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

SAPIENZA
UNIVERSITÀ DI ROMA

Marco Favorito

Ph.D. Candidate Student 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
- Publications:
  - De Giacomo, Iocchi, Favorito, Patrizi, "Reinforcement Learning for $\text{LTL}_f/\text{LDL}_f$ Goals". arXiv preprint arXiv:1807.06333 (2018).
  - De Giacomo, Iocchi, Favorito, Patrizi, "Learning Robot Tasks with $\text{LTL}_f/\text{LDL}_f$ Goals". Cognitive Robotic Workshop 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

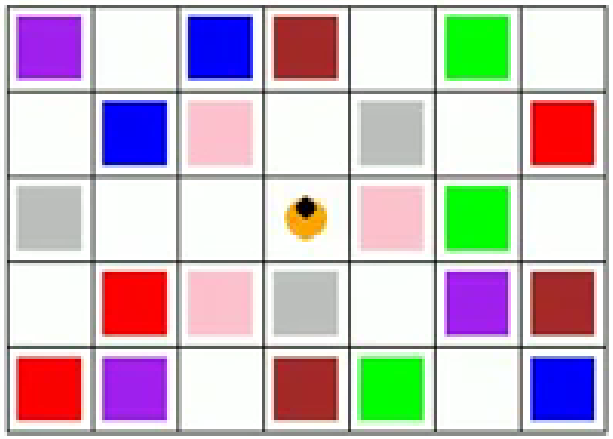
A Toy Example: SAPIENTINO

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

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

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

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



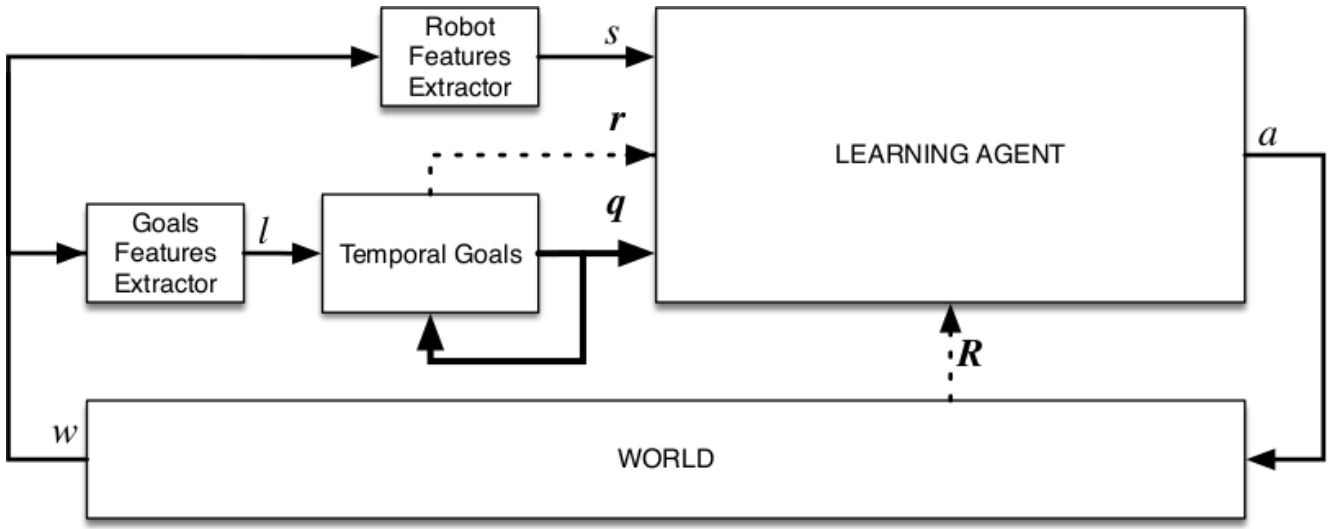
Notice that φ is a **non-Markovian** goal/reward, i.e. **depends on a sequence of transitions**

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 $\mathcal{L} = 2^{\mathcal{F}}$

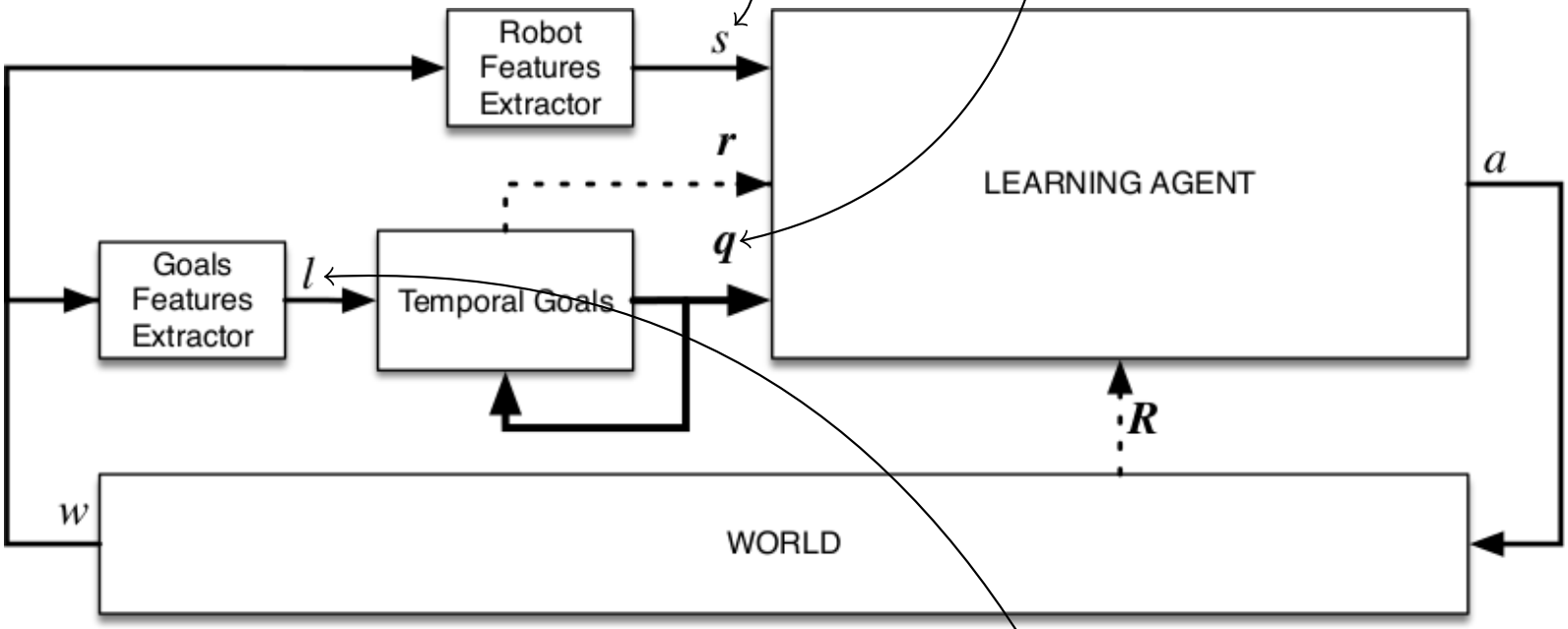
Solution: a non-Markovian policy $\rho : S^* \rightarrow A$ that is optimal wrt rewards r_i and R.



Our approach:

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

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

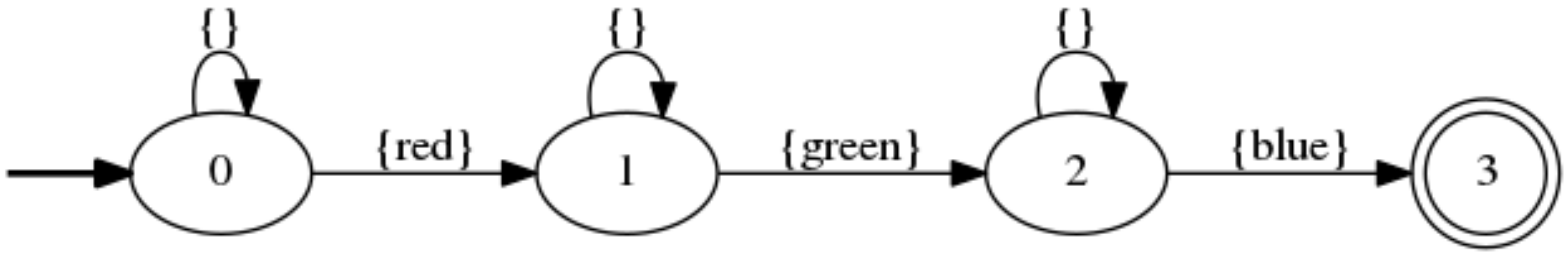


Notice: **the agent ignores the fluents** $\mathcal{L}$!

