

# BDI on Time

“the future of BDI is in the future”

---

*first week of December 2023*

# Agenda

---

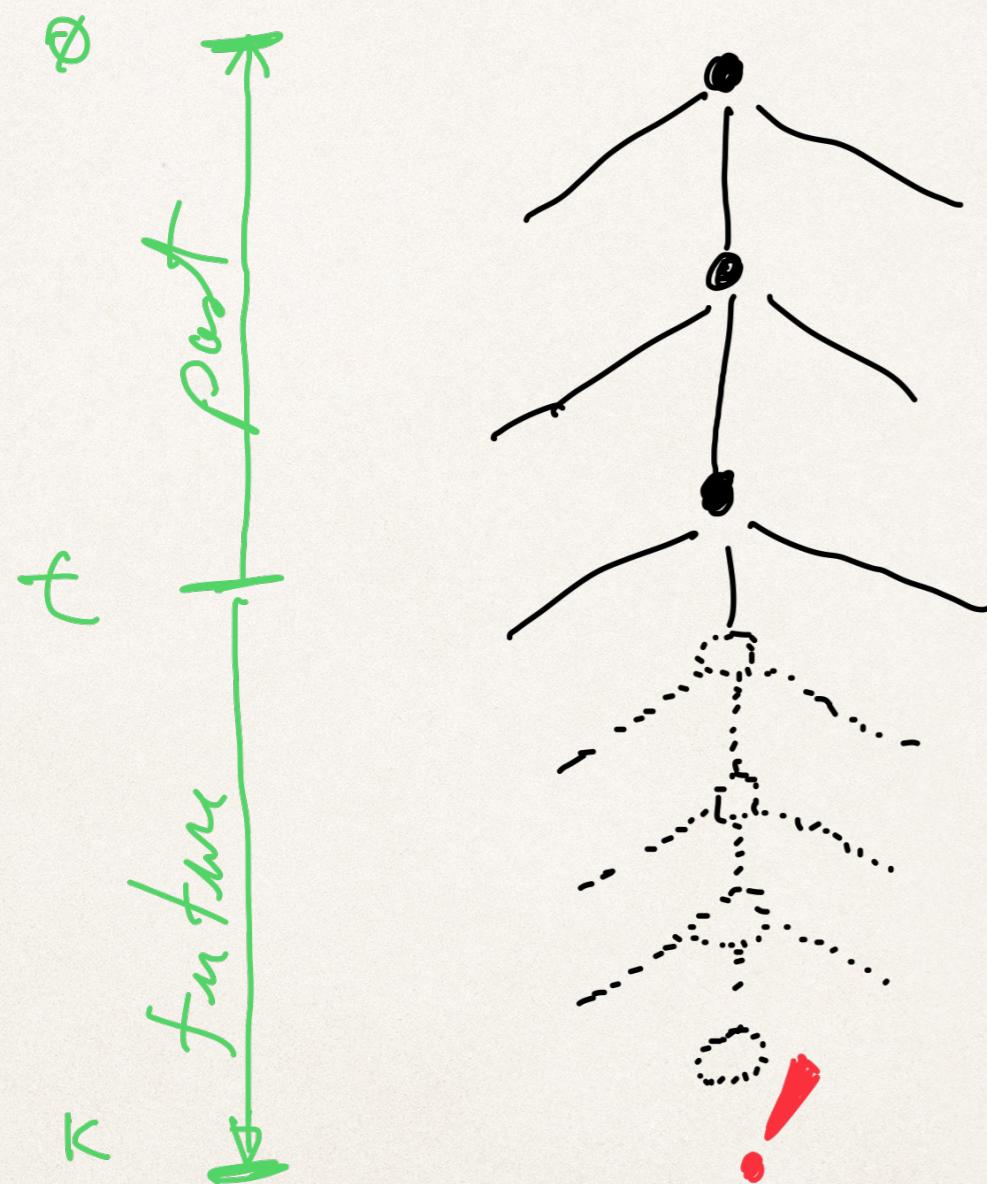
- ❖ simulating the future
- ❖ agent deliberation based on future
- ❖ (initial) experiments
- ❖ single & multi-agent
- ❖ normative reasoning

# Motivations To Consider Future & BDI

---

- ✿ BDI agents may take better (rational) decisions by *looking ahead*
- ✿ Foresee problems
- ✿ Realize better options
- ✿ ...

# Future Simulation



envi  
e :

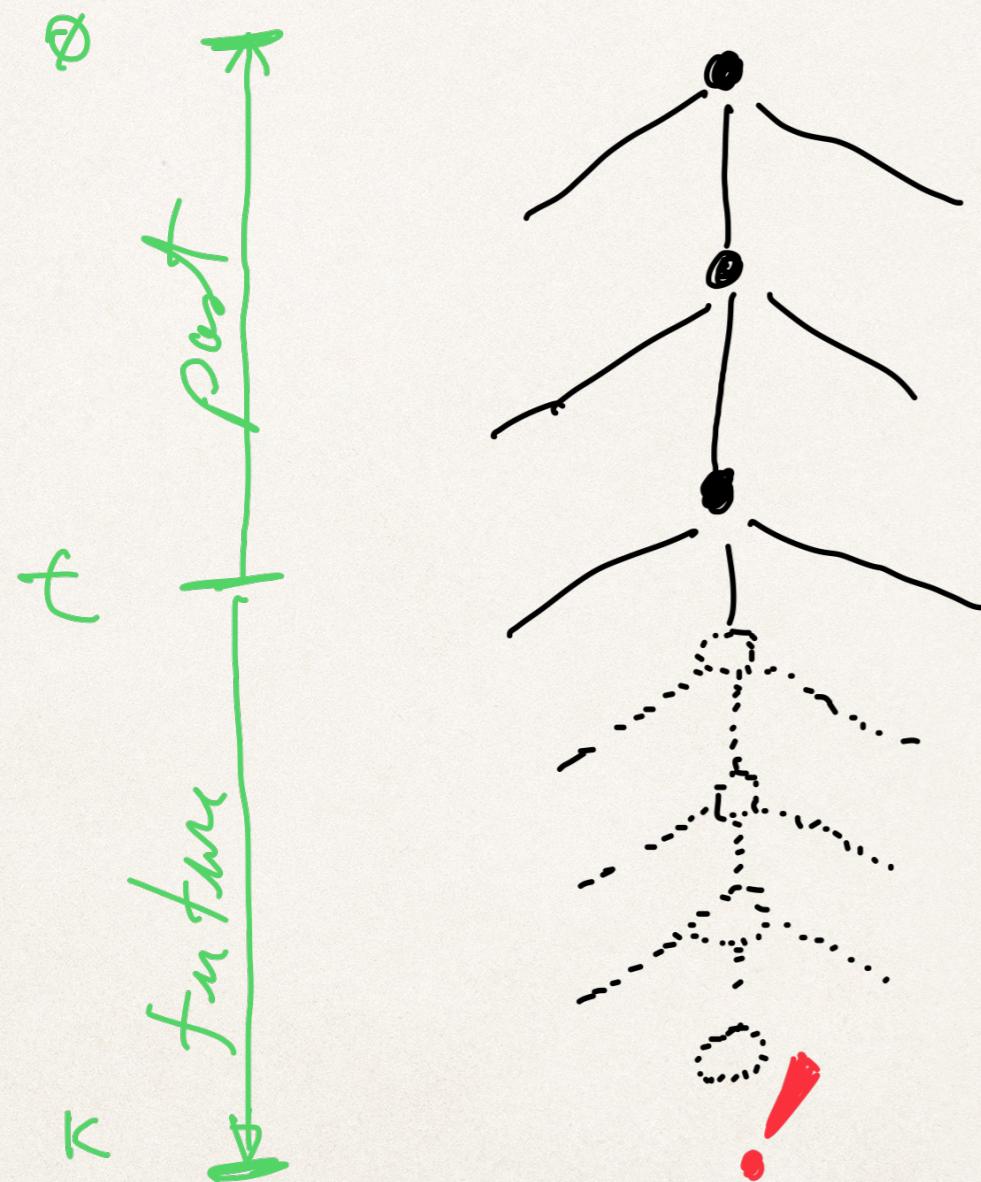
agent policy (and preferences)  
 $\pi : S \rightarrow A^n$

future state at  $k$   
 $f : S \times \mathbb{N} \times \mathbb{N} \rightarrow S$

$$f(s, t, k) = \begin{cases} s & \text{if } t = k \\ f(s', t + 1, k) & \text{if } t < k \end{cases}$$
$$s' = e(s, \pi(s)_1)$$

# Example of Problems

---



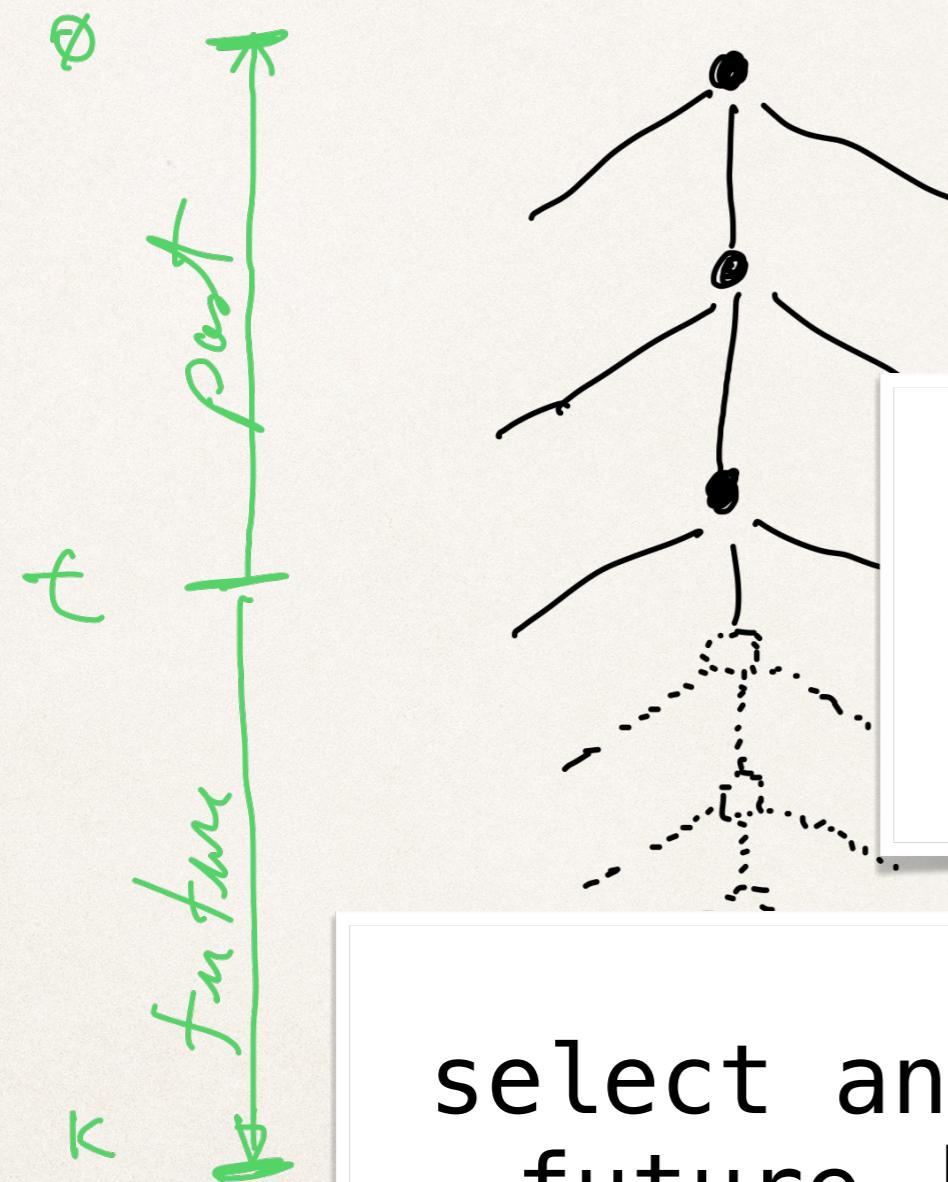
- no option
- goal not achieved
- norm / principle violation
- ethical issues
- undesired state

