BDI on Time

“the future of BDI is in the future”

first week of July 2023

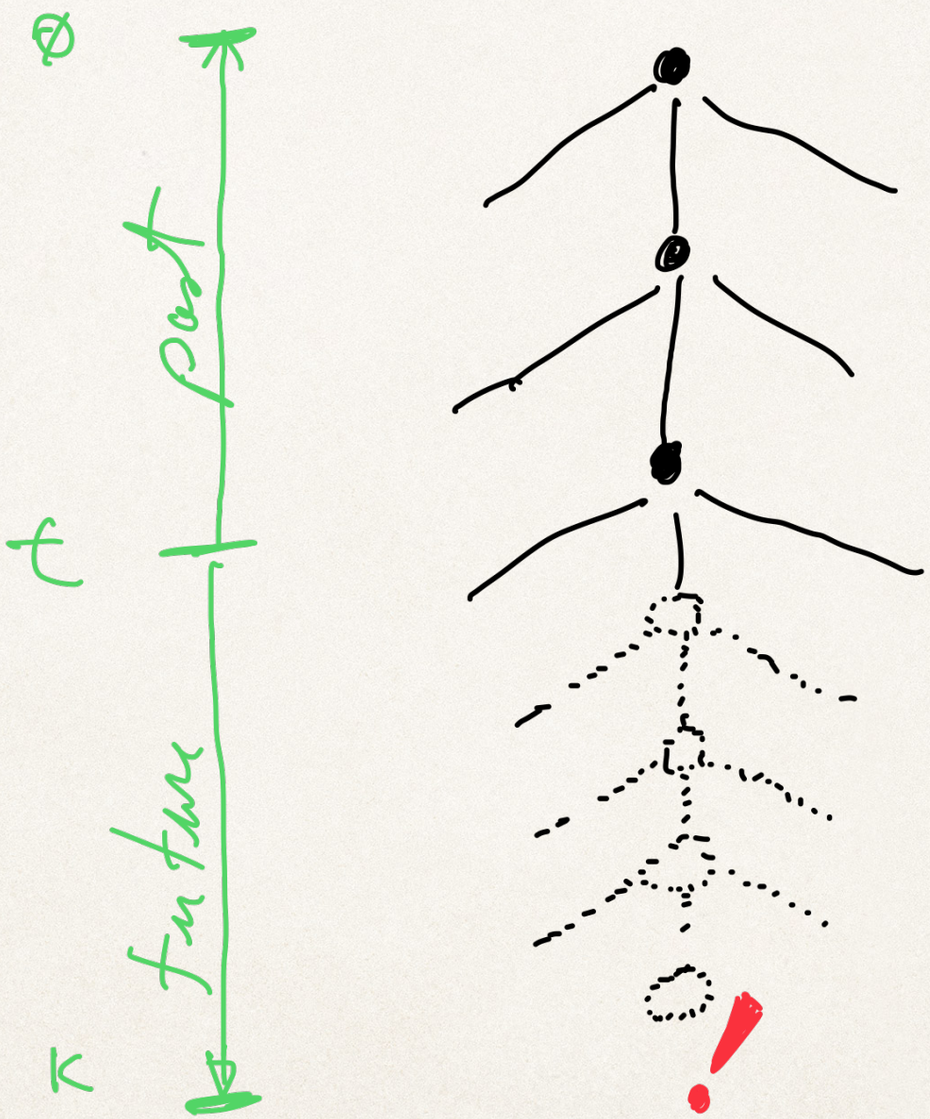
Agenda

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

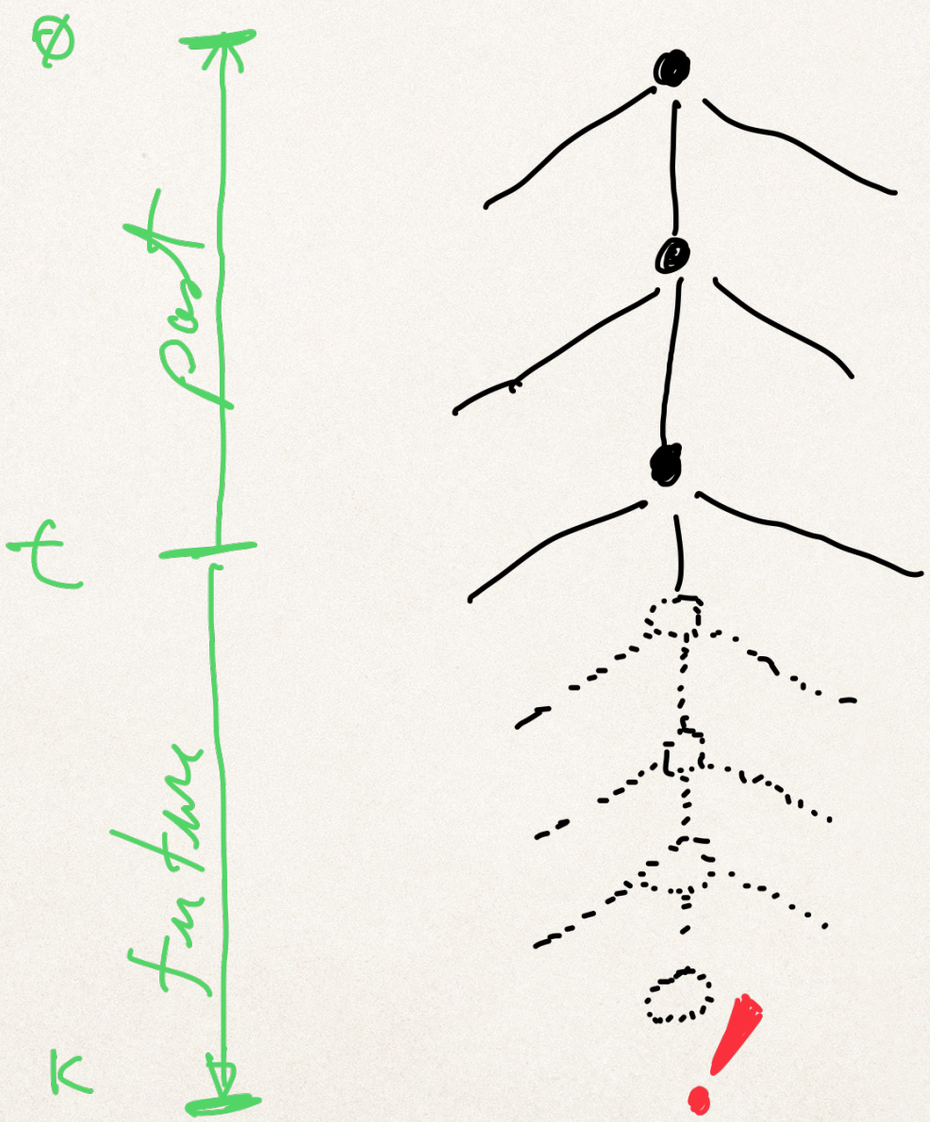
Motivations To Consider Future & BDI

- BDI agents may do better (rational) decisions by *looking ahead*
- Foresee problems
- Realize better options
- …