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

An optimal policy for SAPIENTINO

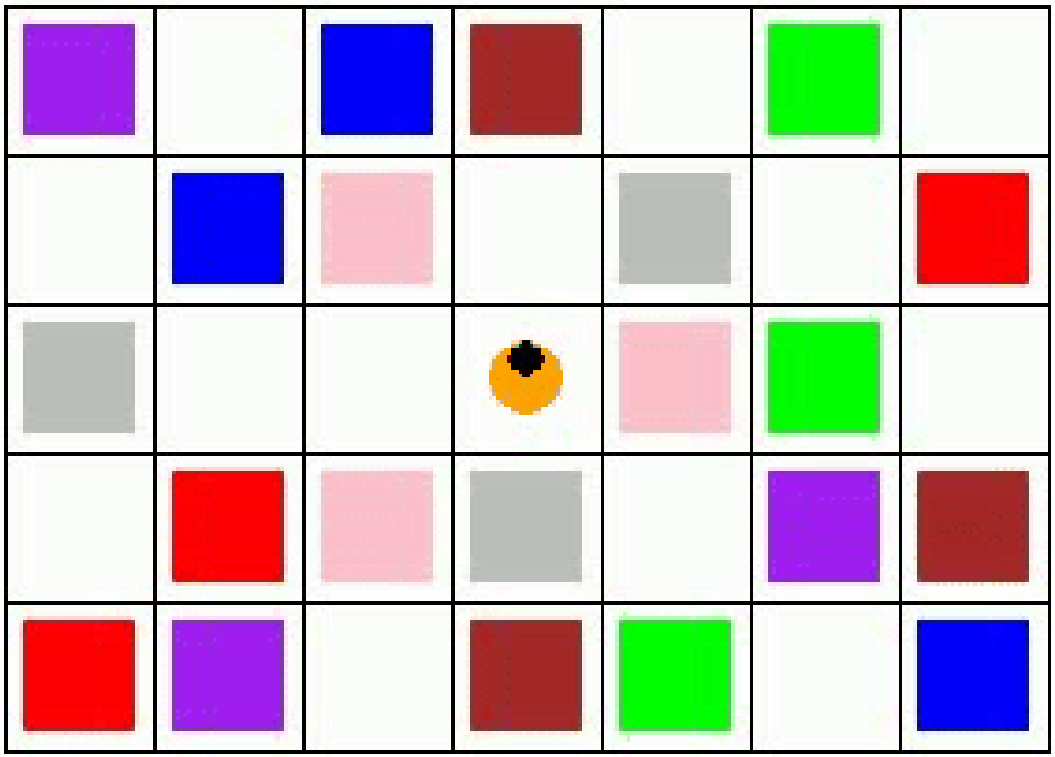
LTL_f goal $\varphi = \Diamond(red \wedge \Diamond(green \wedge \Diamond(blue)))$
The equivalent DFA is (roughly):

