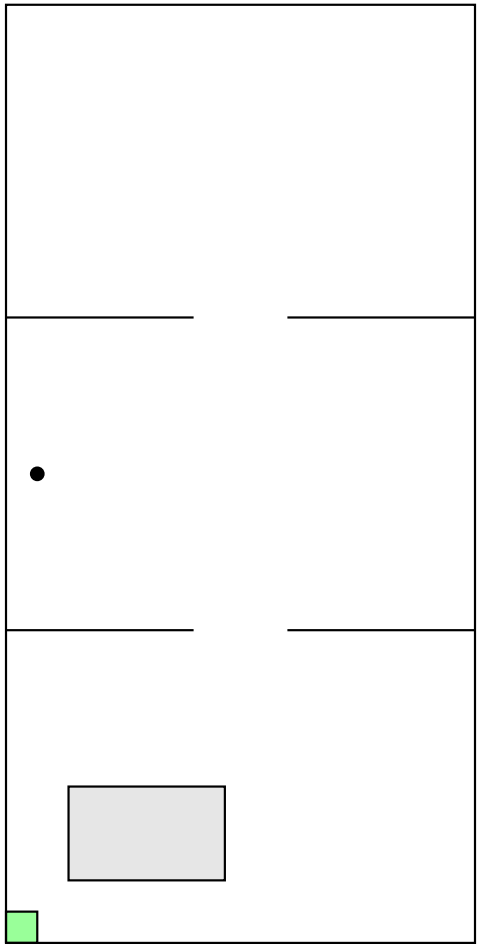
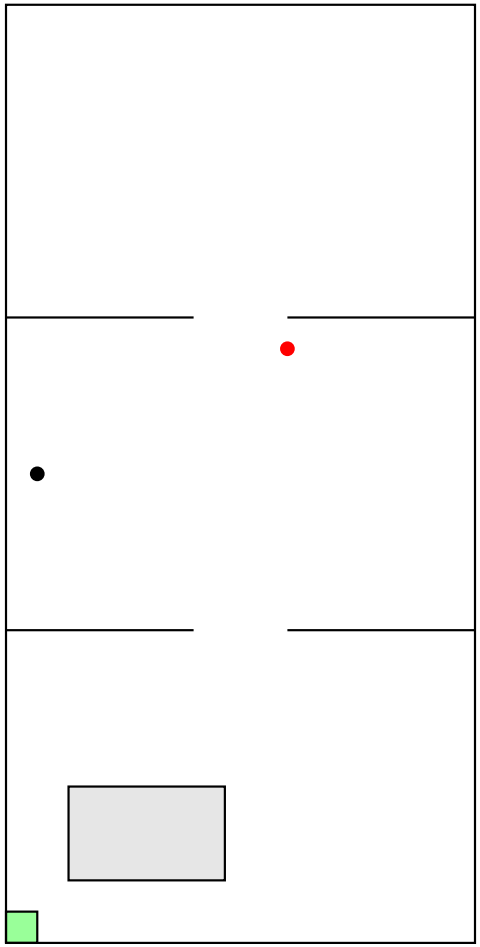
Robot Motion Planning