Future Simulation



Future Simulation



environment transitions

$$e : S \times A \to S$$

agent policy (as ordered options)

$$\pi : S \to A^n$$

future state at k

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

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

$$s' = e(s, \pi(s)_1)$$

Future Simulation

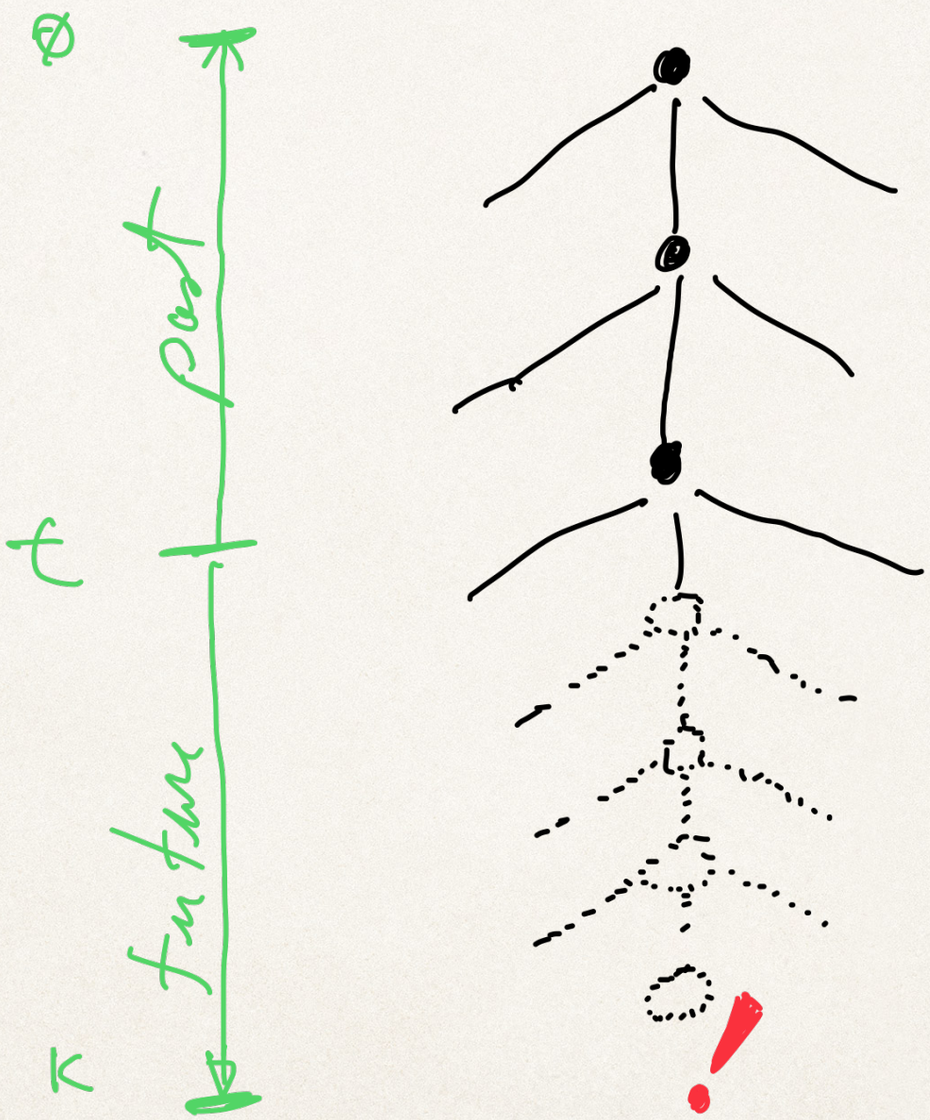
fully observable

deterministic

single-agent

static

discret



envi

e :

agent policy (as ordered options)