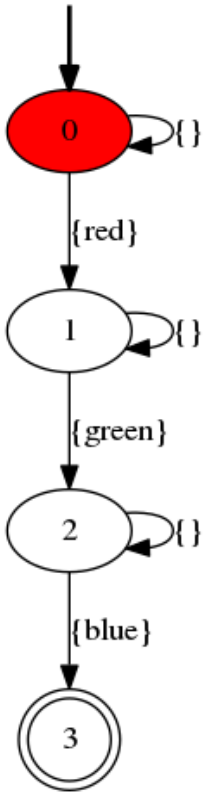


An optimal policy for SAPIENTINO

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

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

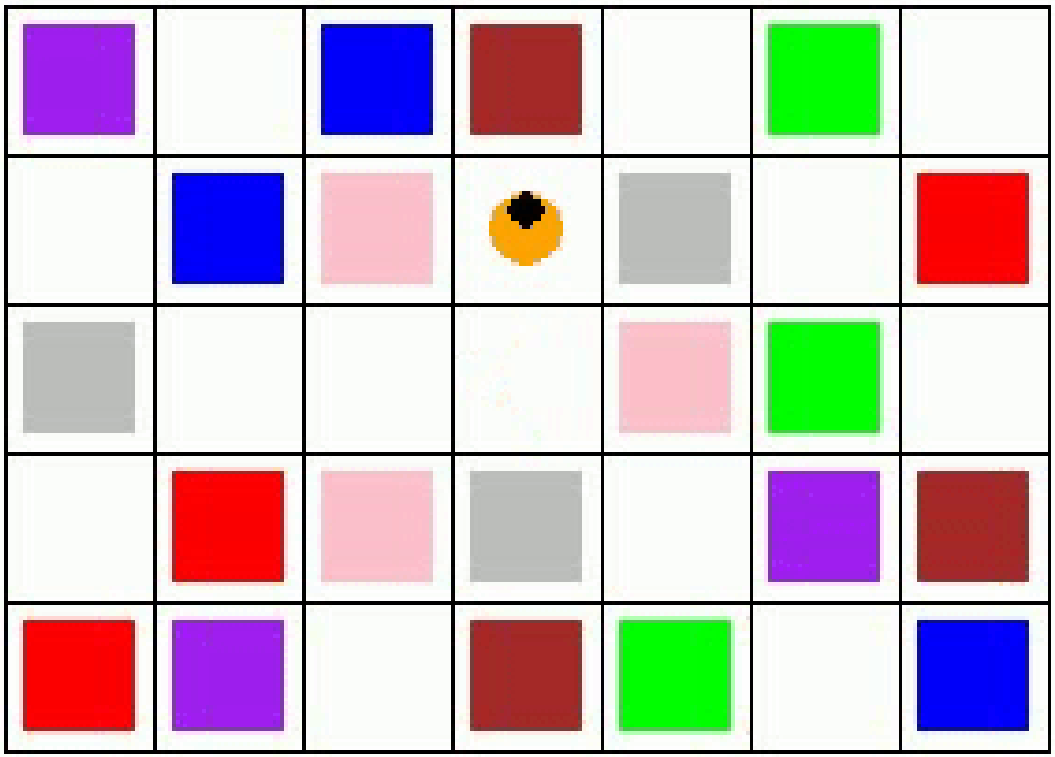


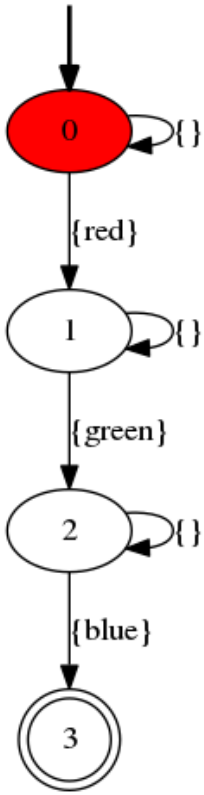


An optimal policy for SAPIENTINO

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

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

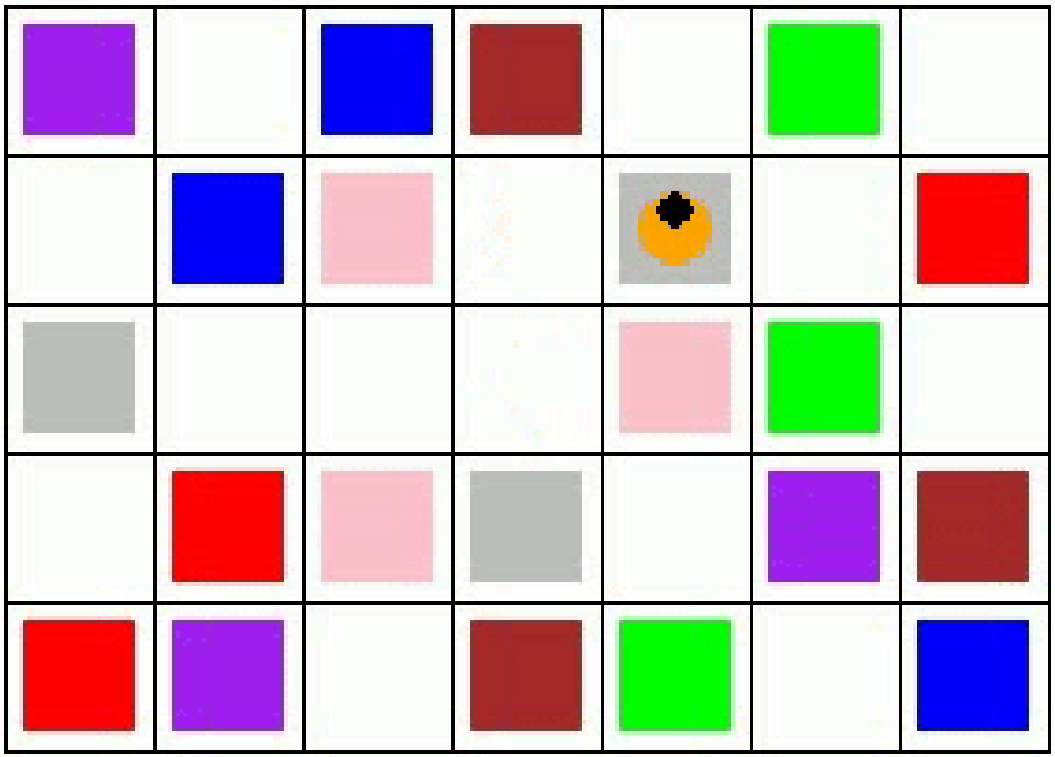


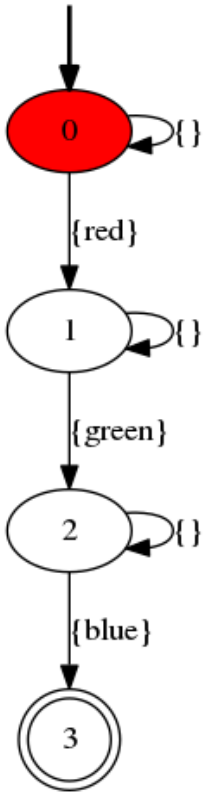


An optimal policy for SAPIENTINO

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

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

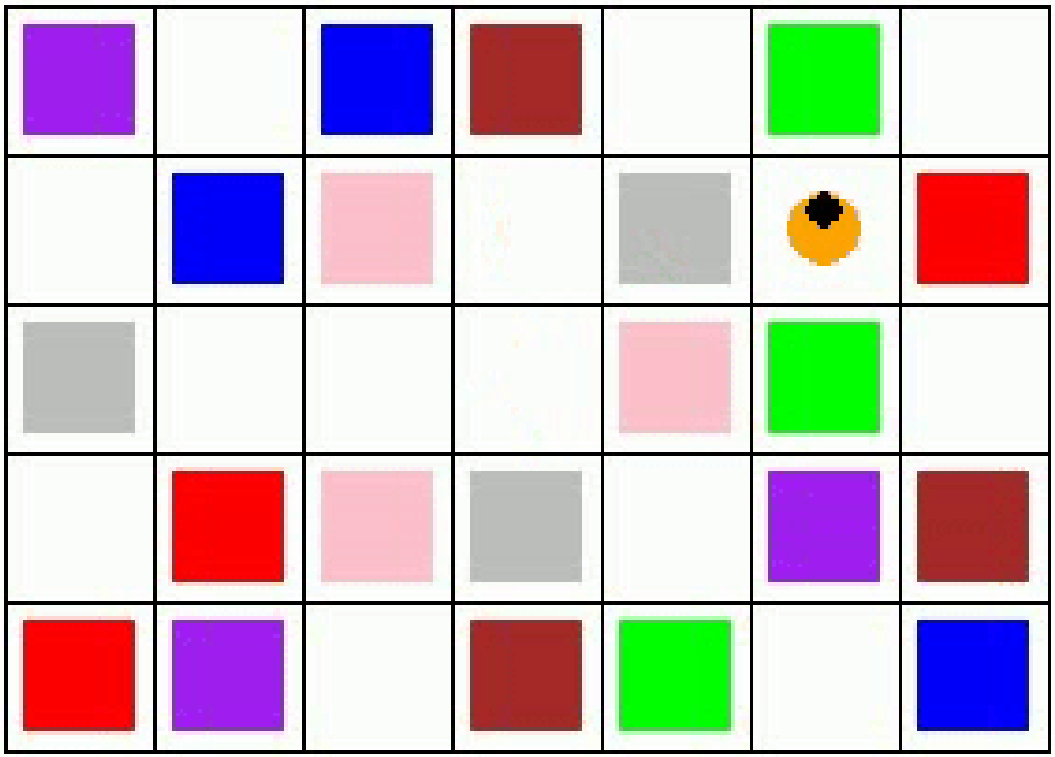


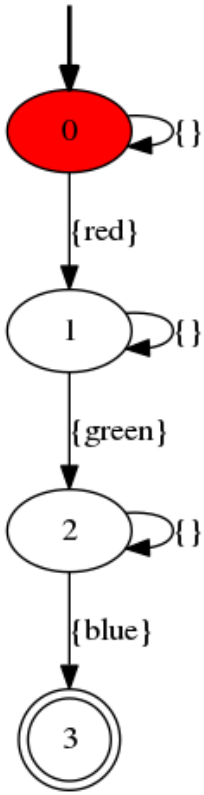


An optimal policy for SAPIENTINO

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