# Two Uses of Future

---

- ✿ Detect a problem
- ✿ Find an alternative option

# What To Do?

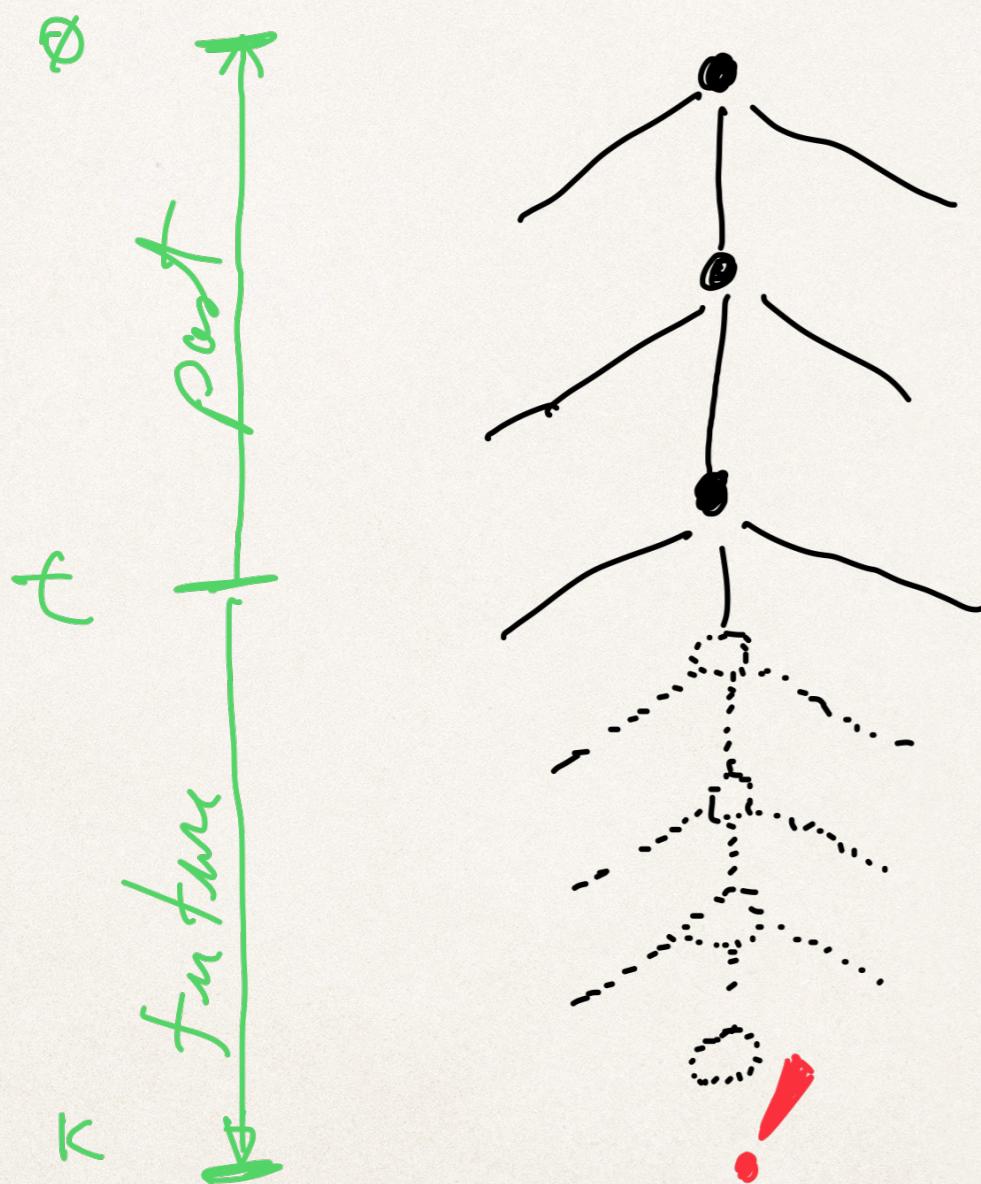


- stop
- reconsider options

what caused the problem?  
[very difficult to answer]  
[I will skip that]

select an option with “nice”  
future [without problems]

# (Initial) Delimitations

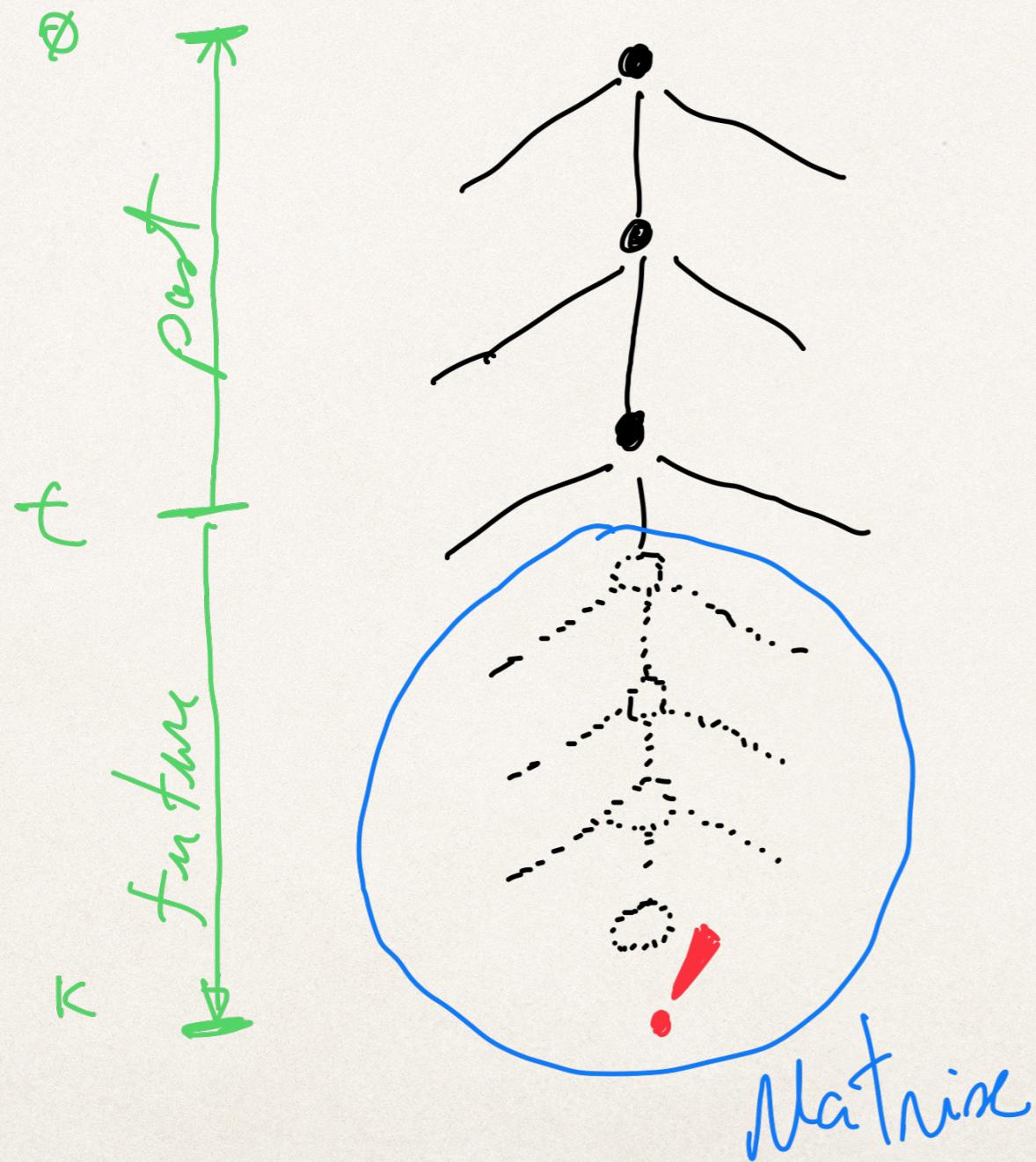


- revise options from now on ( $t..k$ )
- what option?  
[revision strategy]

move to experiments to help to answer

# Jason(F)

---



execution modes:

- normal
- **matrix mode**  
(simulated environment)

at time  $t$ :

- clone itself into another agent
- run it in the matrix
- until success, failure, timeout

branching is given by plan options  
[not actions]