$$\pi : S \to A^n$$

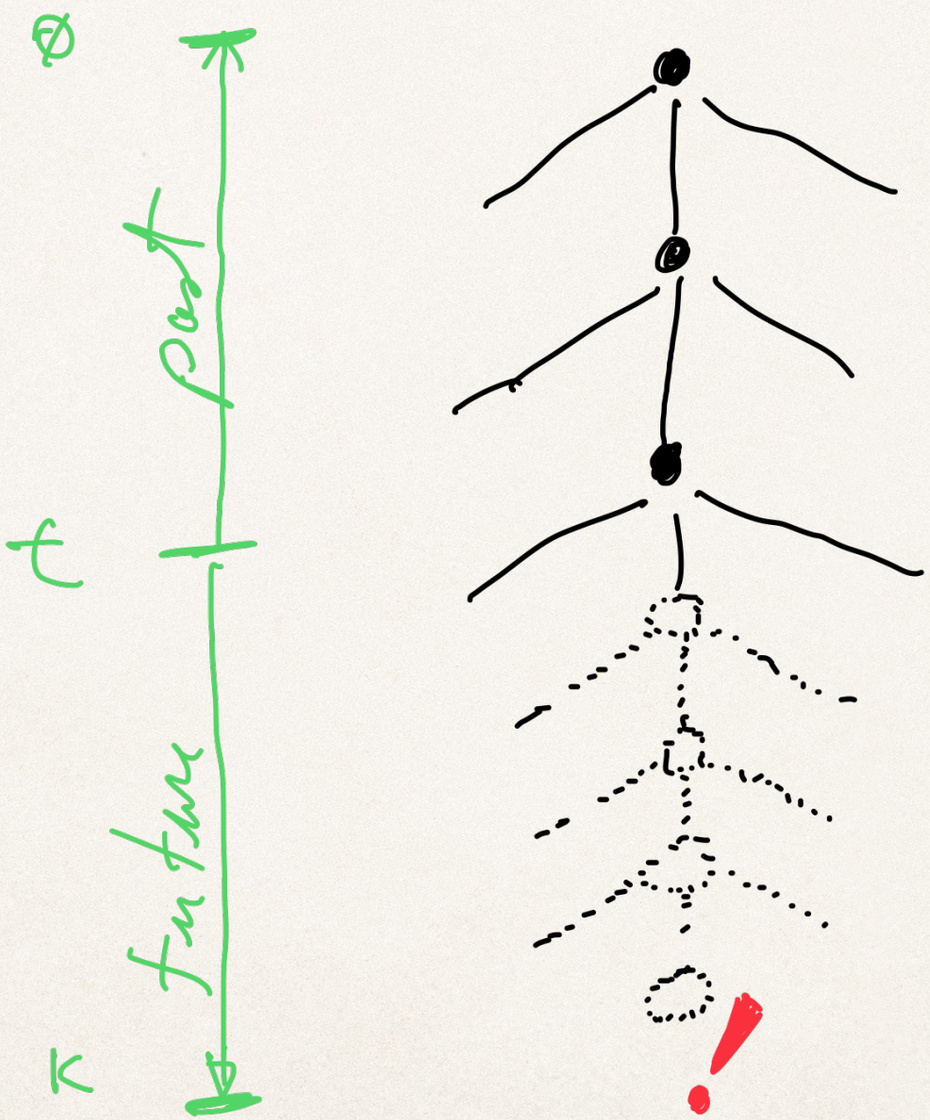
future state at k

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

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

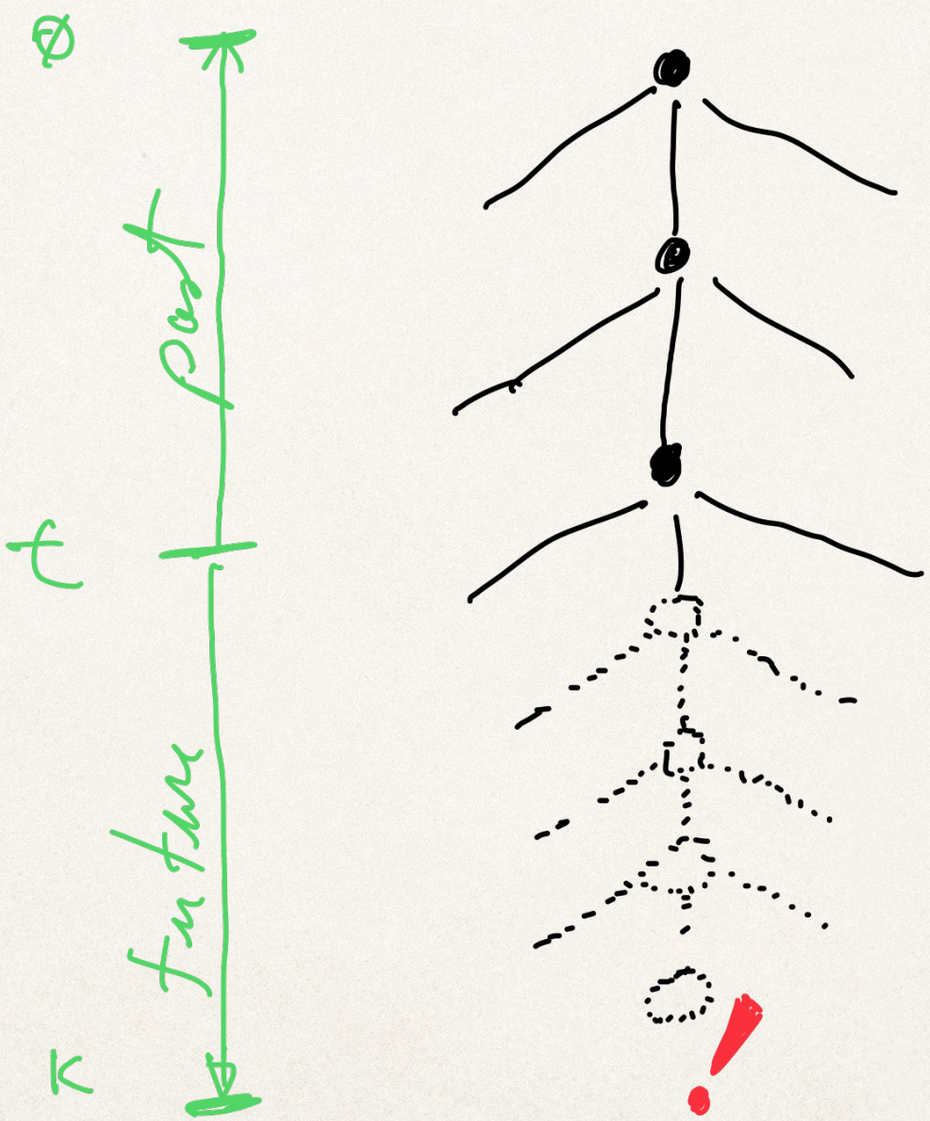
$$s' = e(s, \pi(s)_1)$$

Types of Problem



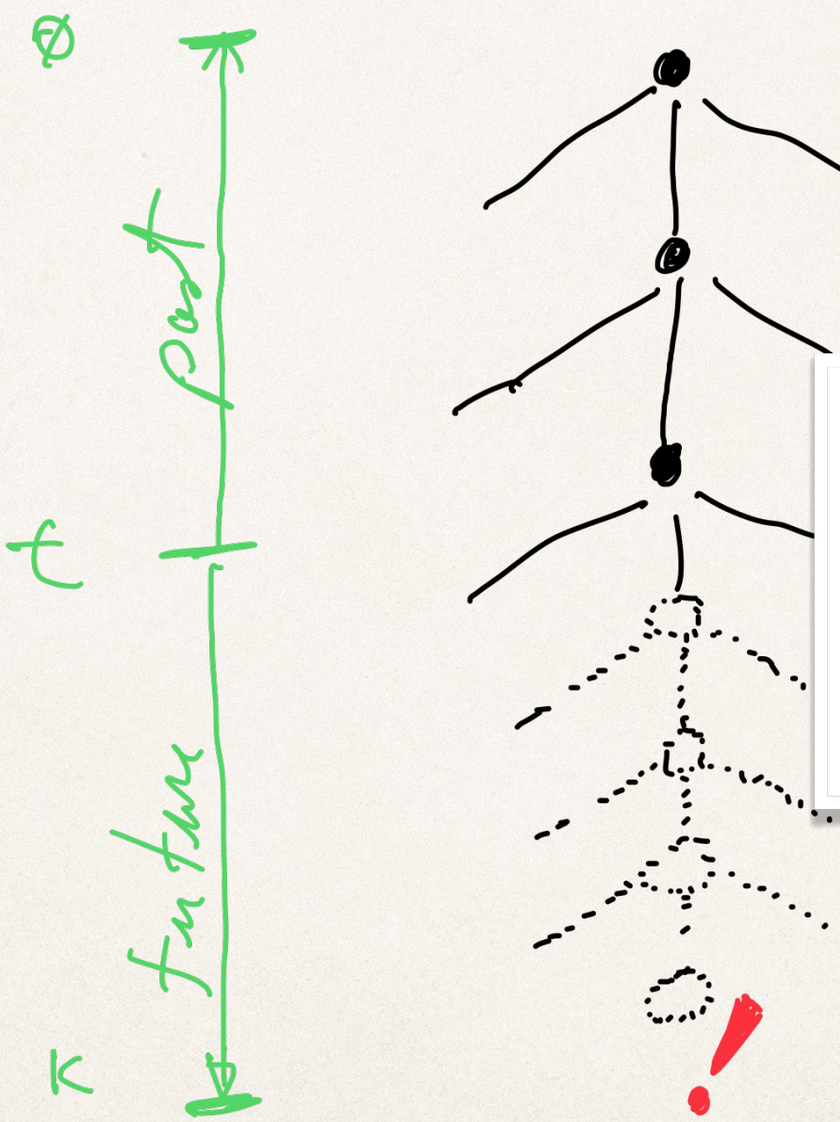
- internal:
  - no option, goal not achieved
  - norm / principle violation
  - ethical issues
- external:
  - undesired state
  - ethical issues
- …

What To Do?