F. Barbosa, K. Grover, J. Křetínský, J. Tumova

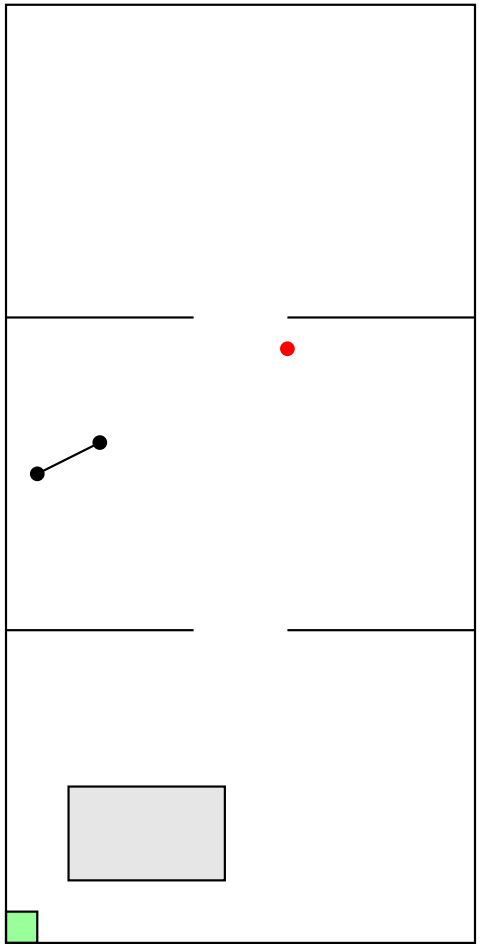
Motion Planning Problem



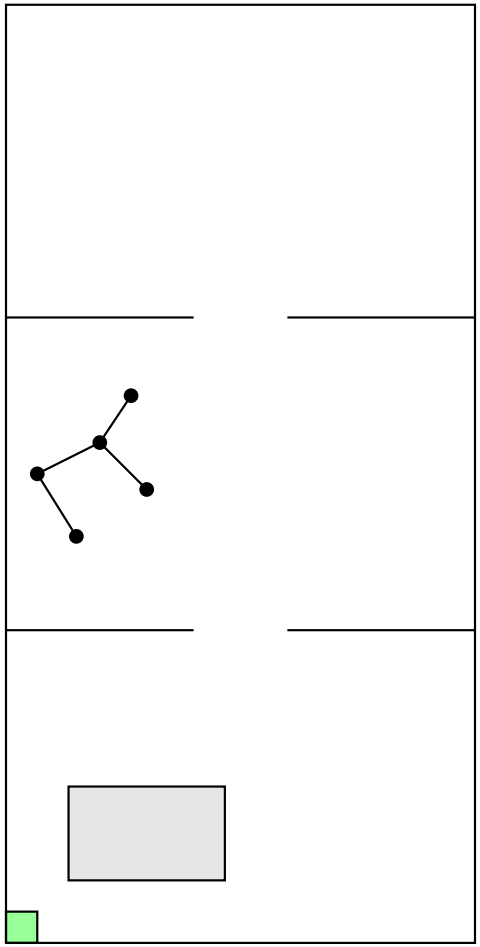
RRG



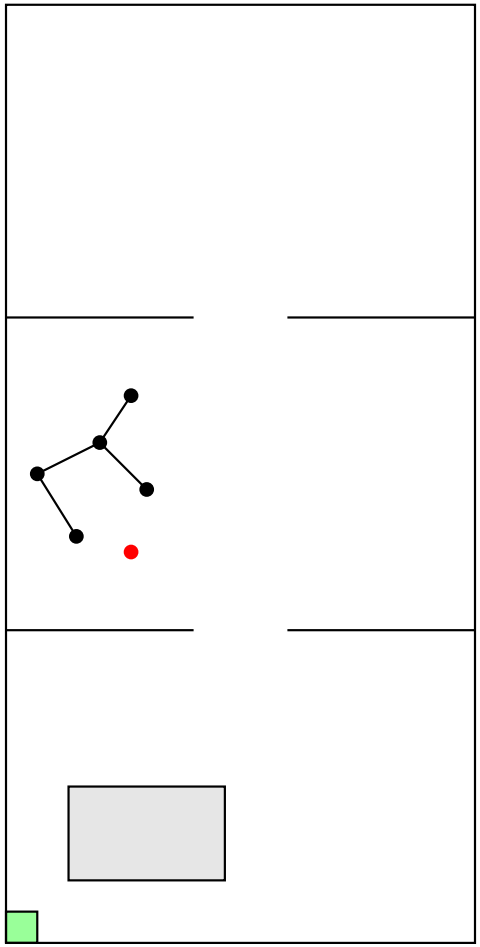
RRG



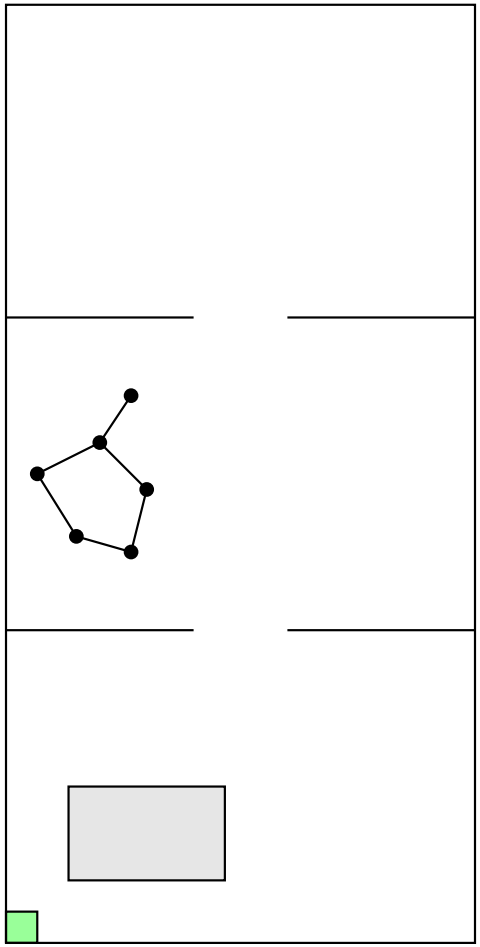
RRG



RRG



RRG



Jan's vision

- Want a robot that can move around on its own and figure out the stuff it has to do.