# Grid Scenario



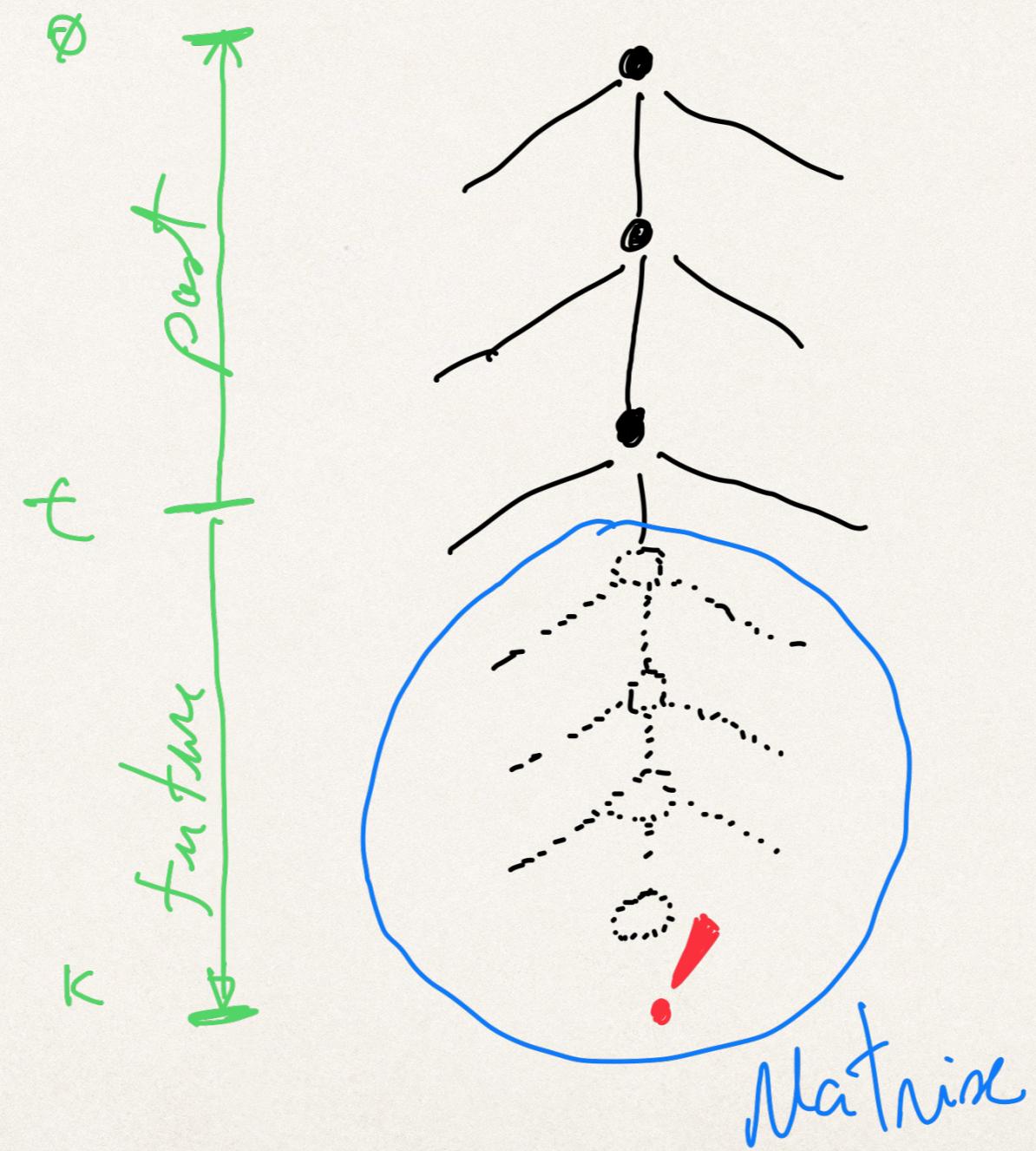
```

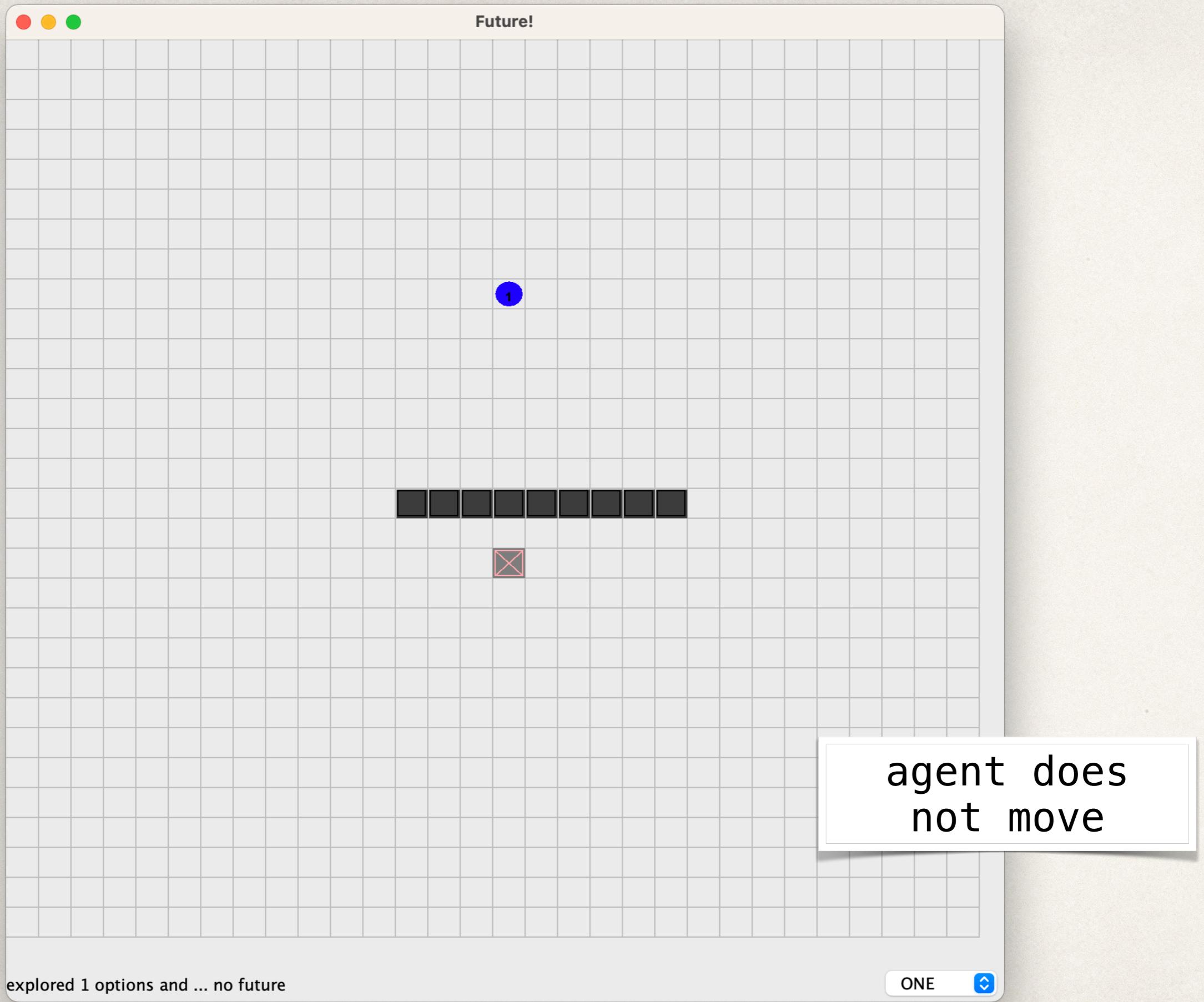
5 +destination(X,Y) :- !pos(X,Y). // create a goal when my destination is perceive
6 -destination(X,Y) :- .drop_all_desires. // drop everything if my destination is
7
8 @ [preference(0)] +!pos(X,Y) : pos(X,Y).
9 @s [preference(D)] +!pos(X,Y) : ok(s) & distance(s,D) <- s; !pos(X,Y).
10 @sw[preference(D)] +!pos(X,Y) : ok(sw)& distance(sw,D) <- sw; !pos(X,Y).
11 @se[preference(D)] +!pos(X,Y) : ok(se)& distance(se,D) <- se; !pos(X,Y).
12 @w [preference(D)] +!pos(X,Y) : ok(w) & distance(w,D) <- w; !pos(X,Y).
13 @e [preference(D)] +!pos(X,Y) : ok(e) & distance(e,D) <- e; !pos(X,Y).
14 @n [preference(D)] +!pos(X,Y) : ok(n) & distance(n,D) <- n; !pos(X,Y).
15 @nw[preference(D)] +!pos(X,Y) : ok(nw)& distance(nw,D) <- nw; !pos(X,Y).
16 @ne[preference(D)] +!pos(X,Y) : ok(ne)& distance(ne,D) <- ne; !pos(X,Y).
17
18 // checks if go to some direction is possible (free cell)
19 ok(D) :- next(D,X,Y) & free(X,Y).
20
21 next(s ,X ,Y+1) :- pos(X,Y). // my next location if doing south
22 next(sw,X-1,Y+1) :- pos(X,Y).
23 next(se,X+1,Y+1) :- pos(X,Y).
24 next(w ,X-1,Y ) :- pos(X,Y).
25 next(e ,X+1,Y ) :- pos(X,Y).
26 next(n ,X ,Y-1) :- pos(X,Y).
27 next(nw,X-1,Y-1) :- pos(X,Y).
28 next(ne,X+1,Y-1) :- pos(X,Y).
29
30 free(X,Y) :- X >= 0 & Y >= 0 & w_size(W,H) & X < W & Y < H & not obstacle(X,Y).
31 distance(Dir,Dist) :- next(Dir,X,Y) & destination(GX,GY) &
32 | | | | | Dist = math.sqrt( (X-GX)**2 + (Y-GY)**2 ) .

```

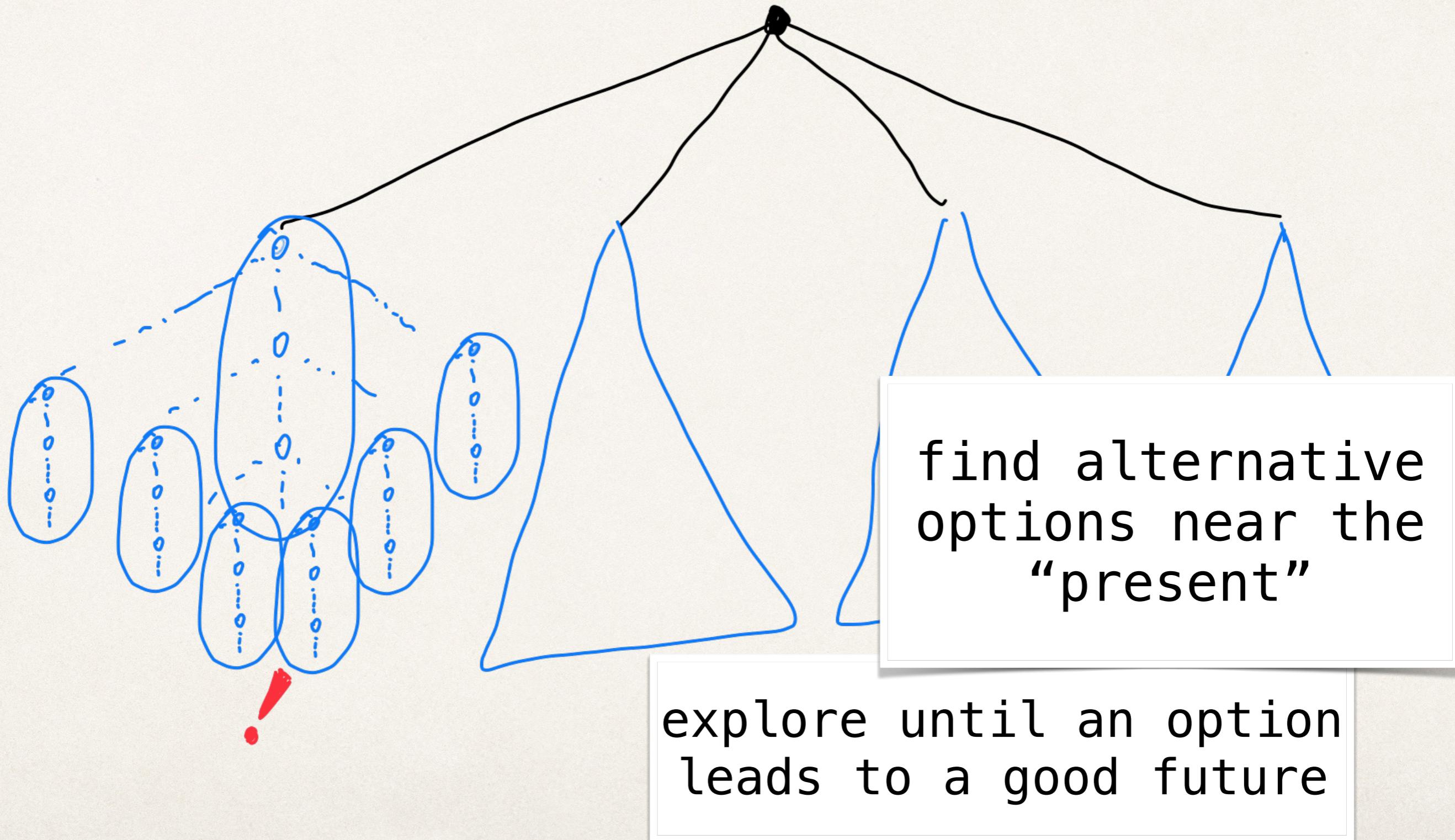
# Strategy ONE (Foresee the Problem)