- stop
- reconsider options
  - now (t)
  - near the problem (k)
  - in the past (0..t)
    (requires backtrack of actions)
- give up the intention

What To Do?



- stop
- reconsider options

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

actions)

- give up the intention

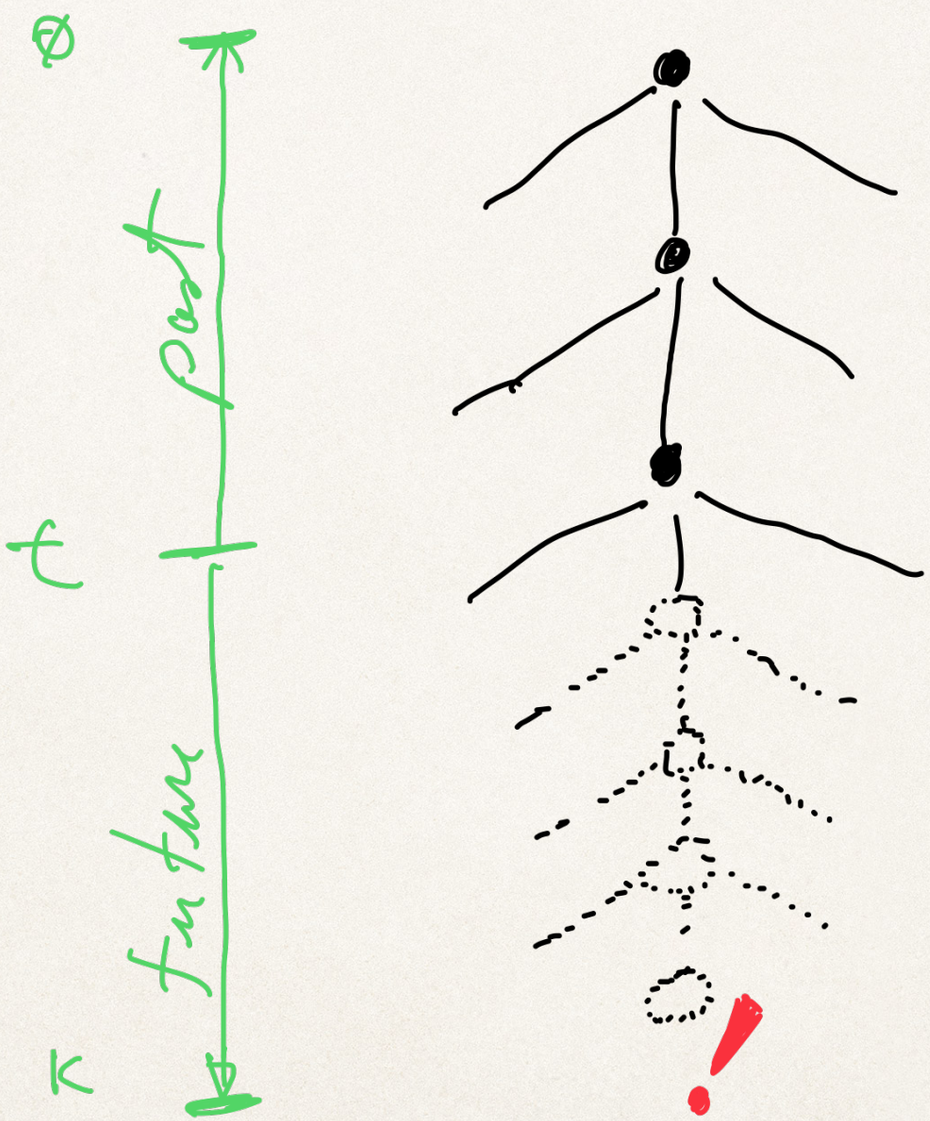
What To Do?

- stop
- reconsider options

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

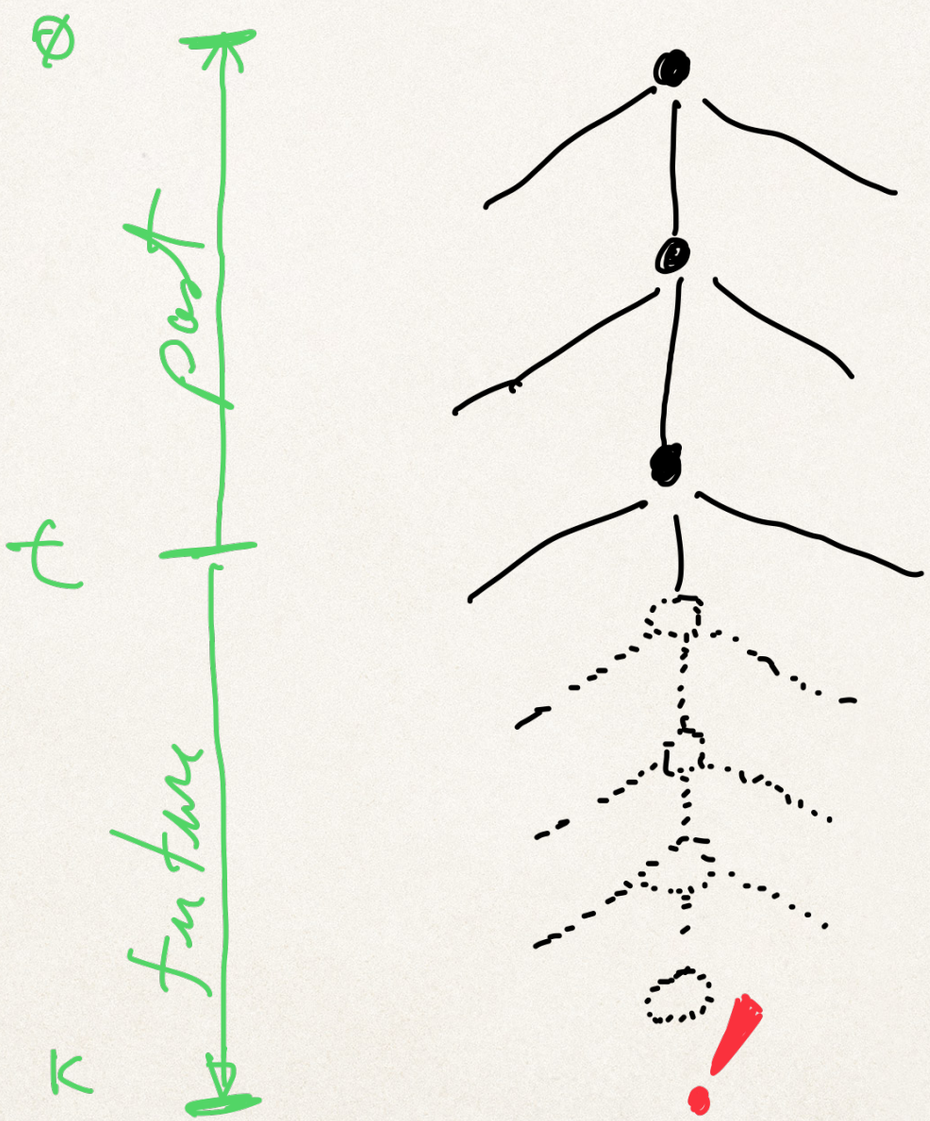
select an option with “nice” future [without problems]