Jan's vision

- Want a robot that can move around on its own and figure out the stuff it has to do. ✗
  Too big a goal right now from both theoretical and practical perspectives.

Jan's vision

- Want a robot that can move around on its own and figure out the stuff it has to do. ✗
  Too big a goal right now from both theoretical and practical perspectives.

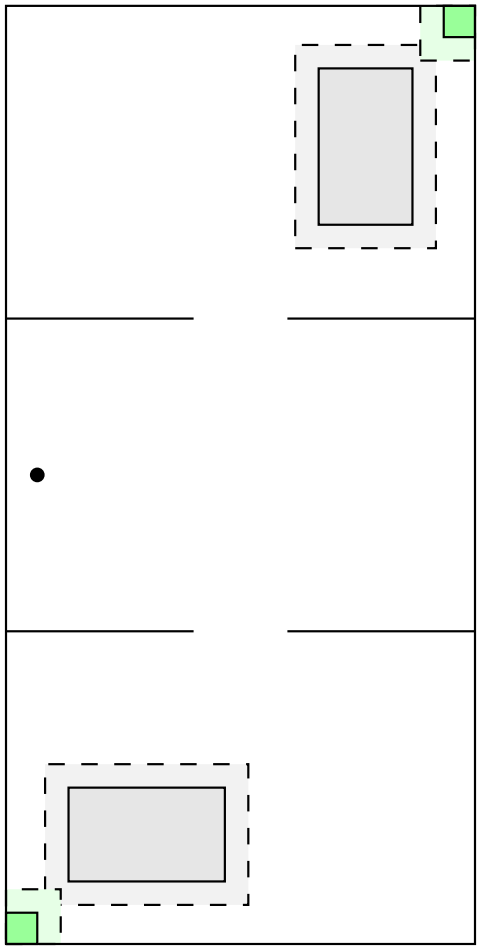
  What can we do?
  Reachability. ✓

Jan's vision

- Want a robot that can move around on its own and figure out the stuff it has to do. ✗
  Too big a goal right now from both theoretical and practical perspectives.

  What can we do?
  Reachability. ✓
  LTL or a subclass of LTL.
  Given a labeling of the environment. Find a path that satisfies a given LTL formula.

LTL Motion Planning



Specification: $GF(r_1 \wedge b) \wedge GF(r_2 \wedge b)$

Solution

- Build an abstraction of the system.
- Construct the product automaton with the property automaton (ldba).
- Find a path in the abstraction.
- Lift this path to the original system.

Better Solution

Can we do better?

Better Solution

Can we do better?

Jan's idea: Get help in sampling from current abstraction. Depending on previous experiences. In other words, predict which samples would be better from previous samples.

Better Solution

Can we do better?

Jan's idea: Get help in sampling from current abstraction. Depending on previous experiences. In other words, predict which samples would be better from previous samples.

For e.g: The bin was close to the table in first room so try to look close to the table in the other room to find the bin.

Better Solution

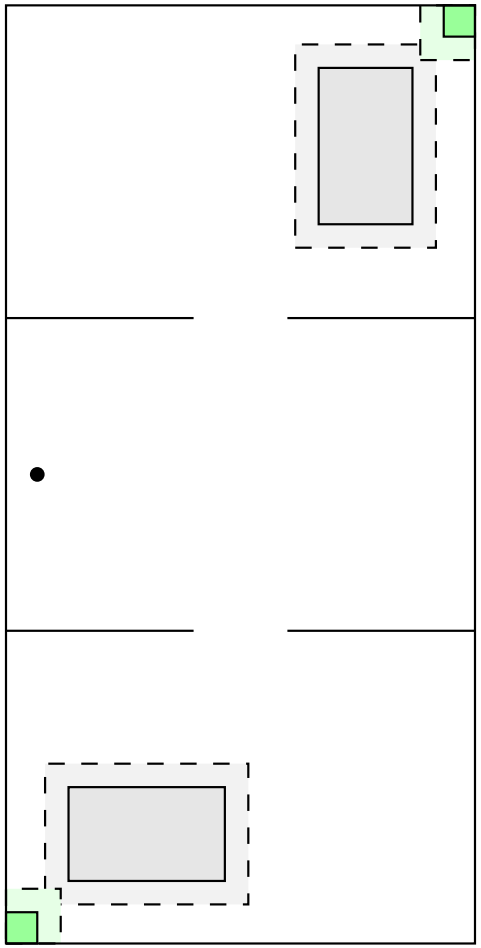
How do we formalize this?