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

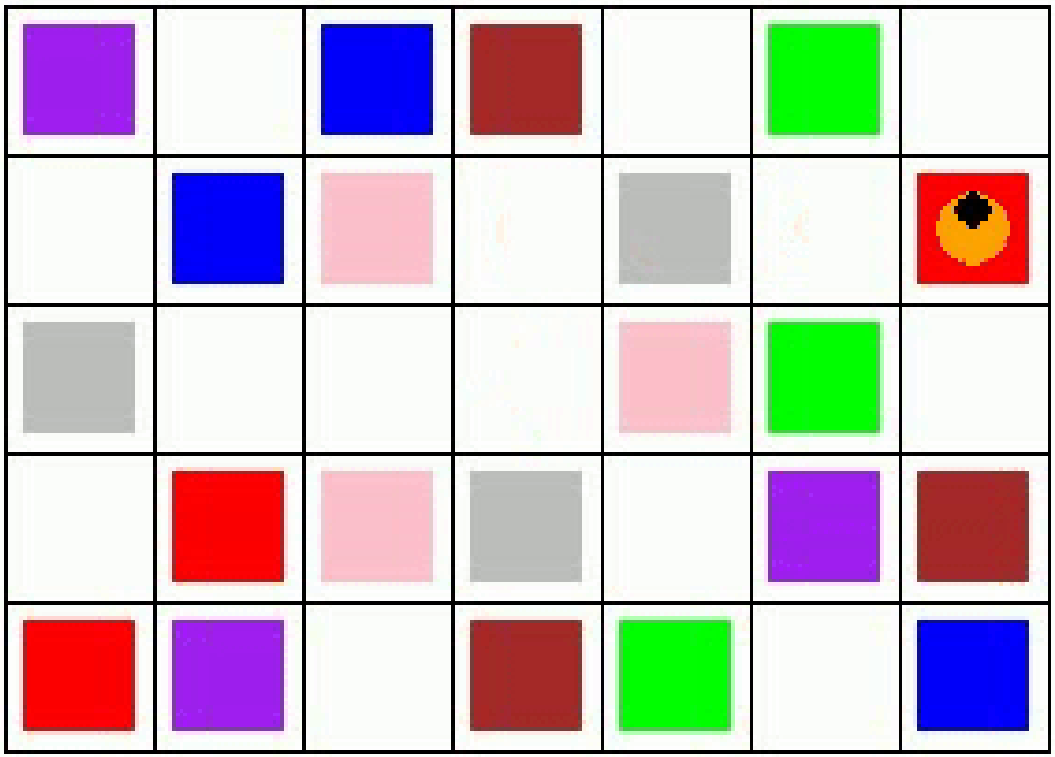


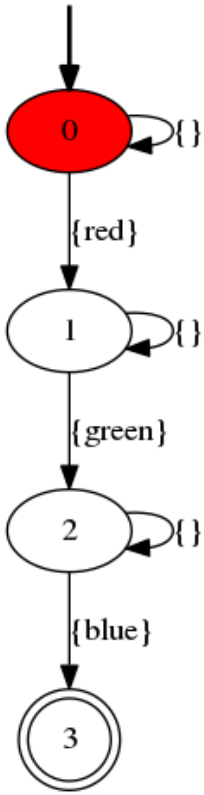


An optimal policy for SAPIENTINO

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

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

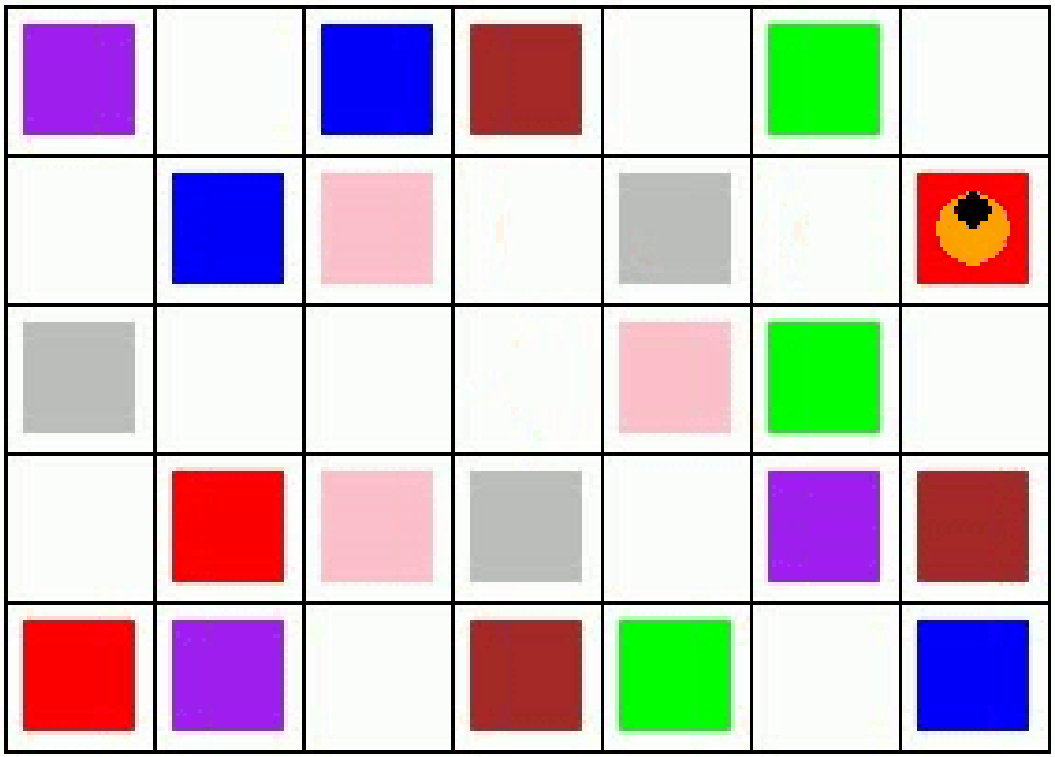


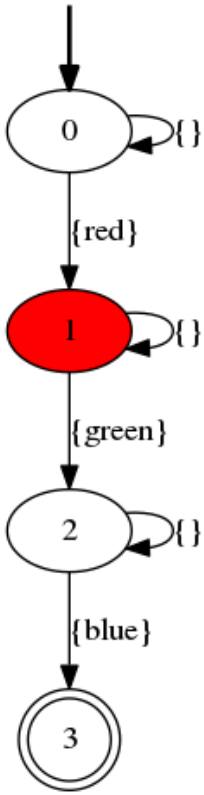


An optimal policy for SAPIENTINO

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

Current state $s' = (x_7, y_2, 1)$, Action $a = BIP$, reward $= +100$

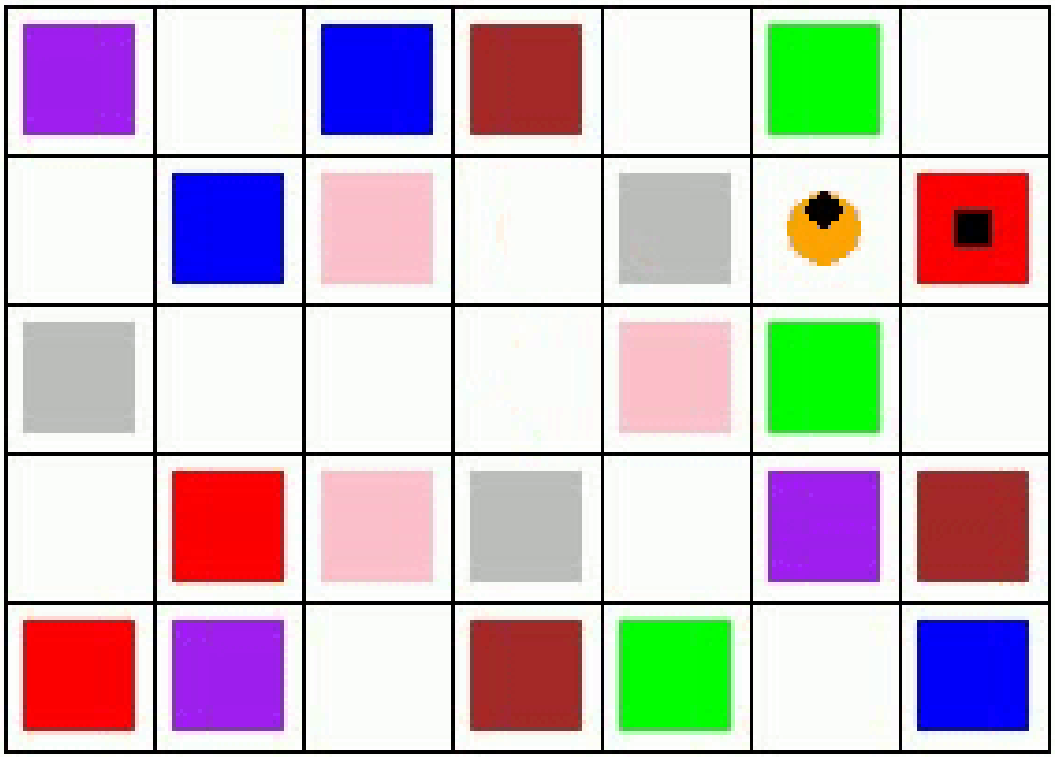


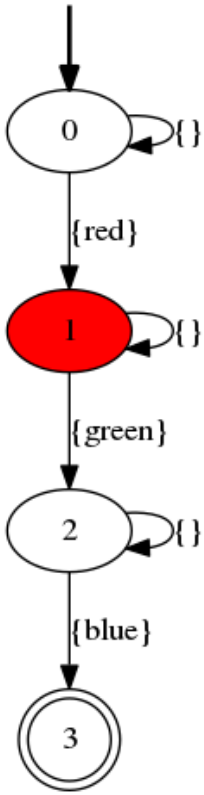


An optimal policy for SAPIENTINO

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

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

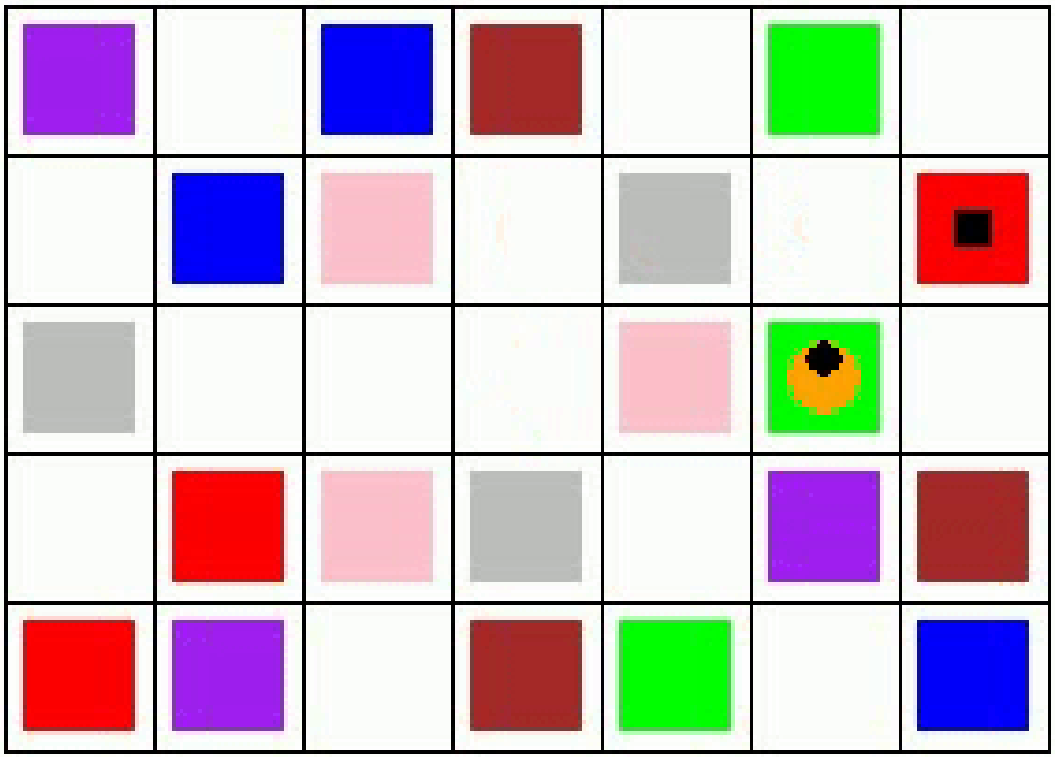