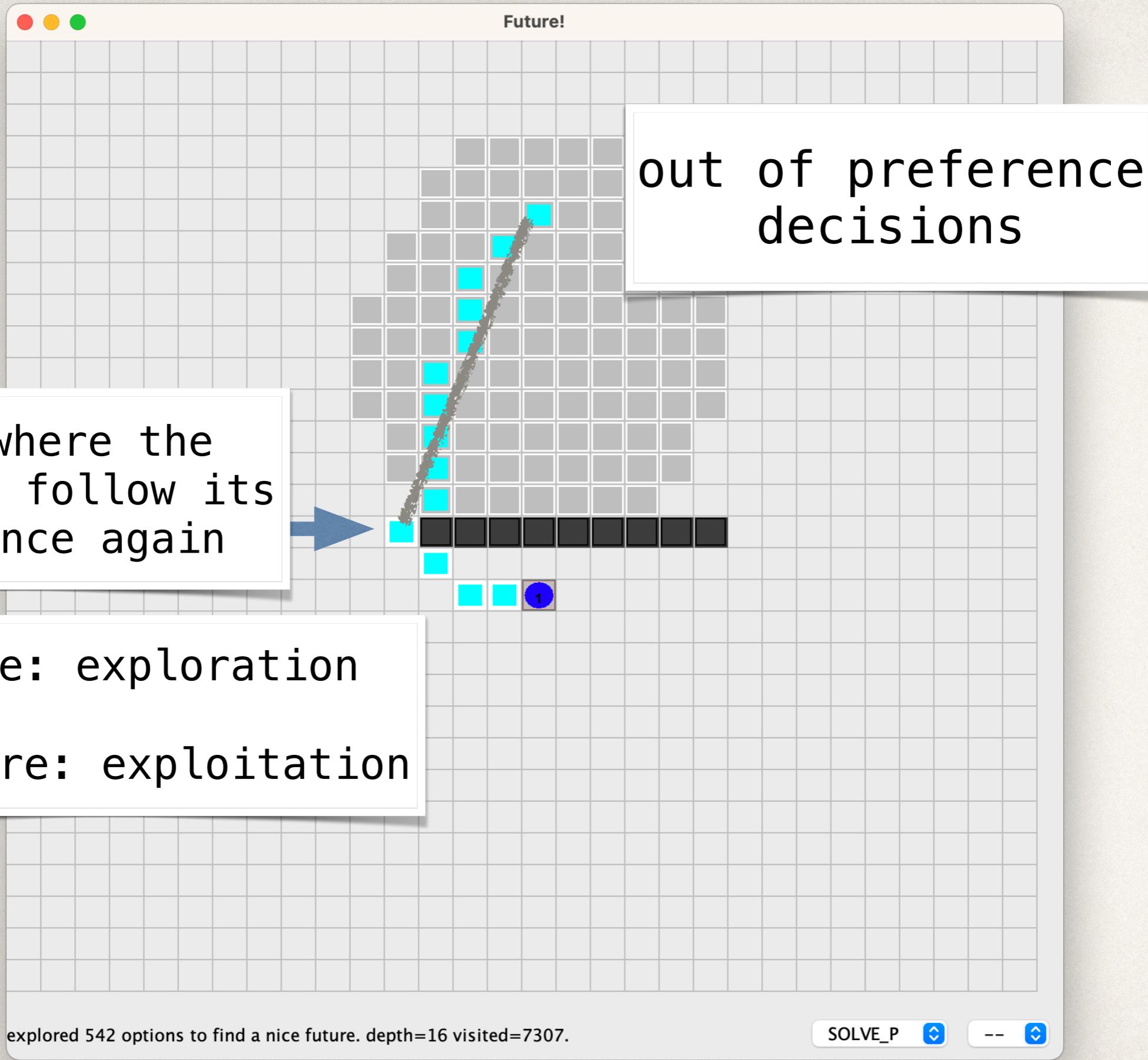
---





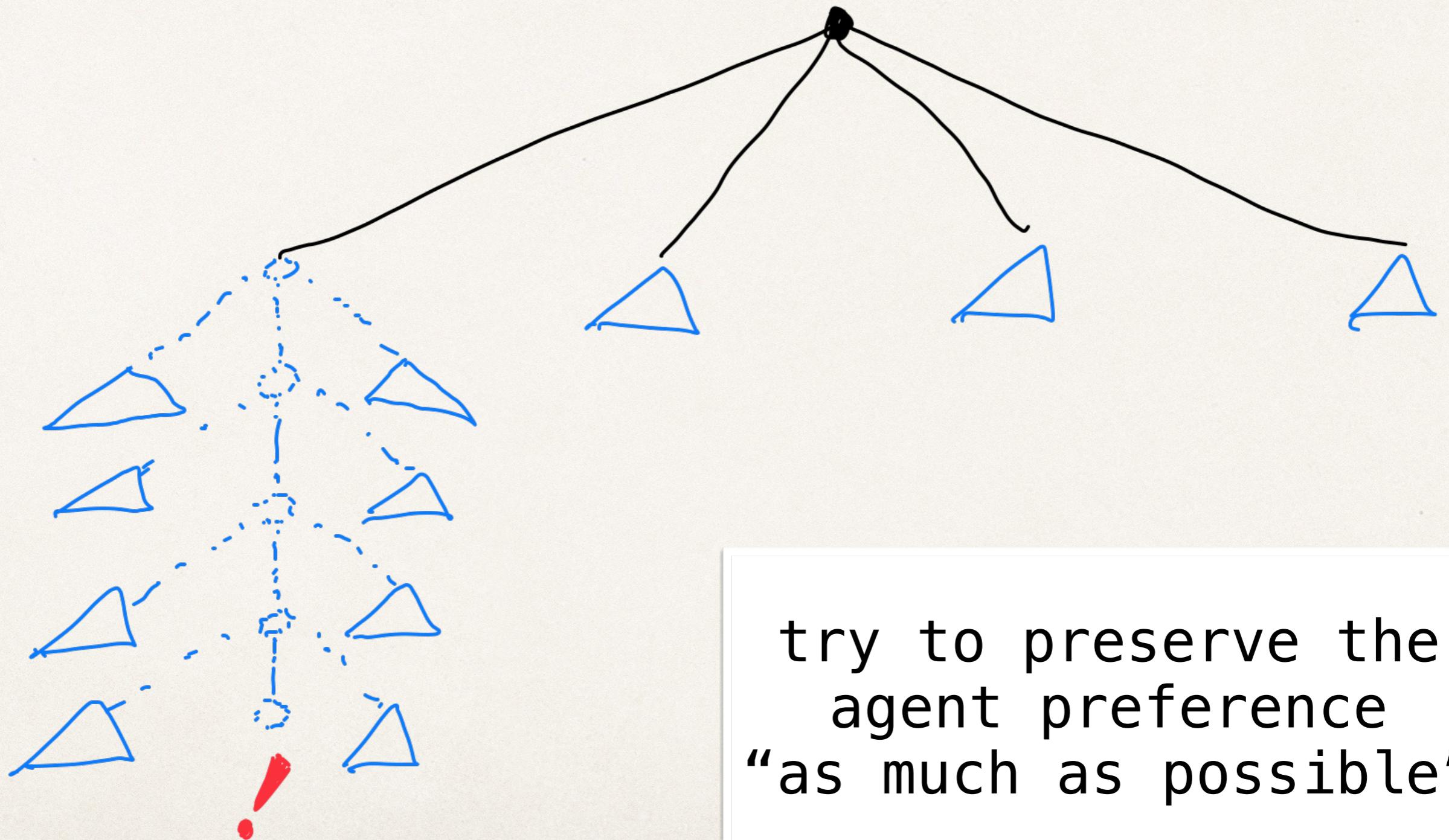
# Strategy SOLVE\_P – Search Tree

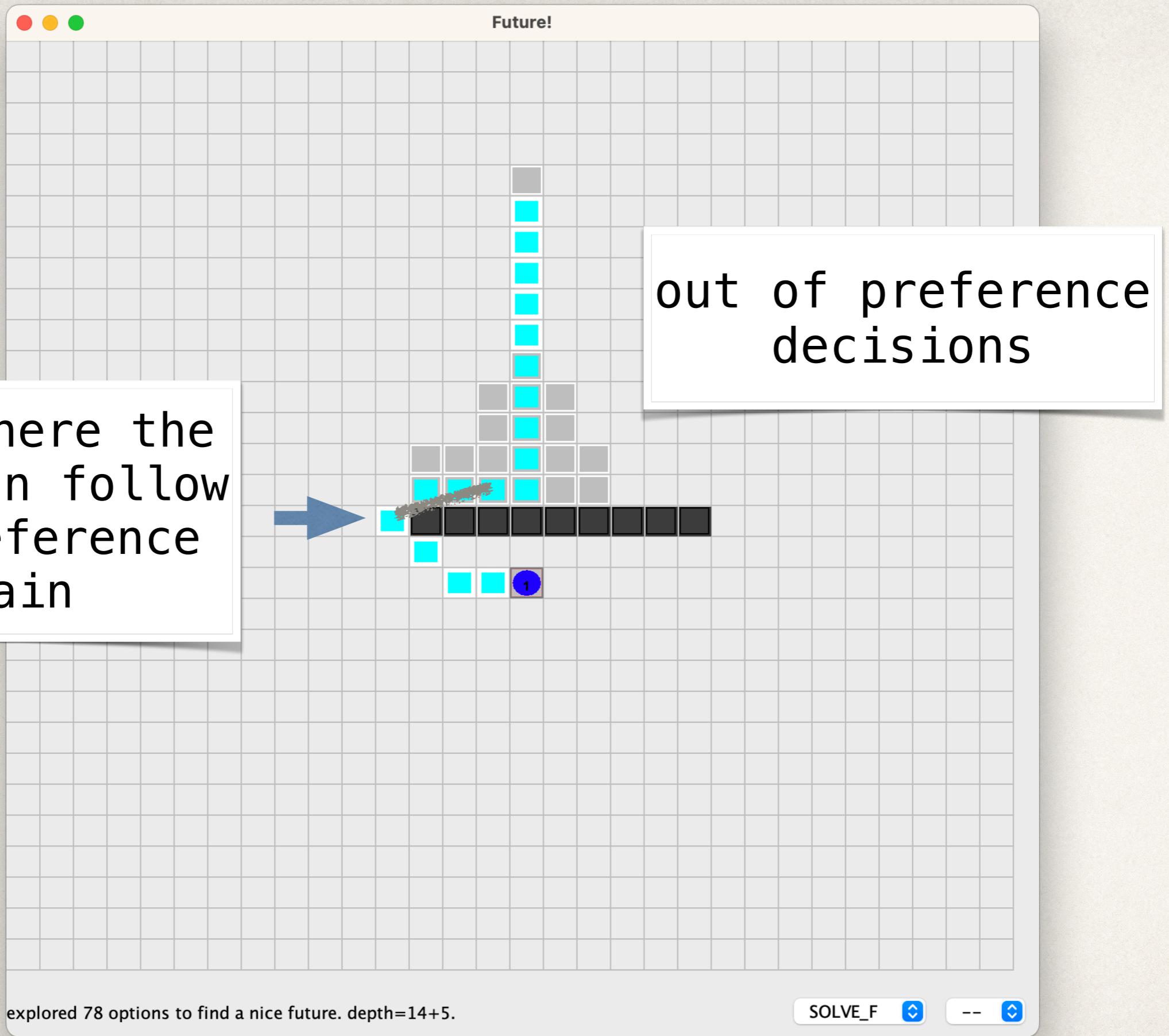


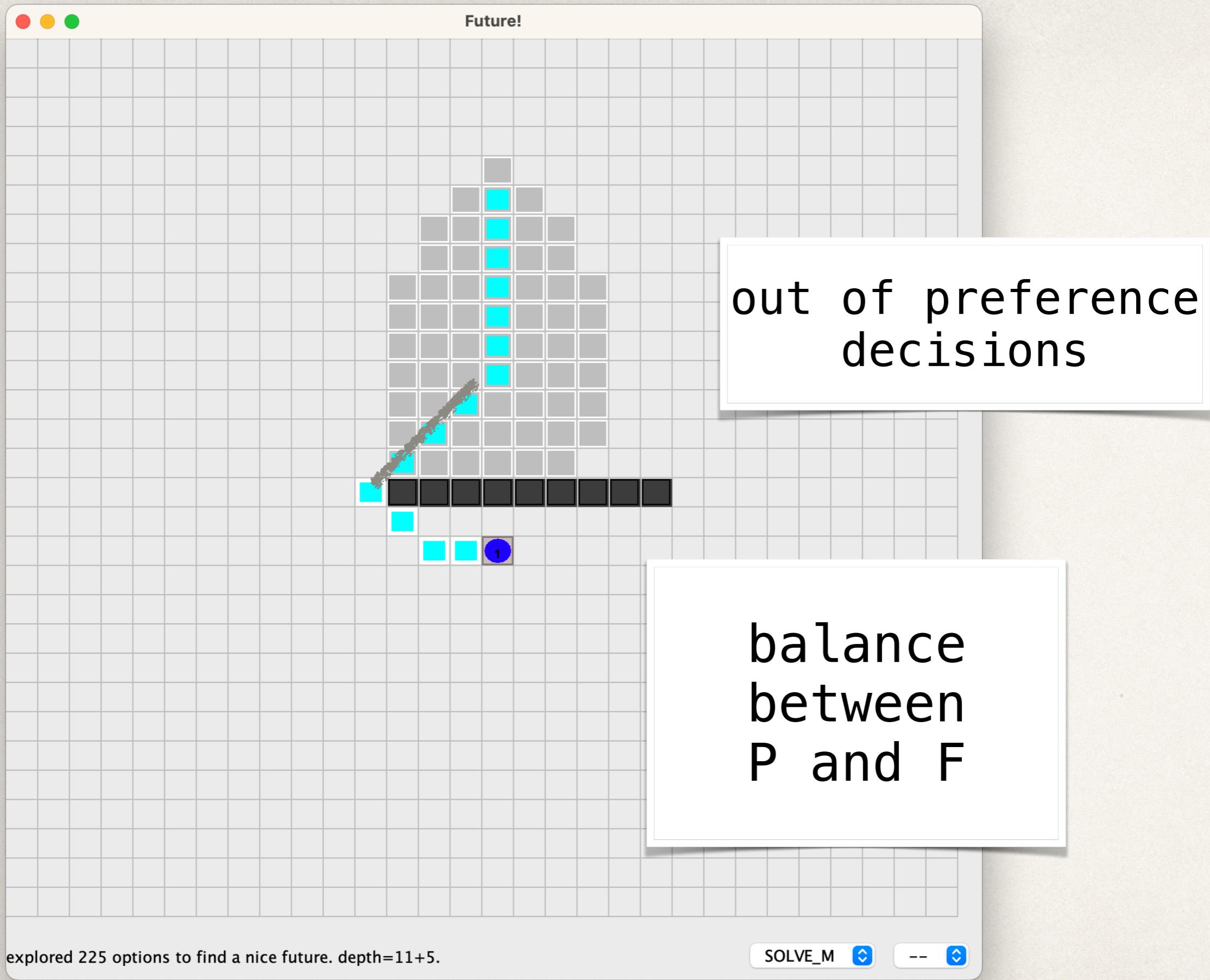


# Strategy SOLVE\_F

---







# version 1.1

## Scenario line

little "detour"

updated  
HERE

## Scenario U

good for the agent policy, need to anticipate that entering the U is not a good option

## Scenario H

bad for the policy

scenario	strategy	visited	steps to solution	steps out of polity	steps in policy
line	SOLVE_P	419	15	9	6 (40%)
	SOLVE_M	217	15	4	11 (73%)
	SOLVE_F	78	18	4	14 (77%)
U	SOLVE_P	112	24	20	4 (16%)
	SOLVE_M	286	24	20	4 (16%)
	SOLVE_F	374	28	24	4 (14%)
H	SOLVE_P	2006	26	22	4 (15%)
	SOLVE_M	2028	26	22	4 (15%)
	SOLVE_F	2001	28	24	4 (14%)

# Results

---

- ✿ move the agent out of its preference  