(Initial) Delimitations



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

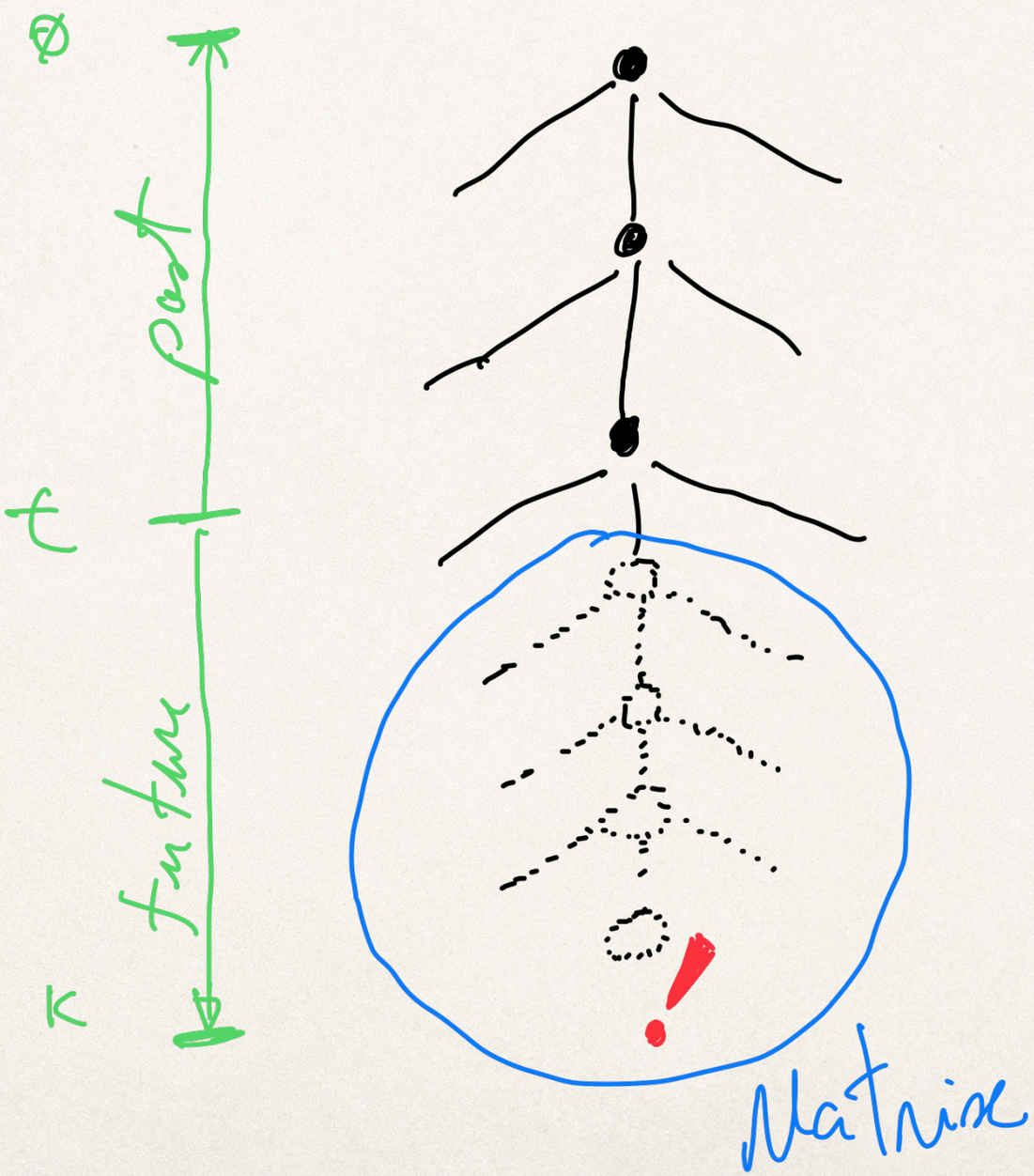
(Initial) Delimitations



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

```
move to experiments to
help to answer
```

Jason(F)