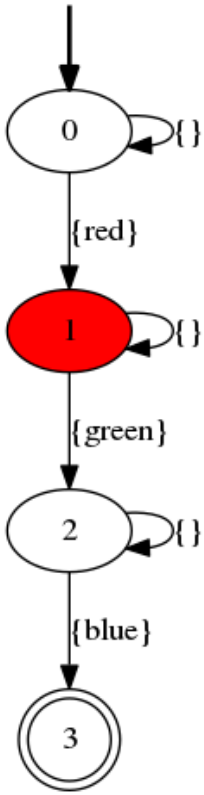


An optimal policy for SAPIENTINO

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

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

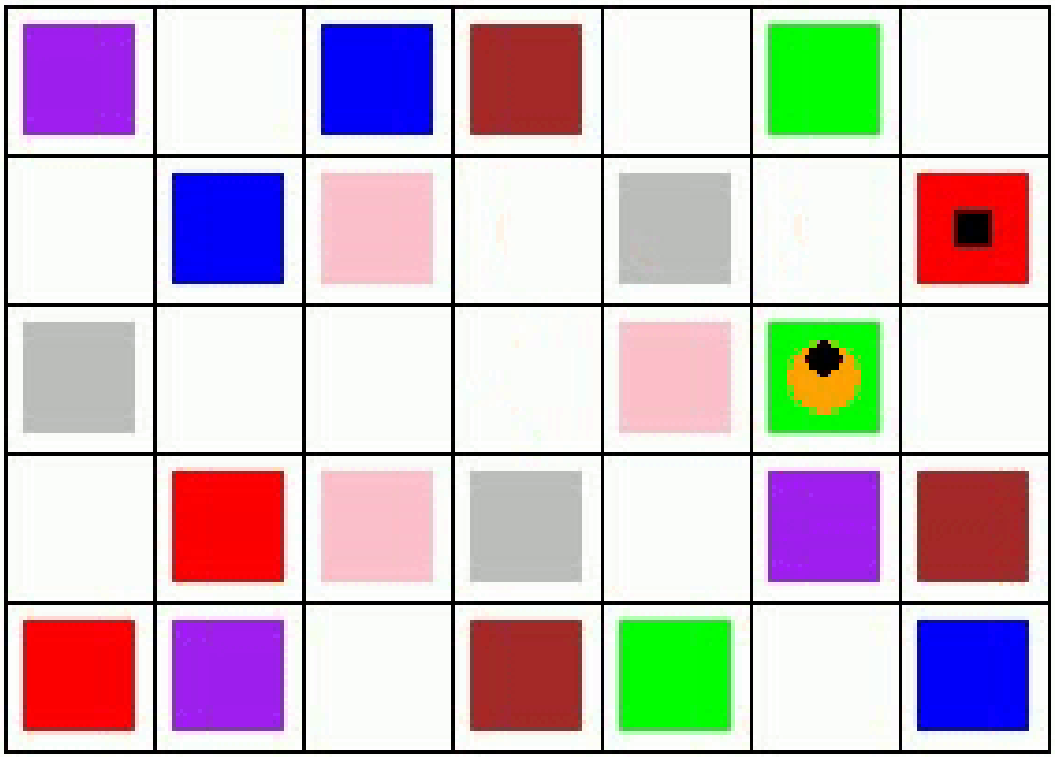


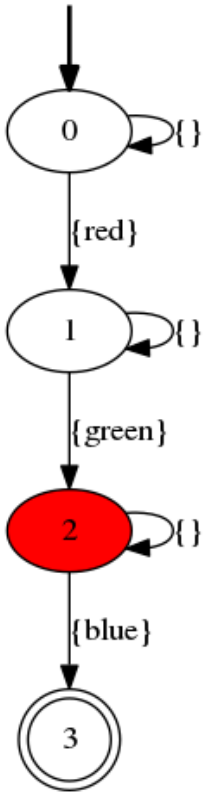


An optimal policy for SAPIENTINO

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

Current state $s' = (x_6, y_3, 2)$, Action $a = BIP$, reward $= +100$

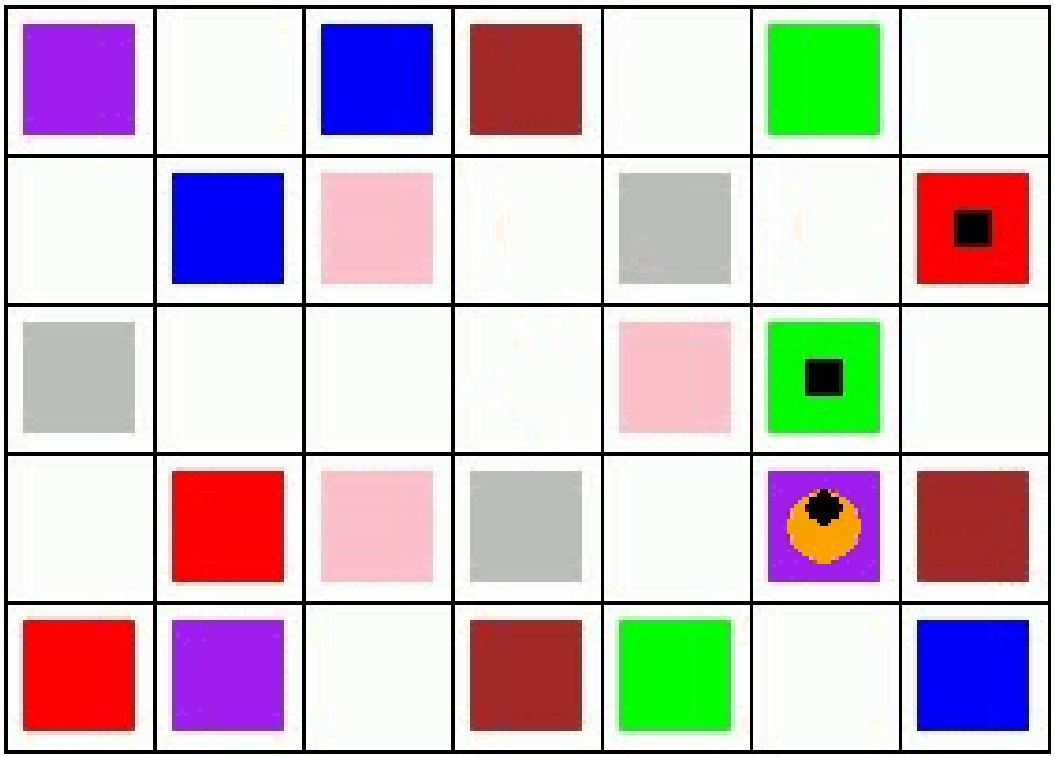


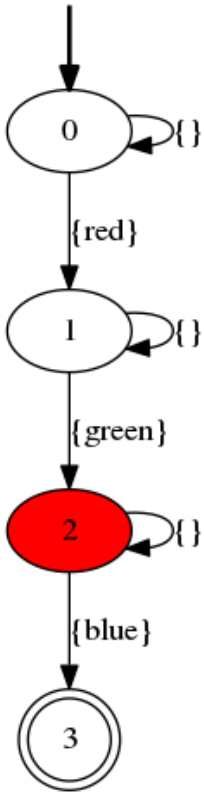


An optimal policy for SAPIENTINO

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

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

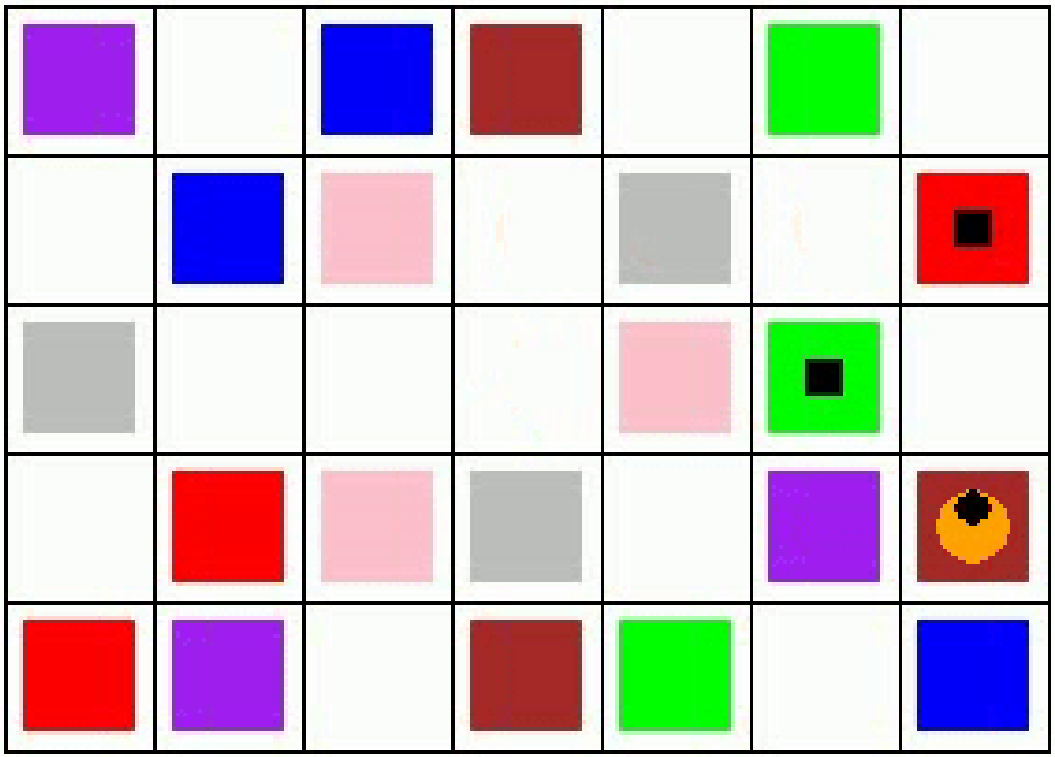


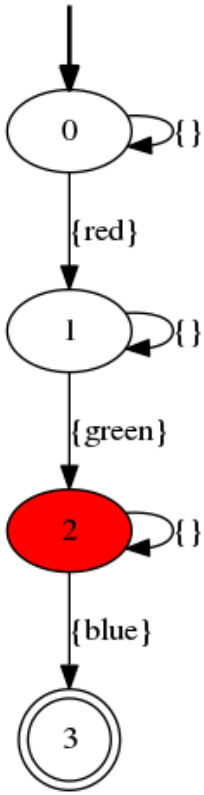


An optimal policy for SAPIENTINO