Better Solution

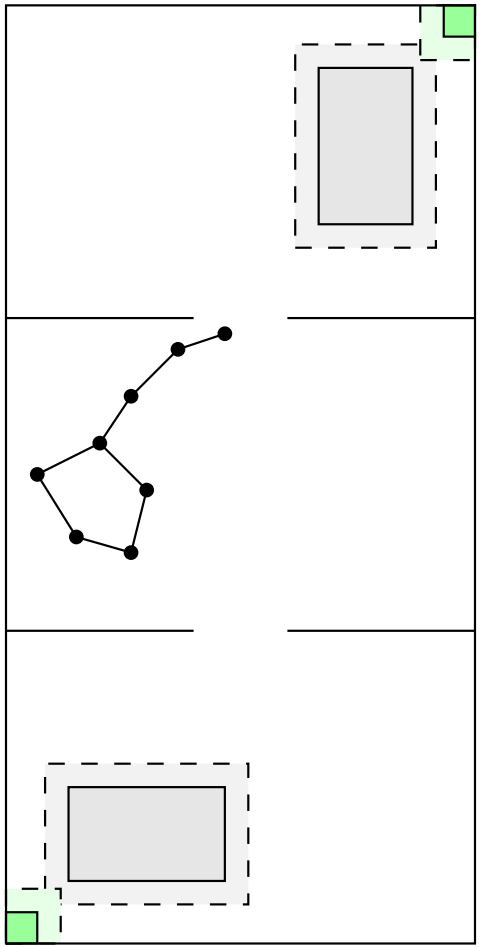
How do we formalize this?

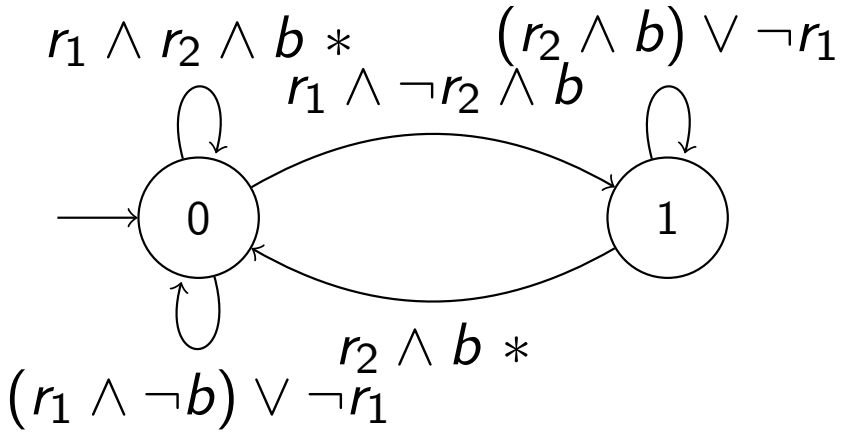
- Have 'maybe' and 'maybe not' transitions in the abstraction.
- Whenever you see a transition, add similar 'maybe' transitions in the abstraction as well.
- What does similar mean here?
  $(r_1, t) \to (r_1, b) \implies (r_2, t) \to (r_2, b)$.
  - Compute domain of changes: Set of APs changing in a transition.
    $DOC = \{t, b\}$, $(s_1 \oplus s_2)$
  - Add transitions $s_1' \to s_2'$ where s_1' and s_2' are states which agree with s_1 and s_2 on DOC respectively.