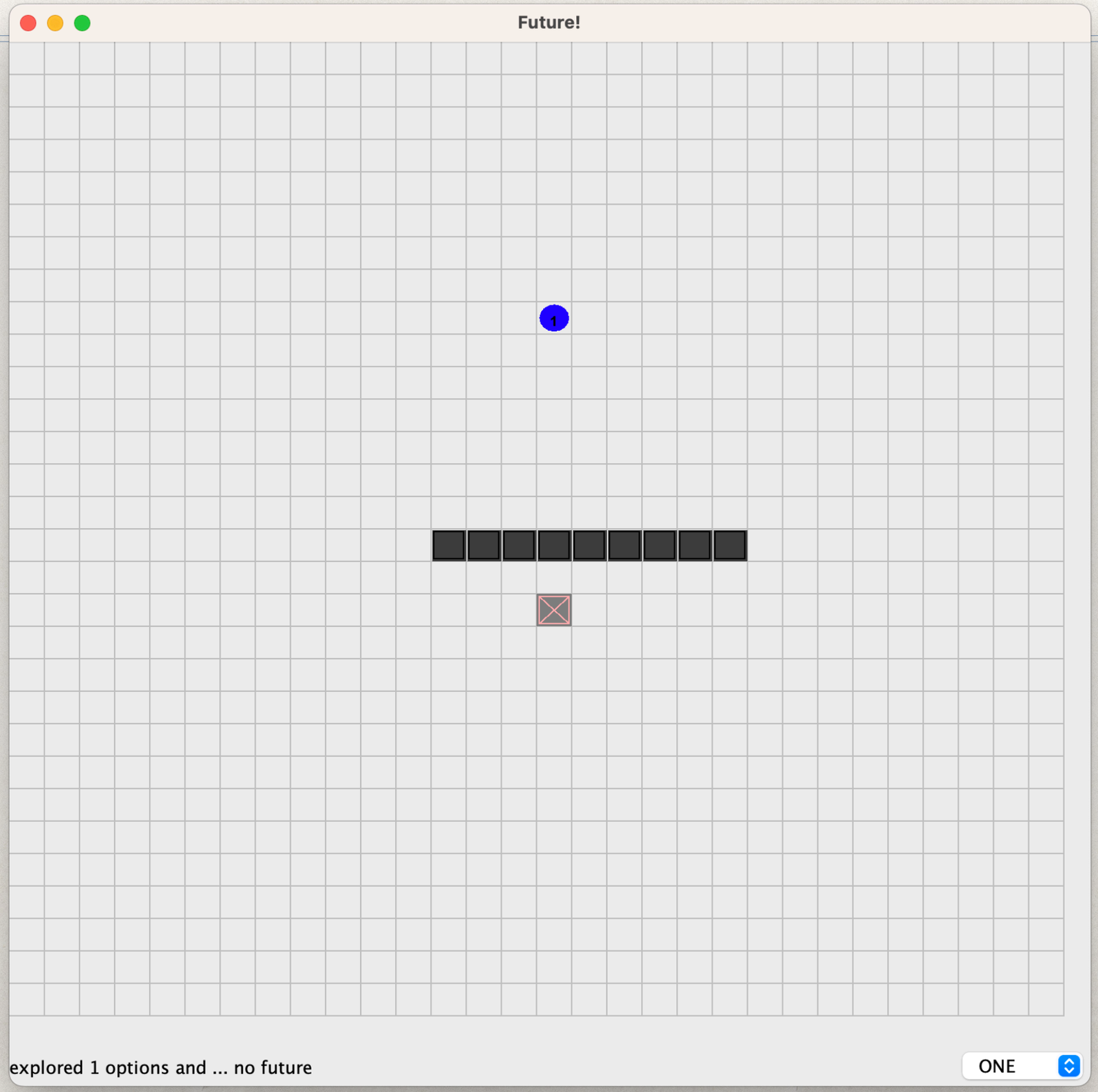


execution modes:
- normal
- **matrix** mode
  (simulated environment)

at time *t*:
- clone itself into another agent
- run it in the matrix
- until success, failure, timeout

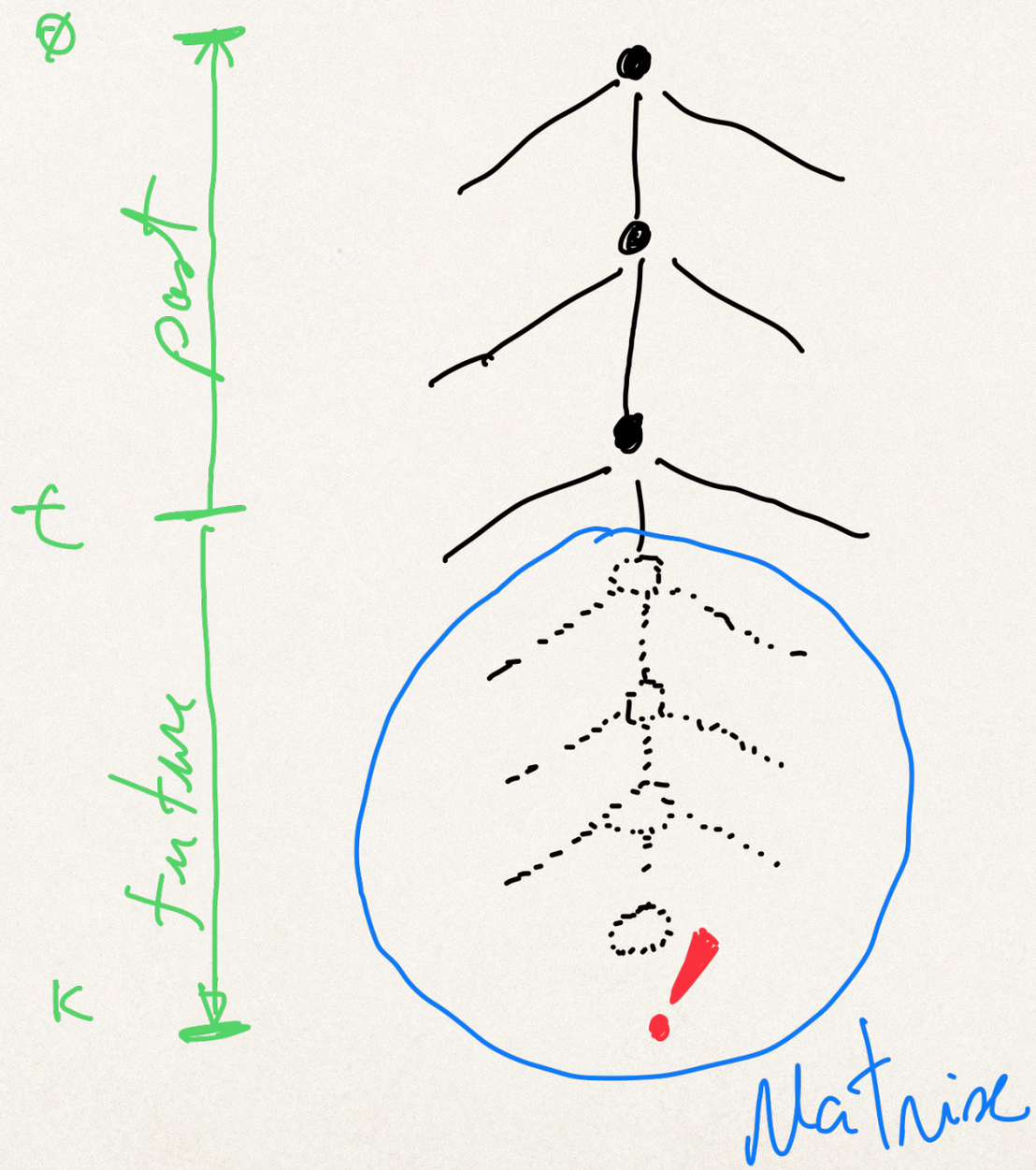
branching is given by plan options
[not actions]