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

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

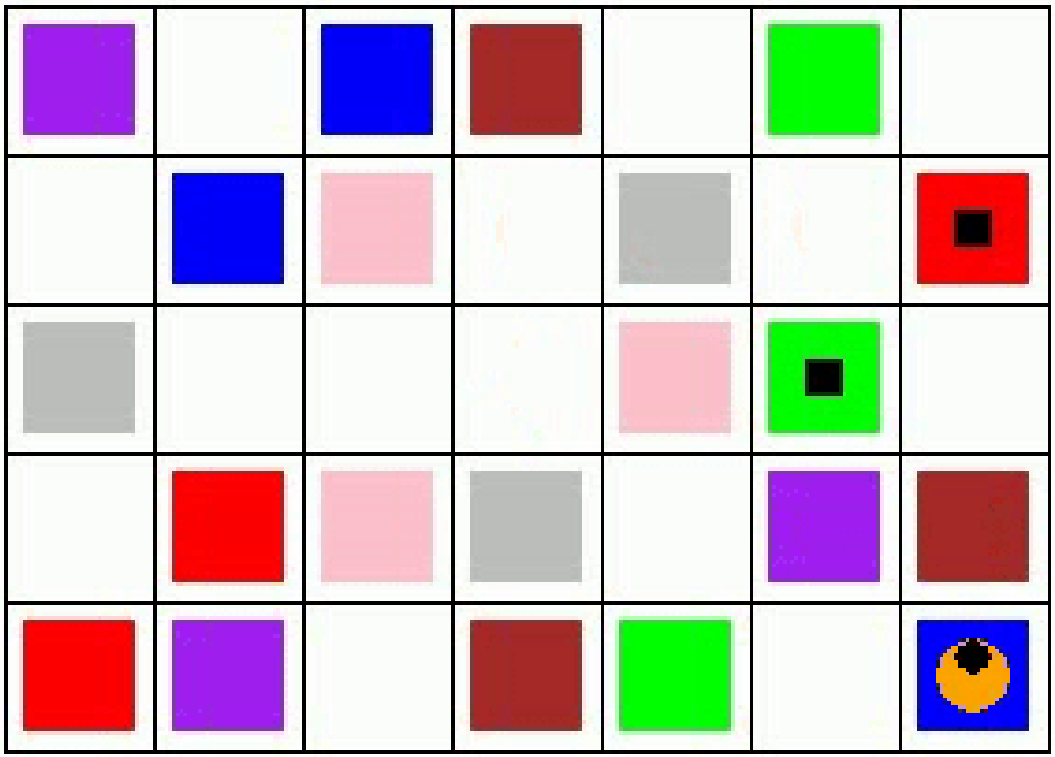


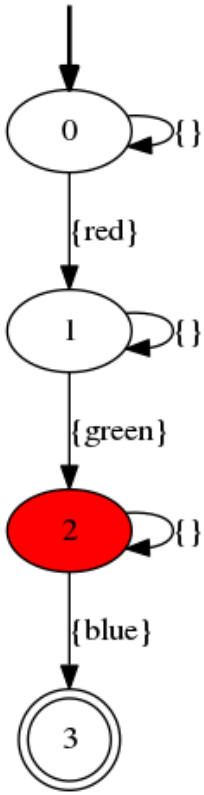


An optimal policy for SAPIENTINO

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

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

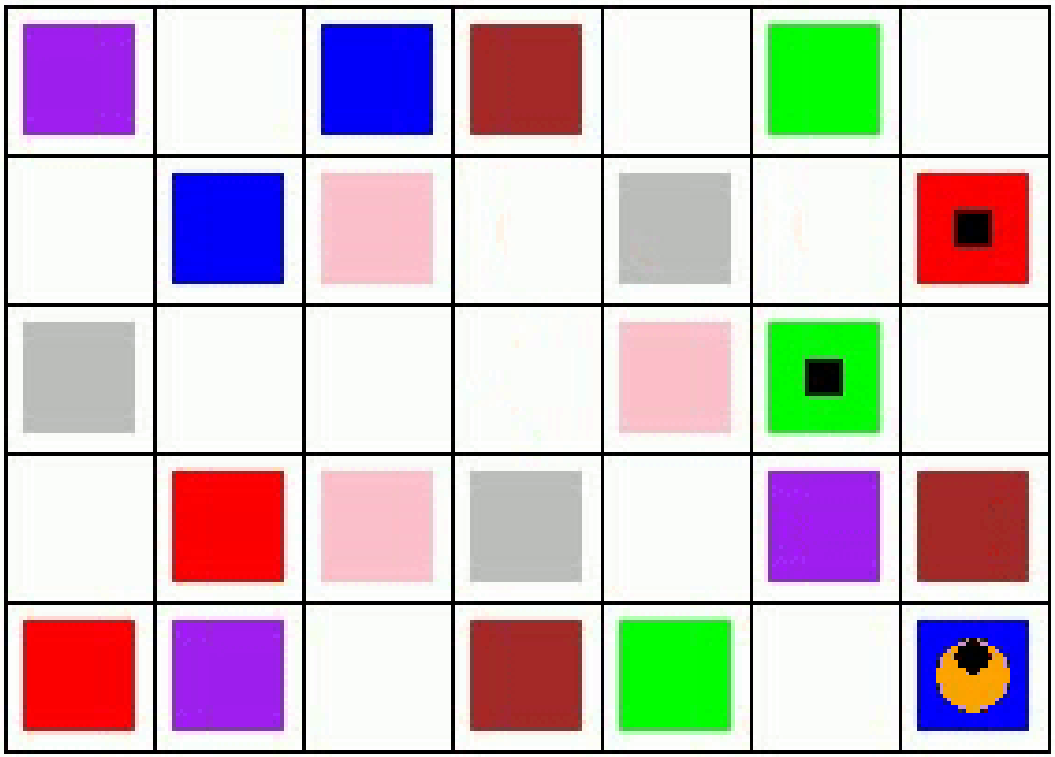


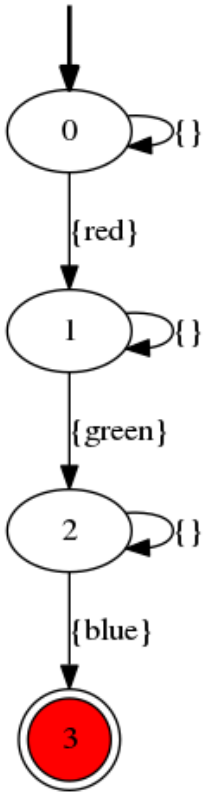


An optimal policy for SAPIENTINO

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

Current state $s' = (x_7, y_5, 3)$, Action $a = BIP$, reward $= +100$





Discussion about SAPIENTINO example

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

- Great! *Separation of concern* and *modularity*
- He is aware that "something changes" (automata 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*.