(while exploiting its policy as much as possible)
  
- ✿ SOLVE\_P: change option as **soon** as possible
  - + shorter plan
  - more exploration
  
- ✿ SOLVE\_F: change option as **latte** as possible
  - + less exploration
  - not so good plan

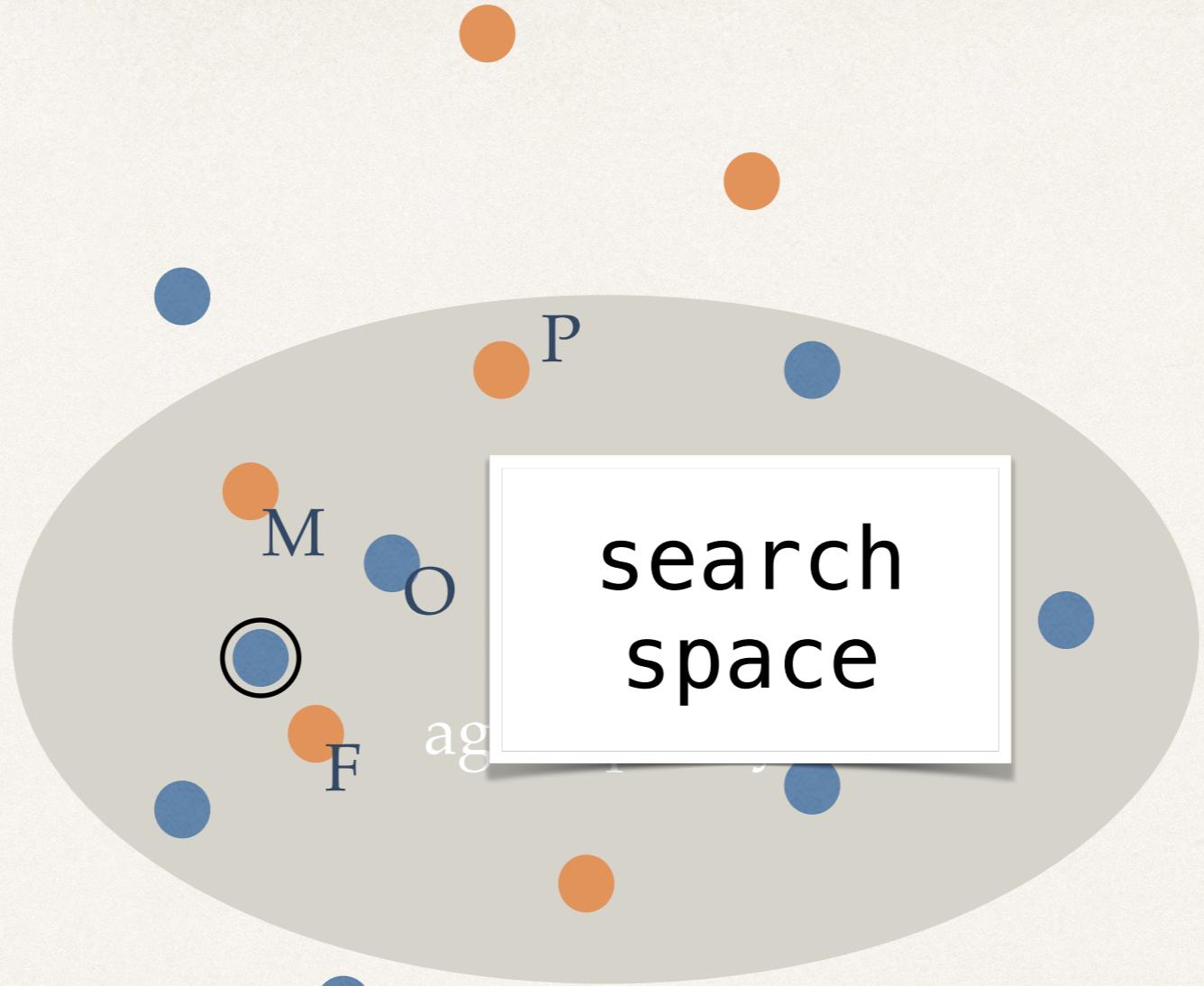
# About the Agent Program

---

- ✿ the agent program (that defines its policy) is the **(how-to) knowledge** the programmer gives to the agent
- ✿ without it, we (may) have traditional search  
(to consider this **K** distinguishes Jason(S) from search)
- ✿ if **K** is very very bad designed, we (may) have traditional search [if, at least, K gives options]
- ✿ if **K** very very well designed, Jason(F) does nothing  
(the future of the policy is always good)

possible behaviors (ok | not ok)

required "energy"