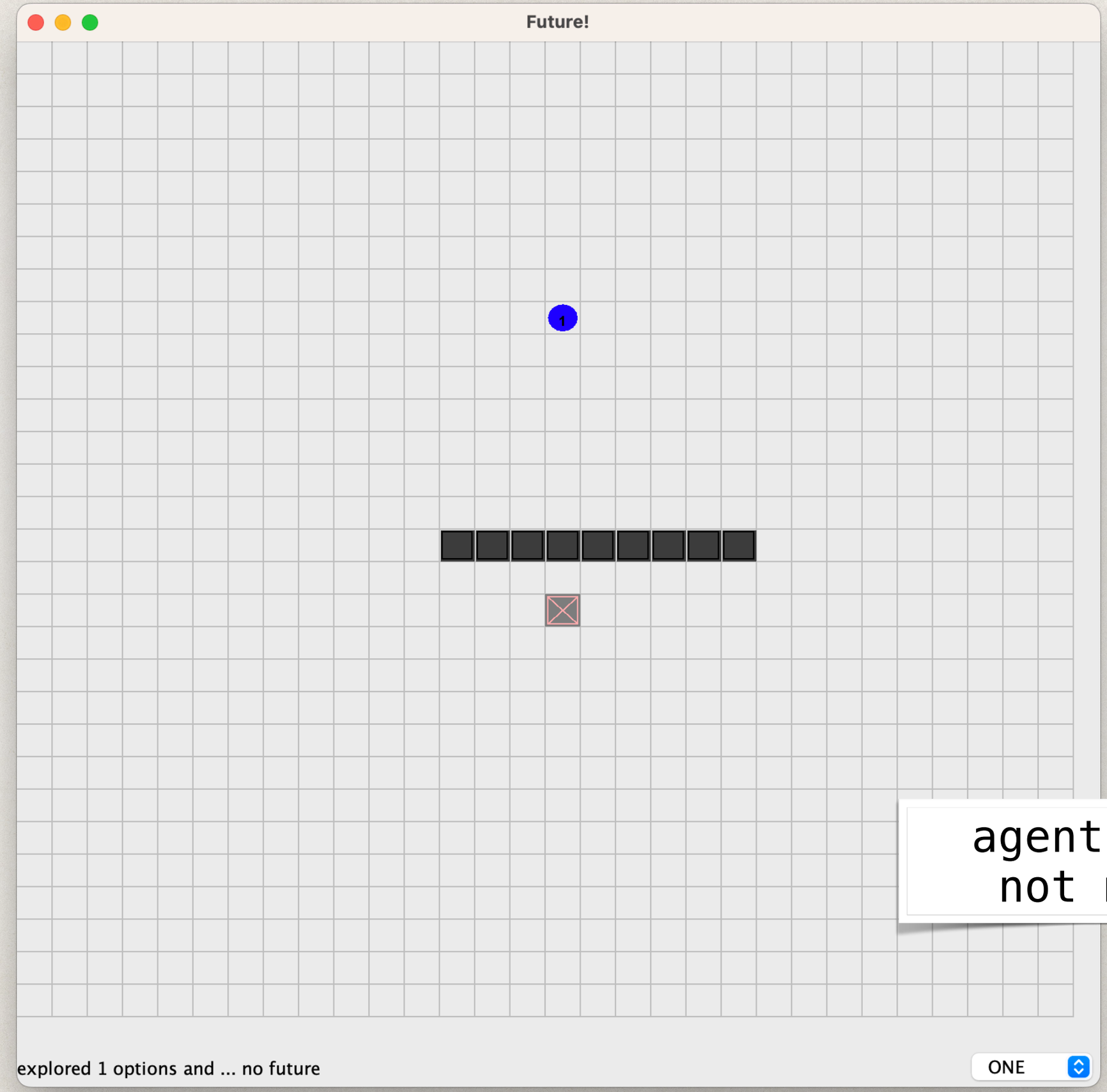
Grid Scenario



```
+destination(X,Y) <- !pos(X,Y). // create a goal when my destination is perceive
-destination(X,Y) <- .drop_all_desires. // drop everything if my destination is

@  [preference(0)] +!pos(X,Y) : pos(X,Y).
@s [preference(D)] +!pos(X,Y) : ok(s) & distance(s ,D) <- s;  !pos(X,Y).
@sw[preference(D)] +!pos(X,Y) : ok(sw)& distance(sw,D) <- sw; !pos(X,Y).
@se[preference(D)] +!pos(X,Y) : ok(se)& distance(se,D) <- se; !pos(X,Y).
@w [preference(D)] +!pos(X,Y) : ok(w) & distance(w ,D) <- w;  !pos(X,Y).
@e [preference(D)] +!pos(X,Y) : ok(e) & distance(e ,D) <- e;  !pos(X,Y).
@n [preference(D)] +!pos(X,Y) : ok(n) & distance(n ,D) <- n;  !pos(X,Y).
@nw[preference(D)] +!pos(X,Y) : ok(nw)& distance(nw,D) <- nw; !pos(X,Y).
@ne[preference(D)] +!pos(X,Y) : ok(ne)& distance(ne,D) <- ne; !pos(X,Y).

// checks if go to some direction is possible (free cell)
ok(D) :- next(D,X,Y) & free(X,Y).

next(s ,X  ,Y+1) :- pos(X,Y). // my next location if doing south
next(sw,X-1,Y+1) :- pos(X,Y).
next(se,X+1,Y+1) :- pos(X,Y).
next(w ,X-1,Y  ) :- pos(X,Y).
next(e ,X+1,Y  ) :- pos(X,Y).
next(n ,X  ,Y-1) :- pos(X,Y).
next(nw,X-1,Y-1) :- pos(X,Y).
next(ne,X+1,Y-1) :- pos(X,Y).

free(X,Y) :- X >= 0 & Y >= 0 & w_size(W,H) & X < W & Y < H & not obstacle(X,Y).
distance(Dir,Dist) :- next(Dir,X,Y) & destination(GX,GY) &
                      Dist = math.sqrt( (X-GX)**2 + (Y-GY)**2 ).
```