In SAPIENTINO, the agent can learn where are colors because each color can be associated to different coordinates.

In general, however, the correlation between $\mathcal{L}$ and $\mathcal{S}$ does not need to be formalized.

- $\mathcal{L}$ is the set of **high-level features**
- $\mathcal{S}$ is the set of **low-level features**

About the two-fold representation

Question 1

Is a $\mathcal{S}$ minimal (and "expressive enough") wrt a $\mathcal{L}$?

Question 2

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

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 $\mathcal{L}$ is **highly environment-dependent**. One should restrict the set of worlds of interest and try to make some assumption.
- The ideal solution would be to have a constructive method that, given $\mathcal{L}$, allow us to build a proper $\mathcal{S}$.

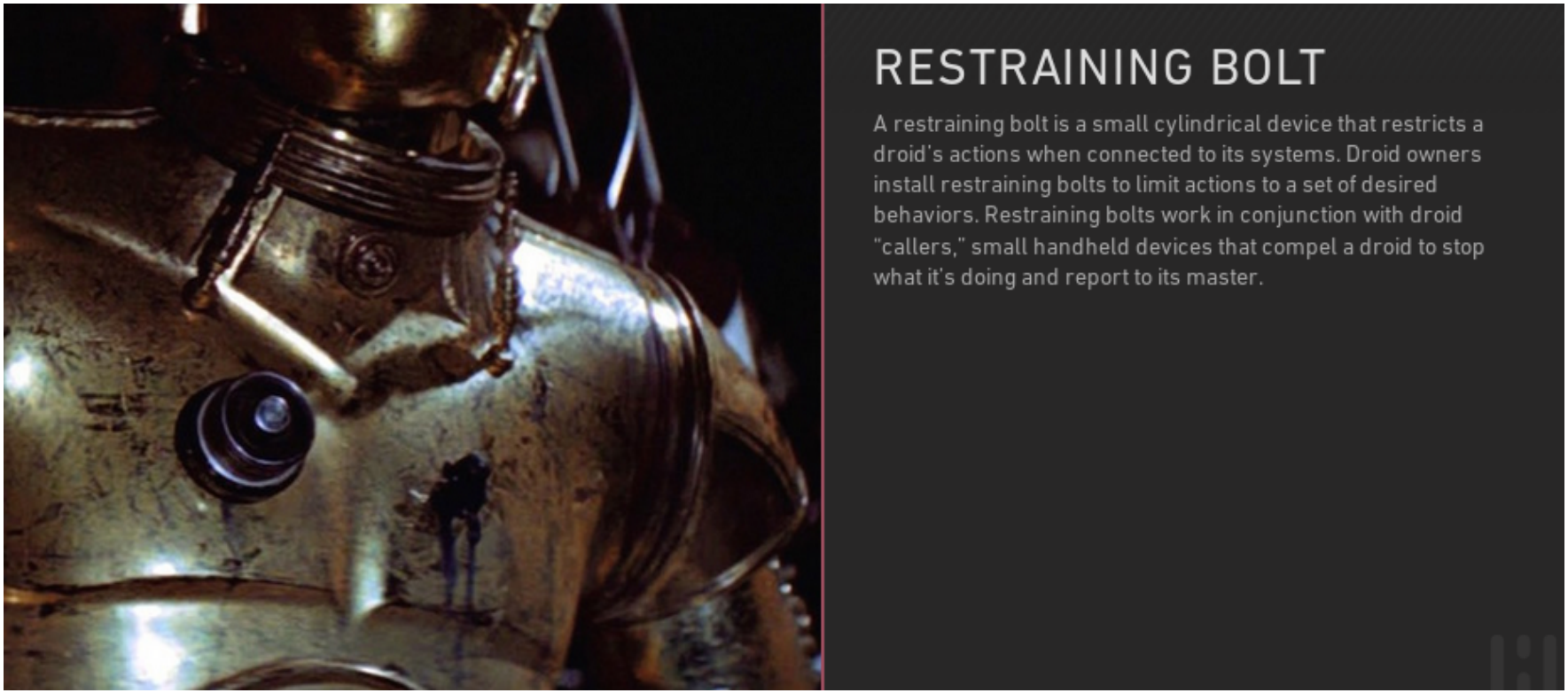
About the two-fold representation

Naïve solution: put in $\mathcal{S}$ as many 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: AI safety

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



Restraining Bolts: AI safety

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.

Restraining Bolts: AI safety

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

The concept of safety guarantees is a hot topic and is of crucial importance in AI.

- Hadfield-Menell, Dylan, et al. *The off-switch game.* arXiv preprint arXiv:1611.08219 (2016).
- Amodei, Dario, et al. *Concrete problems in AI safety.* arXiv preprint arXiv:1606.06565 (2016).

Ad-hoc RL algorithms

One of the advantages of our method is to rely on standard RL algorithms.

However, it might be the case that ad-hoc algorithms leveraging the structure of our solution yield better performances.

Ad-hoc RL algorithms: Options

- Sutton et al., 1999. *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning.* Artificial intelligence, 112(1-2), 181-211

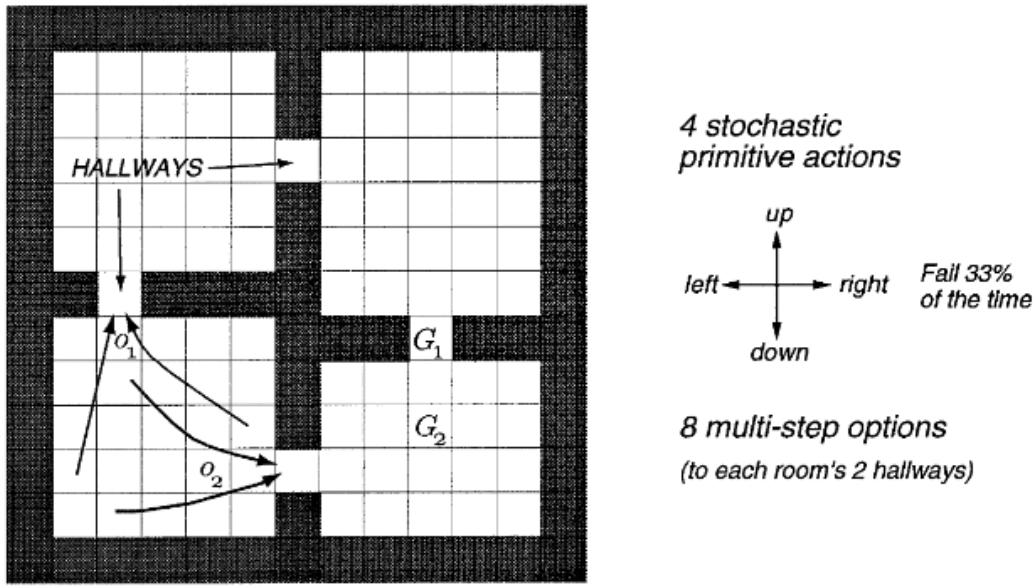


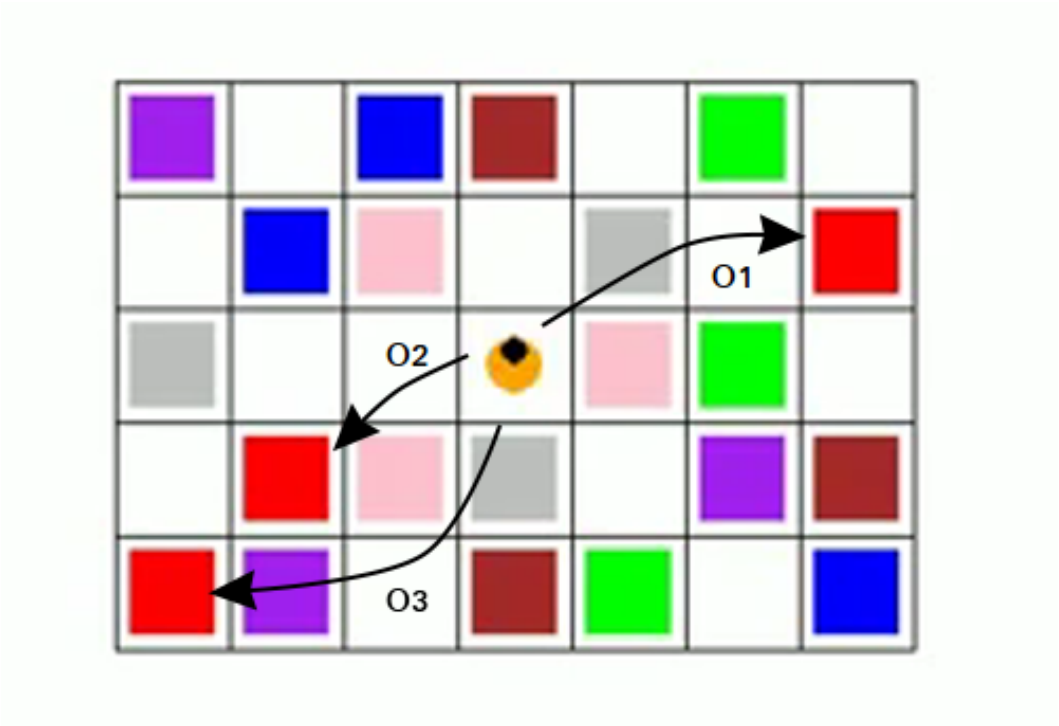
Fig. 2. The rooms example is a gridworld environment with stochastic cell-to-cell actions and room-to-room hallway options. Two of the hallway options are suggested by the arrows labeled o_1 and o_2. The labels G_1 and G_2 indicate two locations used as goals in experiments described in the text.