# Questions

---

- ✿ is it just search? planning? monte carlo?  
[it is inspired on all these, but...]  
[we have the agent policy]  
[no adversarial]
  
- ✿ does agent policy (options + preference) play as heuristic? [i guess so — preference part]  
[Jason(F) scapes from preference when it isn't ok]
  
- ✿ is it optimal?  
[no] [maybe if preference + cost are considered]

# Questions

---

- ✿ does it work for any kind of agent program?  
[it works if the program produce *enough* options]  
[just one option: nothing to do; several options (with no preference): usual search]
- ✿ does it work without preference?  
[yes, but less efficient]
- ✿ how much do we depend on preferences? [for efficiency]  
can we survive without? [yes]

# Jason(F) 1.1

---

- ❖ initial scenarios
- ❖ handles internal problem (goal not achieved)
- ❖ strategies: one, solve\_(p|m|f)
- ❖ generic for any Jason program  
[+ environment model]

# Multi-Agent

- ✿ Requires model of others
  - ✿ ask them for their model
  - ✿ they are as me
  - ✿ developer provides
- ✿ cooperation implies that changes in “path” may be more critical

fully observable  
deterministic

multi-agent  
envi  
 $e :$  static  
agen  
discret

$$\pi_i : S \rightarrow A^n$$

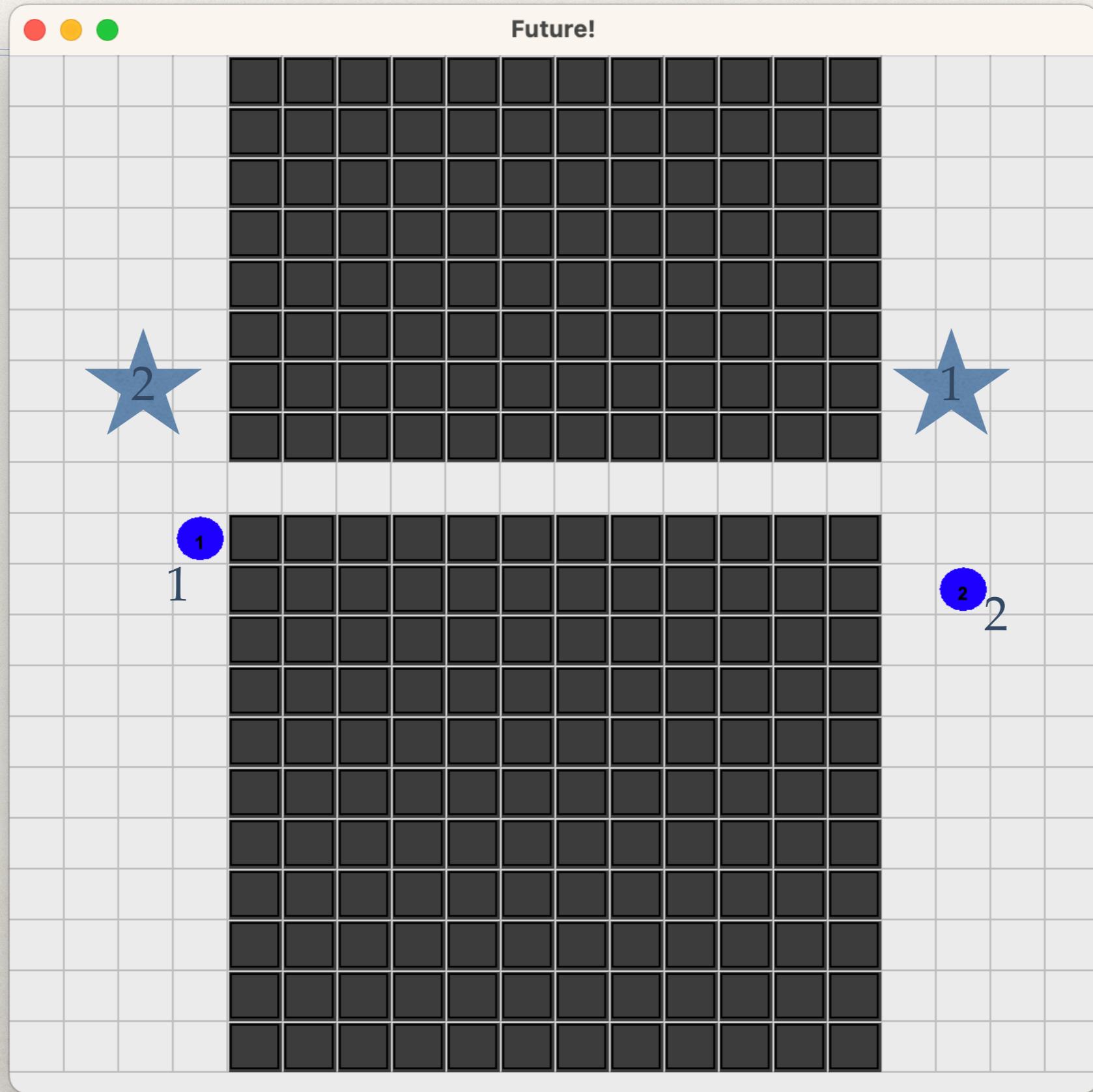
future state at  $k$

$$f : S \times \mathbb{N} \times \mathbb{N} \rightarrow S$$

$$f(s, t, k) = \begin{cases} s & \text{if } t = k \\ f(s', t + 1, k) & \text{if } t < k \end{cases}$$

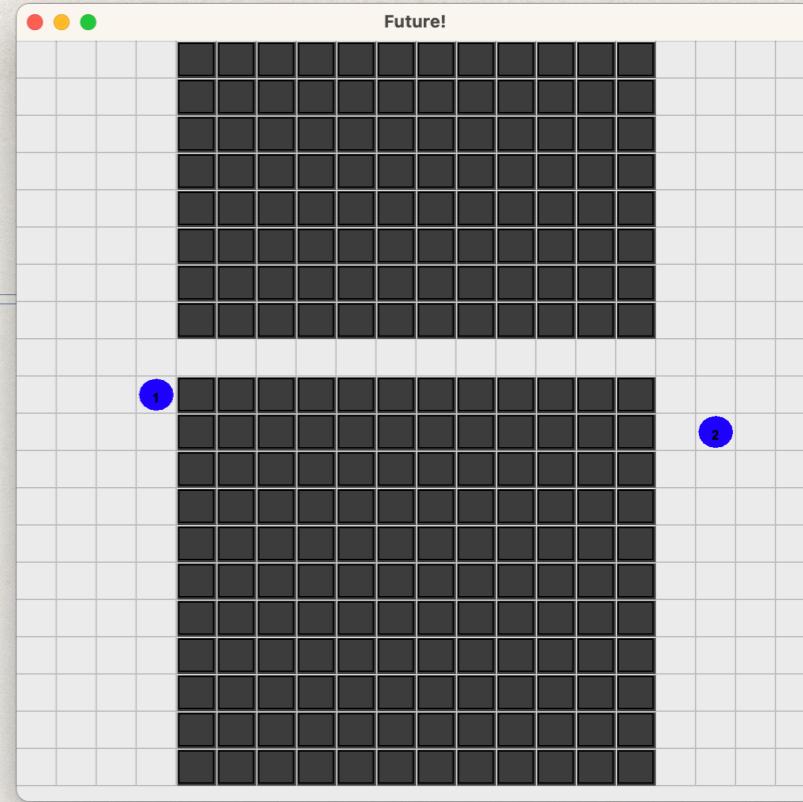
$$s' = e(s, \pi_1(s)_1, \pi_2(s)_1, \dots)$$

# Bridge Scenario



# Strategies

- ✿ none: both agents are blocked in the middle of the bridge
- ✿ left agent with **one**: right agent crosses, left agent gives up
  - ✿ or has a **recovery plan** to wait and try again latter  
(this is the best solution for this problem)
- ✿ left agent with **solve\_p**: right agent crosses, left waits and then crosses  
(takes quite a while to find a good option)



# Results

updated  
HERE

strategy	solve	matrices	visited	steps	steps in policy
NONE	no	0	$\infty$	$\infty$	$\infty$ (100%)
ONE	no	1	25	—	0 (0%)
ONE [1]	yes	15 [2]	15	15	15 (100%)
SOLVE_P	yes	942	8333	27	21 (77%)

solve\_p is slower to find options  
(higher cost of building matrixes)

# Jason(F) 1.3

---

- ❖ **matrix with multi-agents**  
[no communication considered]
- ❖ others are simulated as regular (non-Jason(F)) agents
- ❖ (still) internal problem (goal not achieved)
- ❖ strategies: one, solve\_(p | m | f)
- ❖ generic for any Jason program  
[+ environment model]  
[model of others is trivial in Jason, just get their state]

# Remarks for Multi-agent case

---

- ✿ Nothing new or special for the multi-agent case (!)
- ✿ From the Jason(F) agent perspective, others are part of the environment
  - ✿ only their actuation in the environment is indeed perceived
- ✿ if both agents are Jason(F), both will wait 5 steps and then try to cross at the same time!
  - ✿ some coordination strategy should (must?) be put in place (e.g., a leader that decides who cross)

# Jason(F) 1.4

---

- ❖ Problem detection vs Problem resolution
  - ❖ detection is based on strategy ONE in selectOption
  - ❖ resolution based on internal action that produces a new plan (from search in future options)

```
-!pos(X,Y)[error(no_future),error_msg(M)]  
  : pos(CX,CY) // my location  
<- .print("Failure for goal pos(",X,",",",Y,"):",",M);  
  jason.future.plan_for(  
    pos(X,Y),  
    { @[cost(0), preference(0)] +!pos(X,Y) : pos(CX,CY) },  
    Plan, "SOLVE_M");  
  .add_plan(Plan, chunking, begin);  
  !pos(X,Y).
```

