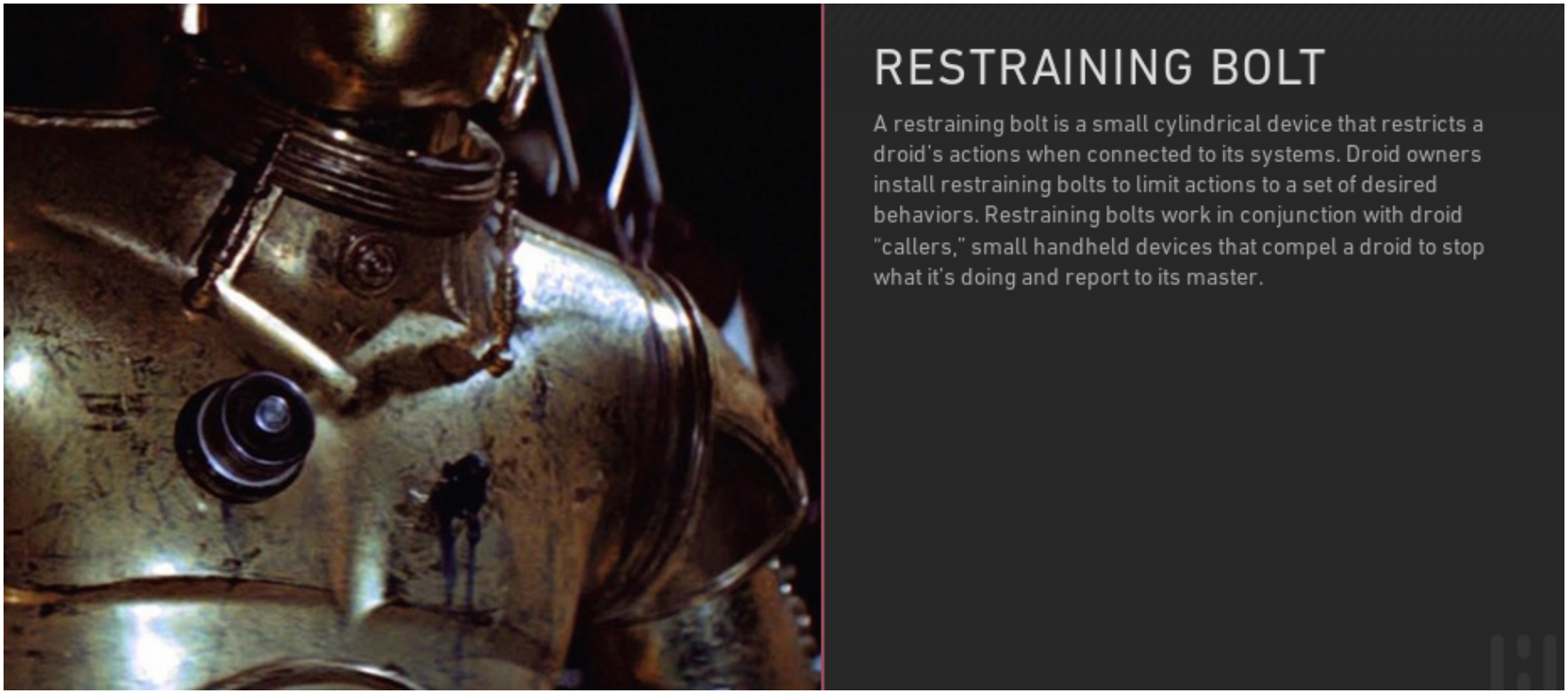
- The relationship b/w $\mathcal{S}$ and $\mathcal{F}$ is **highly environment-dependent**. One should restrict the set of worlds of interest and try to make some assumption.

Naïve solution: put in $\mathcal{S}$ as much features as the designer consider useful, and run some feature selection techniques to cope with the complexity (e.g. by using Deep Learning)
Famous examples:

- Mnih et al. 2015. *Human-level control through deep reinforcement learning*. Nature 518(7540):529–533.
- Silver et al. 2017. *Mastering the game of go without human knowledge*. Nature 550:354–359

Restraining Bolts

`https://www.starwars.com/databank/restraining-bolt`



Restraining Bolts

Two distinct representations of the world:

- one by the agent, used for interacting with the world
- one by the **authority** imposing the bolt.

With our approach, the agent must conform the restraining rules even if these are not expressed in its original representation.
The agent can learn to act while conforming (as much as possible) to the $\mathrm{LTL}_f/\mathrm{LDL}_f$ specifications.

Our work would be to apply this concept to many different applications and scenarios.