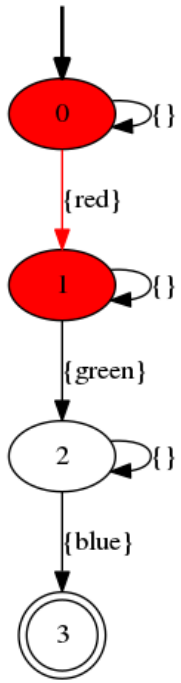
Ad-hoc RL algorithms: Options

In our framework, **options might be associated with automata transition(s), or even the entire automata**.

E.g. in the previous example with SAPIENTINO:

- reach any *red* cell:





Ad-hoc RL algorithms: Options

Problem: we shouldn't specify by hand the needed options:

- it's not scalable
- it depends on choices about $\mathcal{S}$ and $\mathcal{L}$

...But we can leverage the automaton (and reward shaping) to help the agent to learn options.
Further considerations should be made to assess the applicability

Similar work:

- Stolle and Precup. 2002. *Learning options in reinforcement learning.* International Symposium on abstraction, reformulation, and approximation. Springer, Berlin, Heidelberg.

Ad-hoc RL algorithms: Improve Exploration

- Improve exploration (e.g. O-MCTS. M. de Waard et al., 2016). In other words, focus on the next good DFA state.

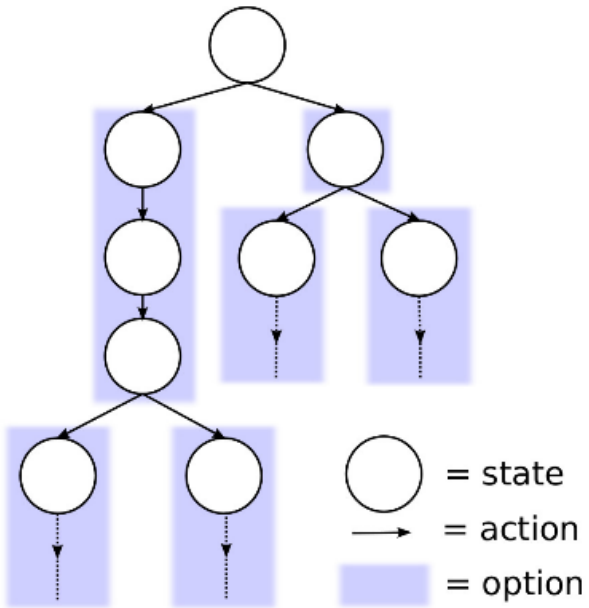
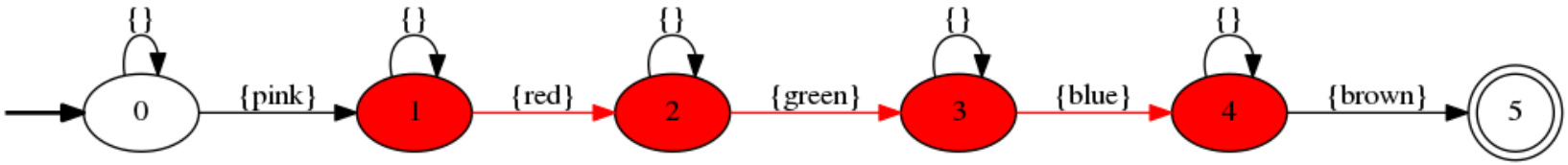


Figure 5.1: The search tree constructed by O-MCTS. In each blue box, one option is followed. The arrows represent actions chosen by the option. An arrow leading to a blue box is an action chosen by the option represented by that box.

Ad-hoc RL algorithms: Transfer Learning

- Transfer learning: We can leverage past knowledge to accomplish other tasks.

E.g. SAPIENTINO:
Consider a learned optimal policy π for the goal $\langle red, green, blue \rangle$.
We can leverage π also for $\langle pink, red, green, blue, brown \rangle$ (and similar)



Assumption: π is applicable after $0 \rightarrow 1$...

Ad-hoc RL algorithms: other ideas

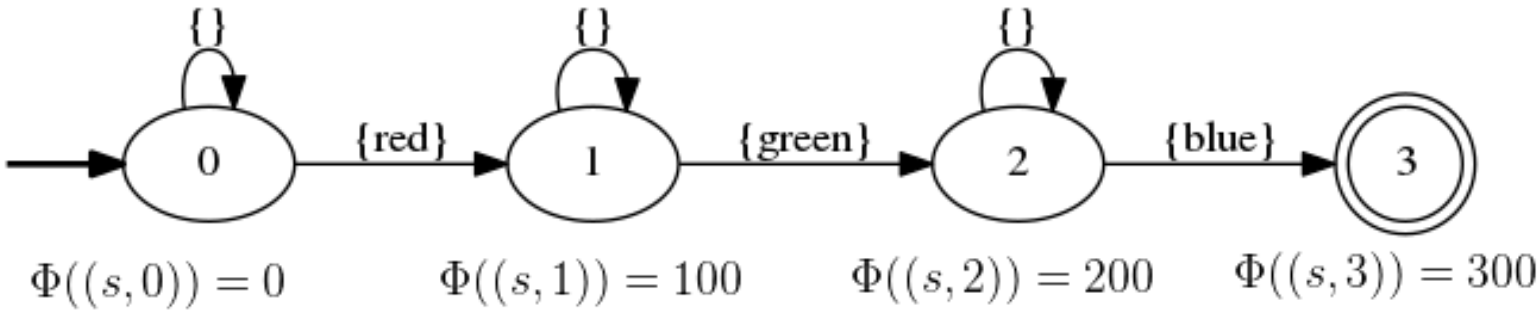
- **Curriculum Learning**: focus the learning on a small part of a single automaton rather than every automaton in its entirety
- **Manifold representations**: Why only two representations? Generalize the approach to arbitrarily nested (high) representations:

	set water temp	add hot
wake up	get wet	add cold
have a shower	shampoo	...
have a breakfast	soap	
...	turn off water	
...	dry off	

- **Multi-agent systems**: extend the approach to multi-agent systems (e.g. automaton for common goal, shared high-level representation)

Reward Shaping improvements

Our system employs the use of (potential-based) reward shaping to facilitate the exploration, i.e.:
Rewards are computed by simply split total rewards proportionally.



Reward shaping reduces to find a *potential function* $\Phi(s) : \mathcal{S} \rightarrow \mathbb{R}$.
Guarantee of policy invariance (Ng et al., 1999)

Reward Shaping improvements

Drawbacks of "static" RS:

- **Fixed rewards**: potentials do not change according to the environment changes
- **Arbitrary rewards**: shaping rewards do not reflect the true "hardness" an automaton transitions

Solutions:

- Employ **different rewards assignment schemes** (e.g. inverse proportion, attenuation factors)
- "Shaping shaping rewards": update shaping rewards *dynamically*, during the learning process, **proportionally to the hardness** of the transitions (we should define *hardness*...)

Miscellaneous

- Extend the approach to different logical formalisms that can be translated to DFA (e.g. PLTL, LTL safe, LTL co-safe, ...)
- Improve existing implementations (i.e. RLTG[1]/ FLLOAT[2])
- Benchmarking with other approach on common task/environments
- Test the approach in many environments/contexts/goals (also physical ones)

[1] `https://github.com/MarcoFavorito/RLTG`
[2] `https://github.com/MarcoFavorito/FLLOAT`

Summary

Improvements and extensions of "*RL for* $\mathrm{LTL}_f/\mathrm{LDL}_f$ *goals*".

Experiments, experiments, experiments.

More generally, working in the middle of logic-based (i.e. Planning and Reasoning) and learning-based (i.e. Reinforcement Learning) approaches.

Thank you for your attention!