# (Un)Certainty

environment transitions

$$e : S \times A^m \times S \rightarrow [0,1]$$

$e(s, a_1, \dots, a_m, s')$  is the probability of  
s' from s doing a

agent  $i$  policy

$$\pi_i : S \rightarrow A^n$$

state and certainty in the future

$$f : S \times \mathbb{N} \times \mathbb{N} \rightarrow S \times [0,1]$$

$$f(s, t, k) = \begin{cases} s, 1 & \text{if } t = k \\ s', e(s, \dots, s') f(s', t + 1, k)_2 & \text{if } t < k \end{cases}$$

$s' = \max_{s''} e(s, \pi_1(s)_1, \pi_2(s)_1, \dots, s'')$  most probable next state

fully observable

**stochastic**  
**dynamic**  
environment

deterministic  
agent

multi-agent

discret

# Jason(F) 1.5

---

- ❖ Uncertainty
  - ❖ matrix stops on lack of certainty

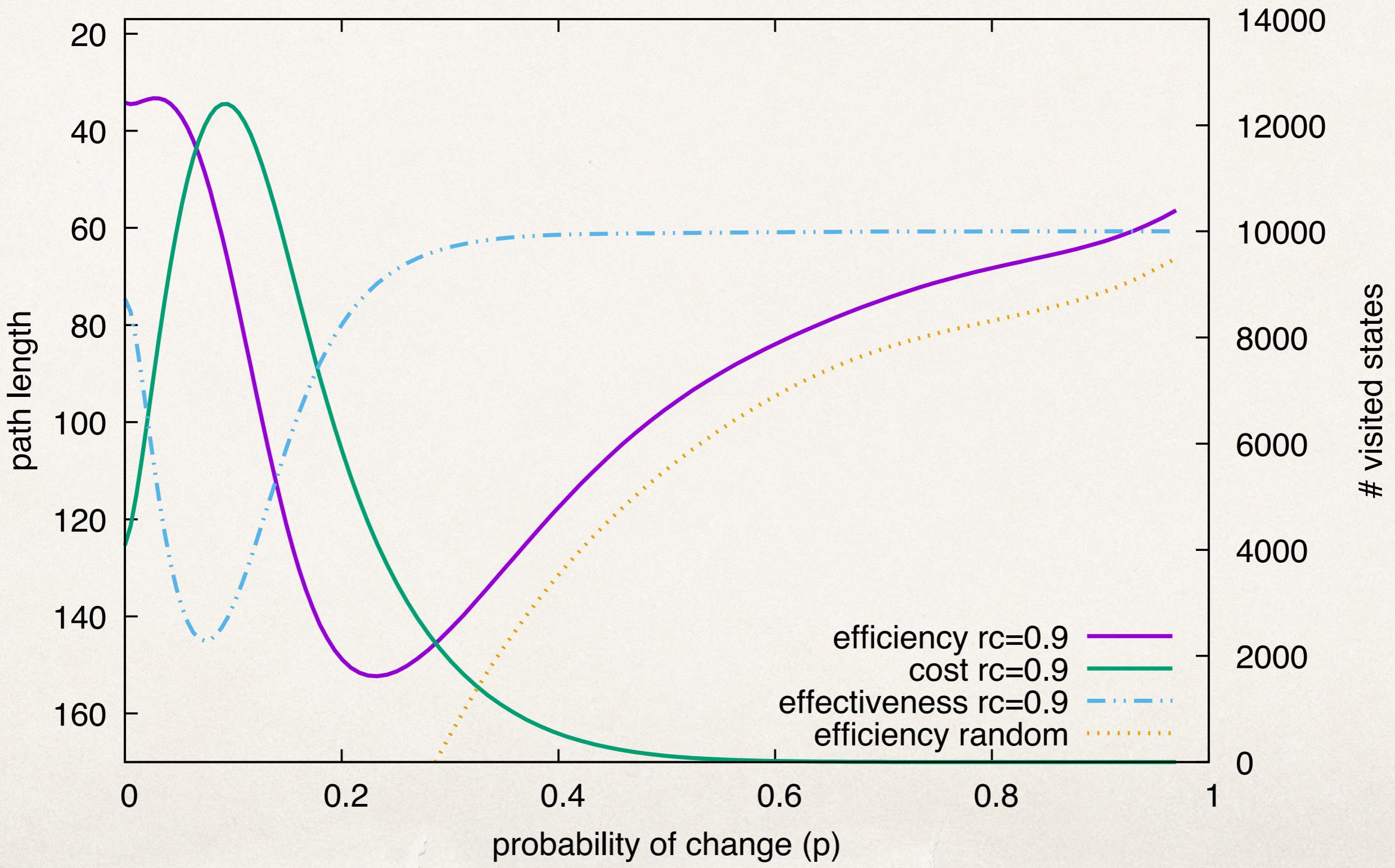
# Evaluation

---

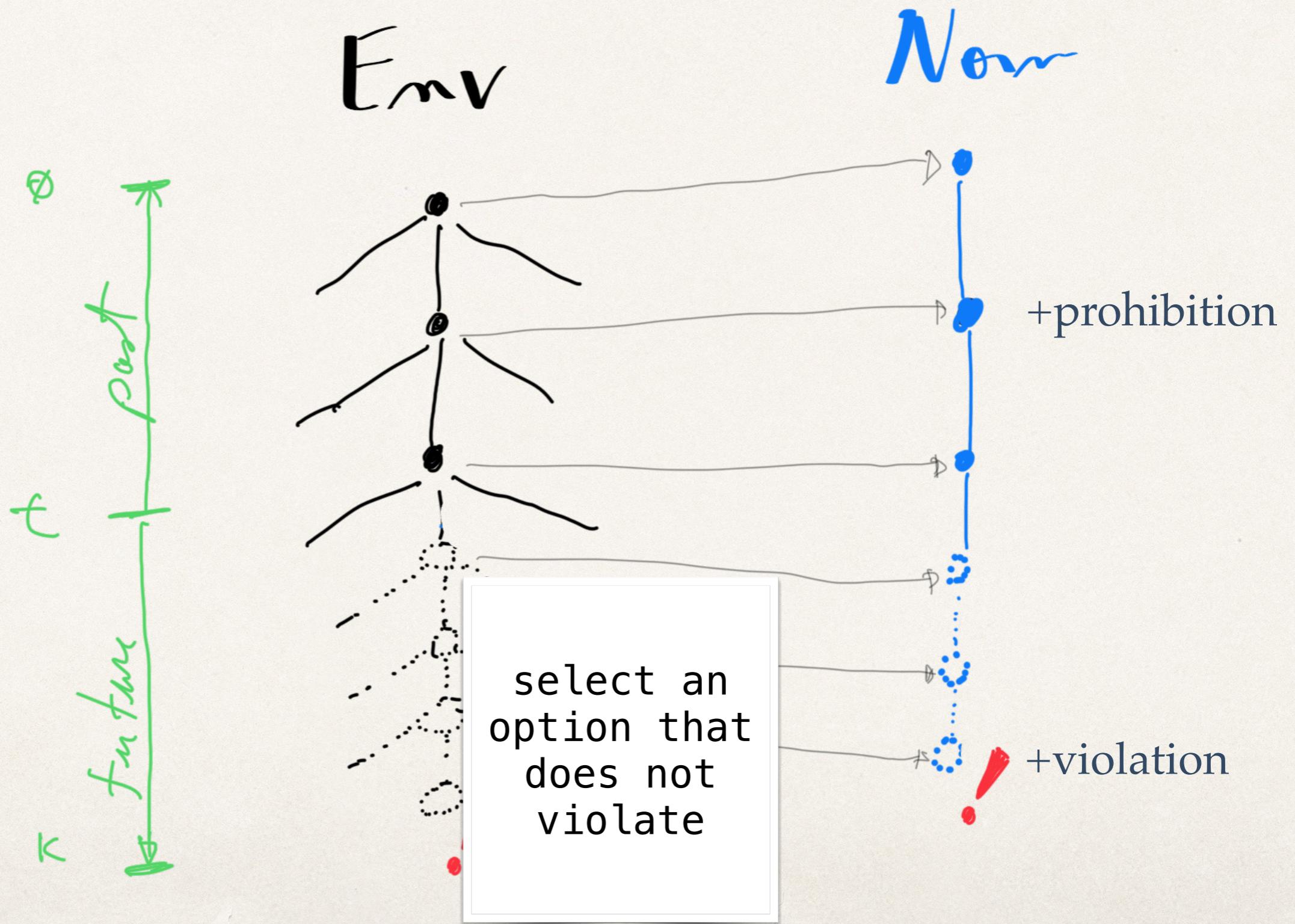
- ✿ Measure:
  - ✿ efficiency (steps to achieve a goal),
  - ✿ (computational) cost,
- ✿ Varies
  - ✿ uncertainty of the environment
  - ✿ how far to look ahead