Algorithm

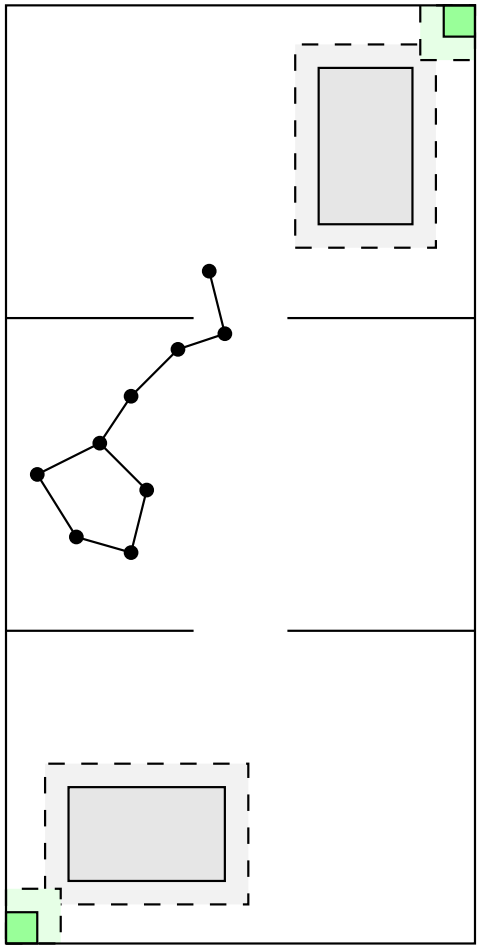


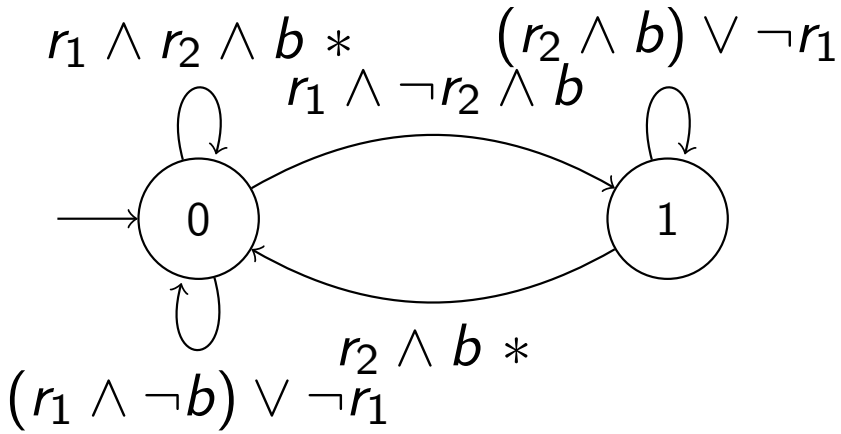
Algorithm

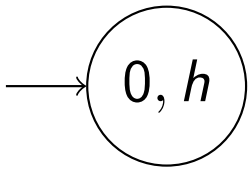



$r_1 \wedge r_2 \wedge b *$
$(r_2 \wedge b) \vee \neg r_1$
$r_1 \wedge \neg r_2 \wedge b$
0
1
$r_2 \wedge b *$
$(r_1 \wedge \neg b) \vee \neg r_1$