# Preliminary results

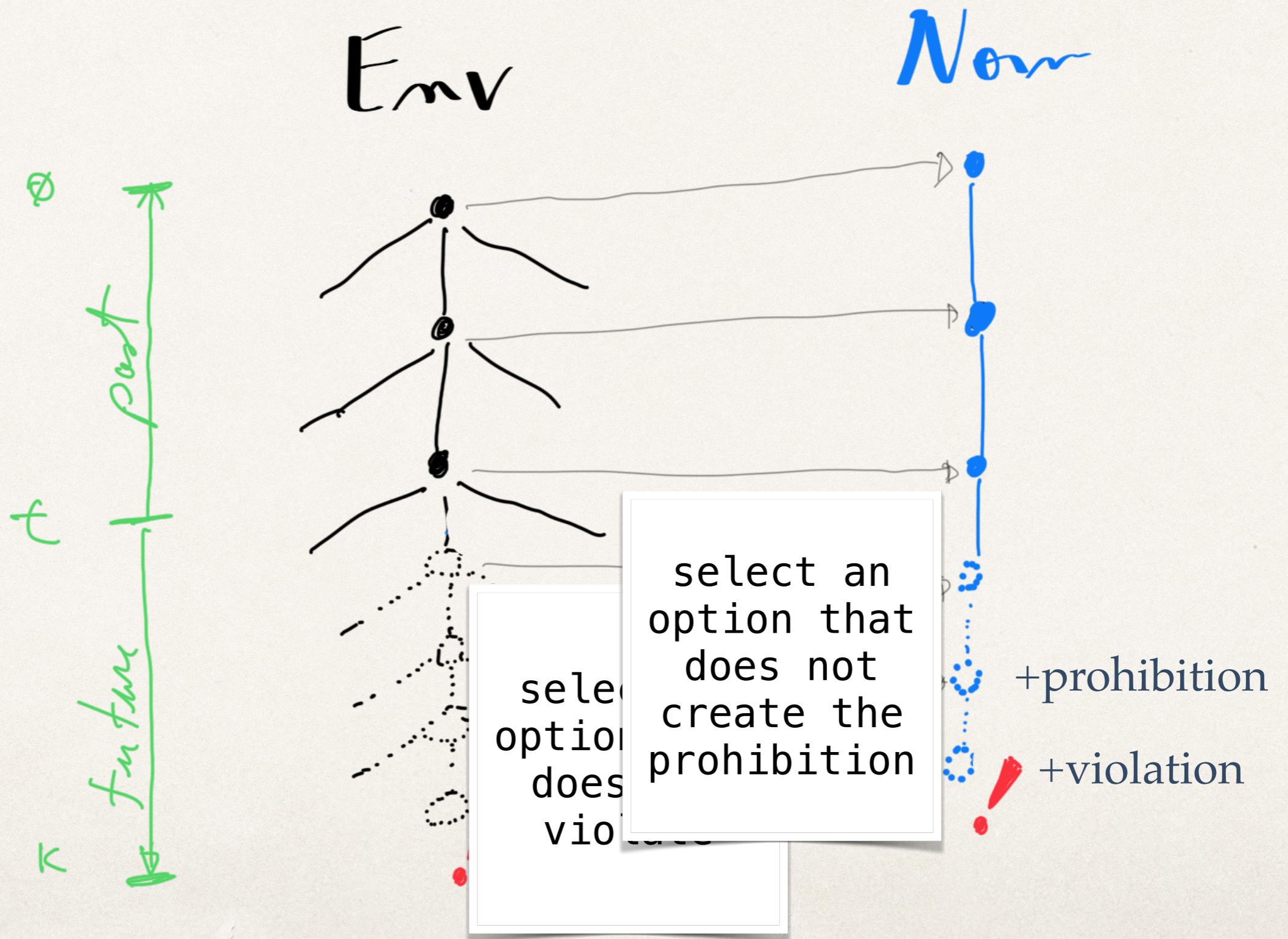
updated  
HERE



# Norm Violation



# Norm Violation



# Normative State ( $N$ )

---

next normative state:

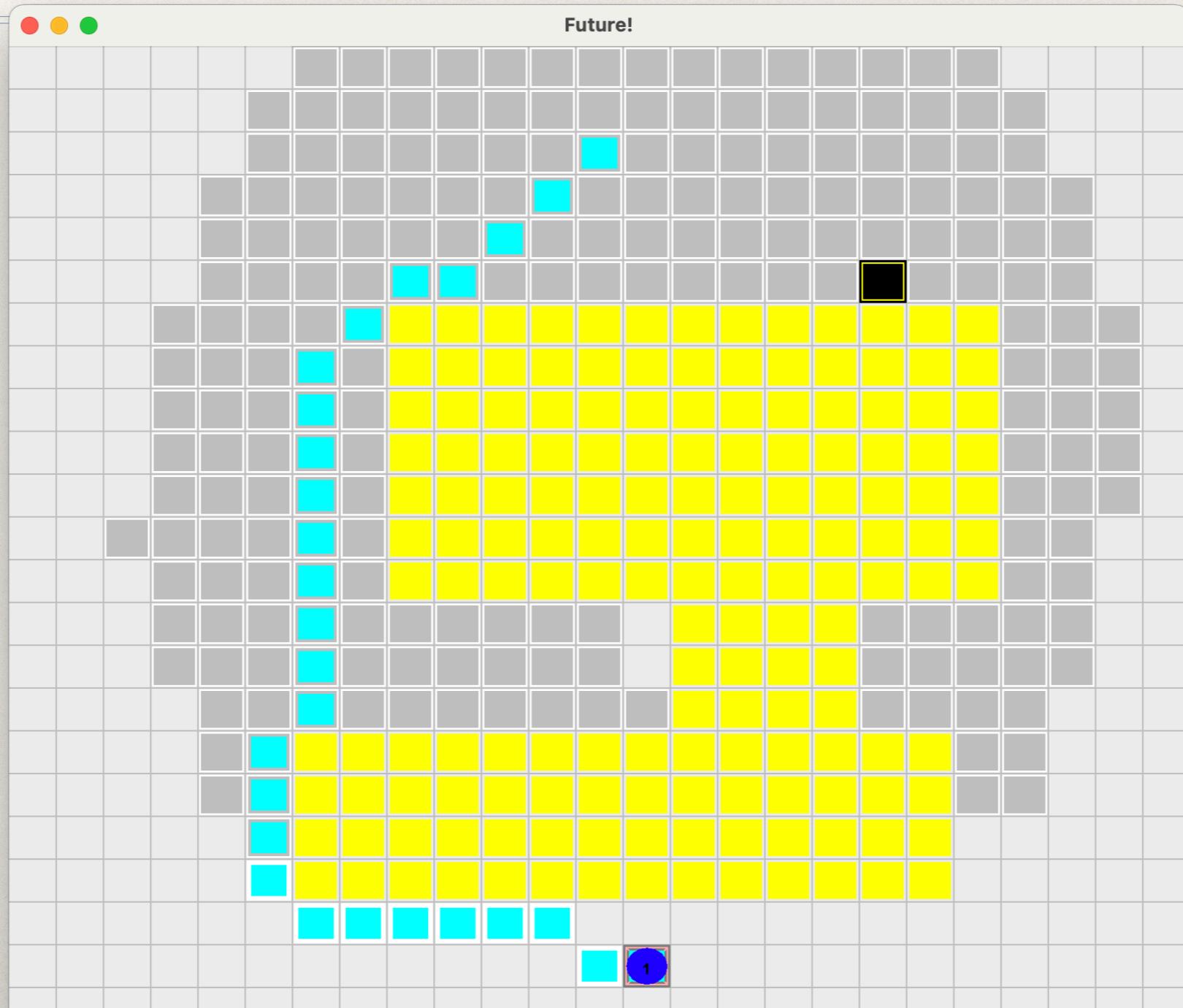
$$nns : S \times N \rightarrow N$$

$n_{t+1} = nns(s_{t+1}, n_t)$  is the next normative state

a normative state consists of a set of

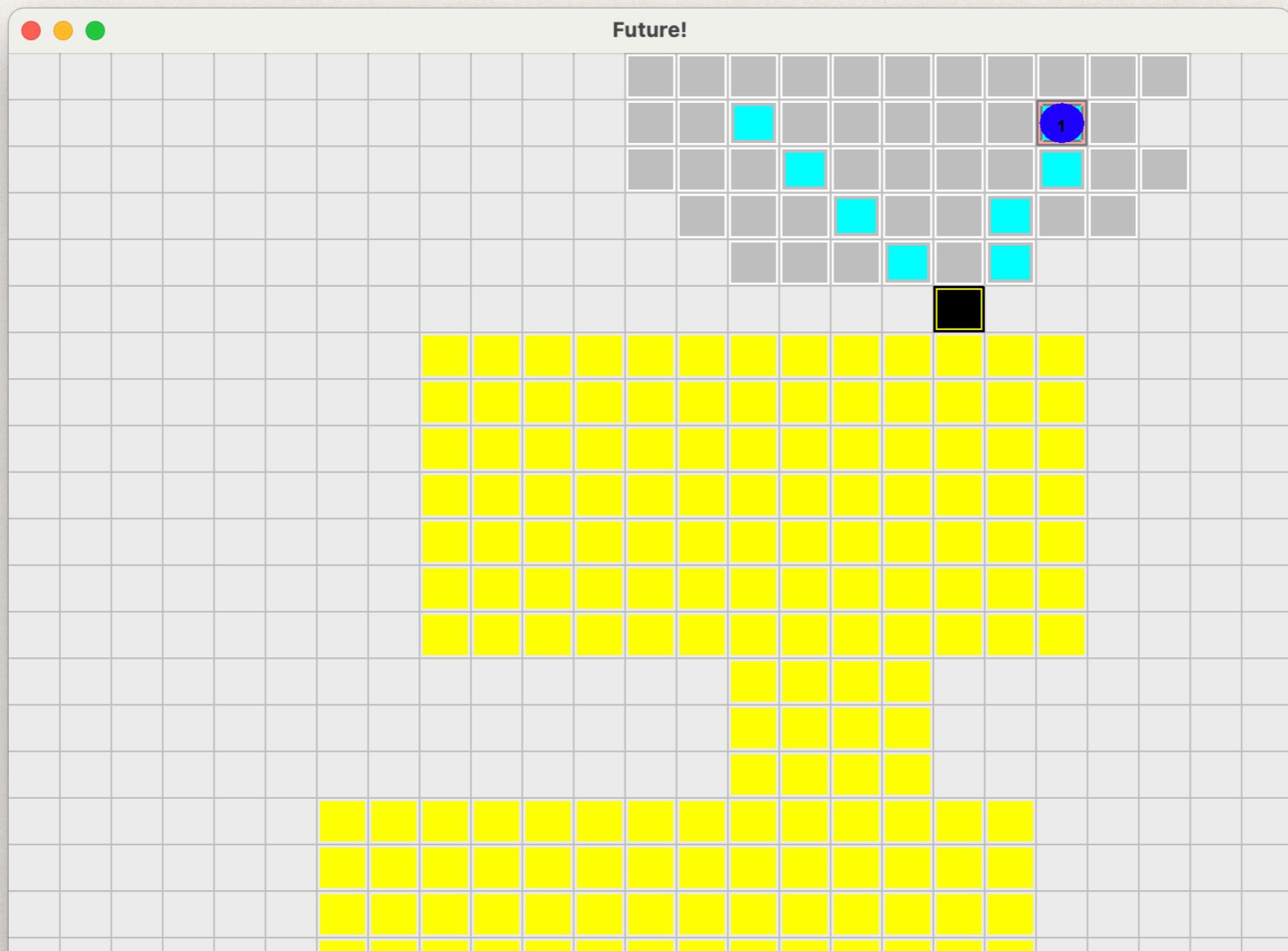
- active obligations, prohibitions, ...
- unfulfillments

# Jason(F) 1.6: Normative Future Reasoning



```
norm n1 : true  
-> prohibition(A,n1, pos(A,X,Y) & ltz(X,Y), false).
```

# Jason(F) 1.6: Normative Future Reasoning



```
norm n2: portal(PX,PY) & destination(DX,DY)
-> obligation(A, n2, visited(A,PX,PY), pos(A,DX,DY) ).
```

# Notes about norms

---

- ✿ A new kind of “future problem” that can not be determined simply by an environment or agent state
  - ✿ we require the progression of **normative states** too
- ✿ It avoids the case of a simple agent that realises “ops, my next action will violate a norm”
  - ✿ wasting a lot of previous effort
  - ✿ we **foresee** that situation before committing to a (long term) goal.
- ✿ Future work:
  - ✿ decide to violate! considering sanctions consequences

# Next Steps

---

- ❖ Relax assumptions
- ❖ More scenarios (more complex and realistic)
- ❖ Running in background
- ❖ Opportunistic options
  - oracle: “your policy is ok, but you could do better”
  - “if I will be wet in the future, then take an umbrella”
- ❖ Consider backtrack (options in the past)
- ❖ Other types of problem
- ❖ Shared matrix