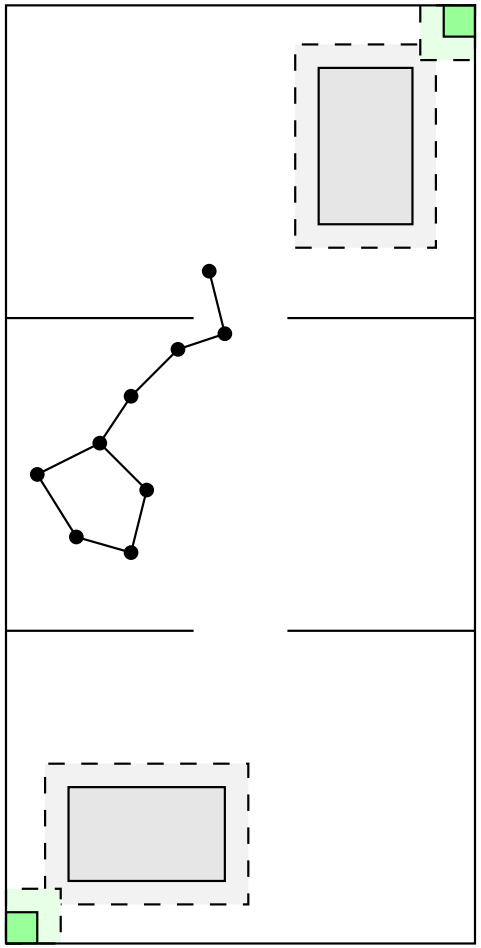
Algorithm

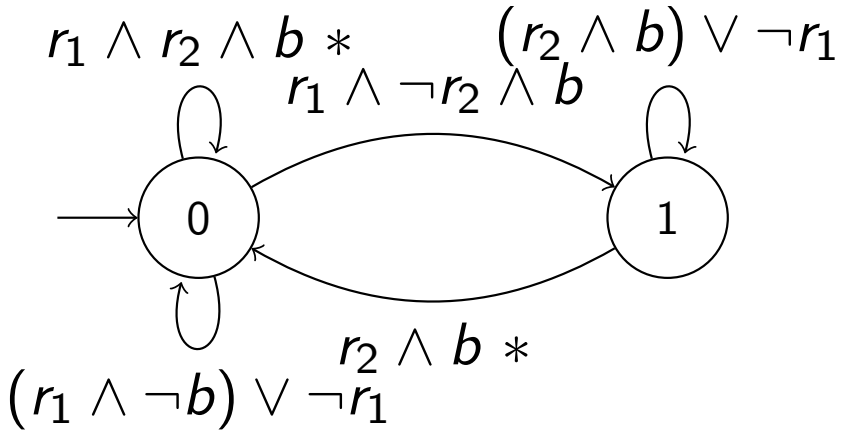


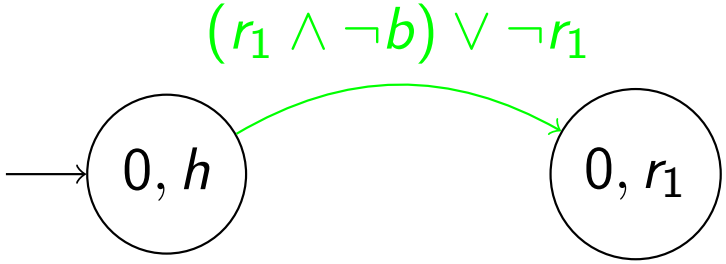

$r_1 \wedge r_2 \wedge b *$
$(r_2 \wedge b) \vee \neg r_1$
$r_1 \wedge \neg r_2 \wedge b$
0
1
$r_2 \wedge b *$
$(r_1 \wedge \neg b) \vee \neg r_1$


$0, h$

Algorithm




$r_1 \wedge r_2 \wedge b *$
$(r_2 \wedge b) \vee \neg r_1$
$r_1 \wedge \neg r_2 \wedge b$
0
1
$r_2 \wedge b *$
$(r_1 \wedge \neg b) \vee \neg r_1$


$(r_1 \wedge \neg b) \vee \neg r_1$
$0, h$
$0, r_1$

Learn

- Sample a transition s_1 to s_2.

Learn

- Sample a transition s_1 to s_2.
- Add similar transitions.

Learn

- Sample a transition s_1 to s_2.
- Add similar transitions.
- If s_1 has been visited 50 times, all the 'maybe' transitions gets converted to 'maybe not' transitions.

Ask

- Maintain a set of states in the abstraction which we have visited till now, *visitedStates*.

Ask

- Maintain a set of states in the abstraction which we have visited till now, *visitedStates*.
- Look at the accepting 'maybe' transitions and get a list of sets of states $reach[n]$, such that $reach[i]$ will have all the states in *visitedStates* that can reach an accepting transition in i steps. First try to sample transitions from $reach[1] \rightarrow reach[0]$, then from $reach[2] \rightarrow reach[1]$ and so on.

What's next?

- 'Definitely not' transitions.

What's next?

- 'Definitely not' transitions.
- Can we do full LTL? or LTL $\setminus\{X\}$.

What's next?

- 'Definitely not' transitions.
- Can we do full LTL? or LTL $\setminus\{X\}$.
- What kind of filters can we use to learn useful similar transitions?

What's next?

- 'Definitely not' transitions.
- Can we do full LTL? or LTL $\setminus\{X\}$.
- What kind of filters can we use to learn useful similar transitions?
- Make implementation faster.

What's next?

- 'Definitely not' transitions.
- Can we do full LTL? or LTL $\setminus\{X\}$.
- What kind of filters can we use to learn useful similar transitions?
- Make implementation faster.
- Figure out the best thresholds.

What's next?

- 'Definitely not' transitions.
- Can we do full LTL? or LTL $\setminus\{X\}$.
- What kind of filters can we use to learn useful similar transitions?
- Make implementation faster.
- Figure out the best thresholds.
- Still very hard because we have to compete with very efficient algorithm.

After this project

- A robot that does not have information of the whole environment and can only look at things nearby.

After this project

- A robot that does not have information of the whole environment and can only look at things nearby.
- Figure out atomic propositions as it goes on to make learning better (not the ones in the property).