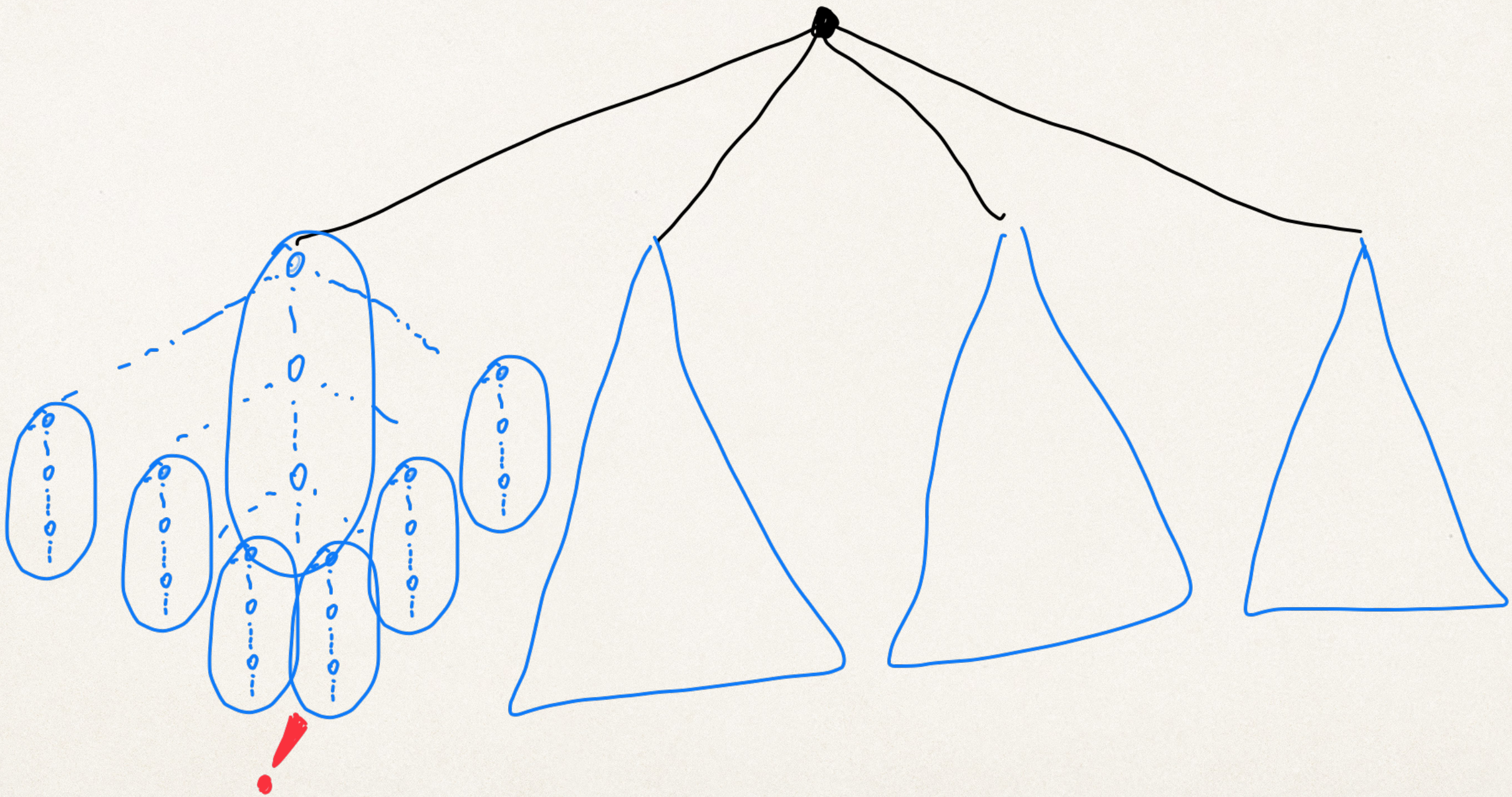
Strategy ONE



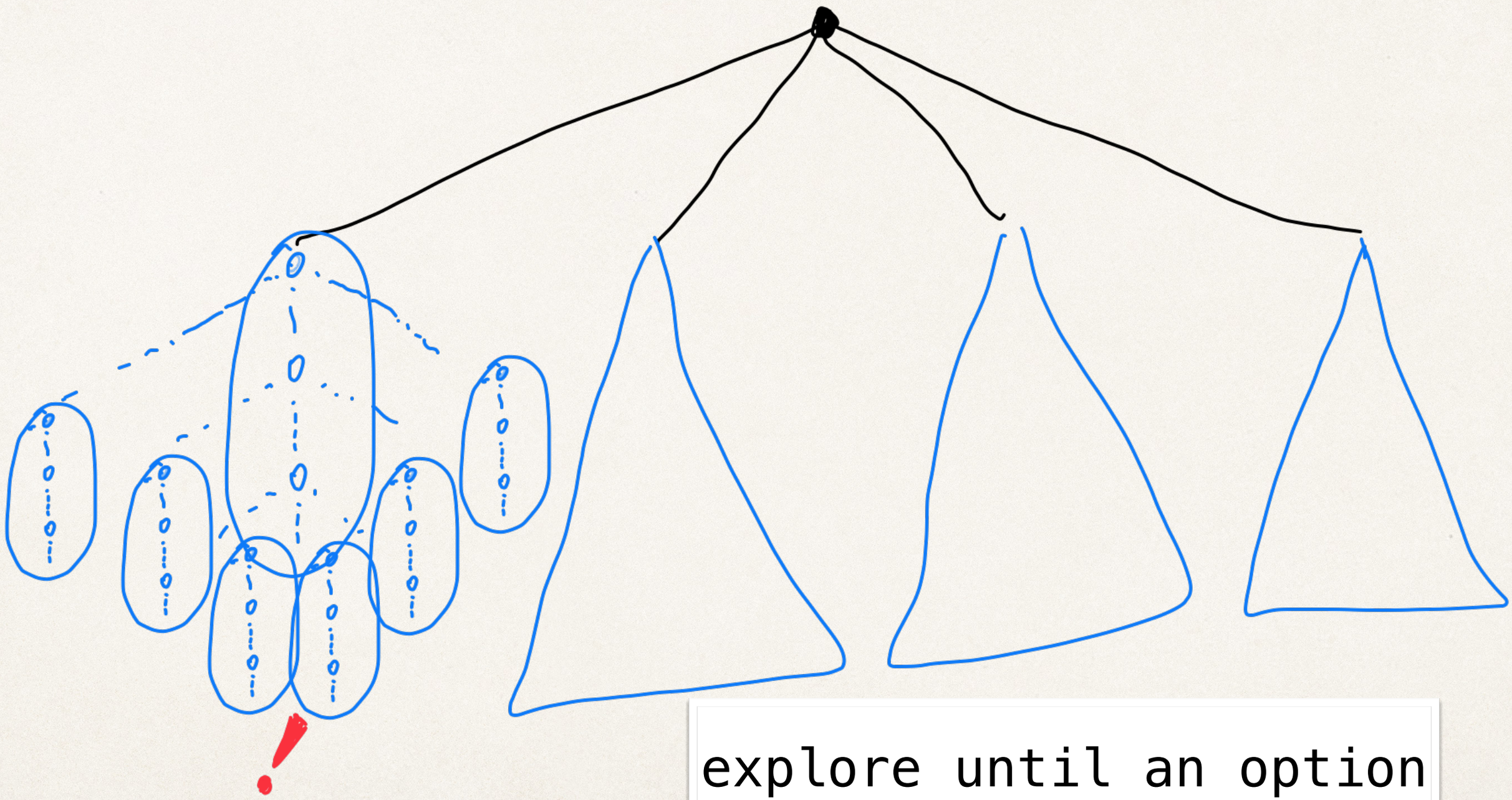


agent does
not move

Strategy SOLVE_P —Search Tree

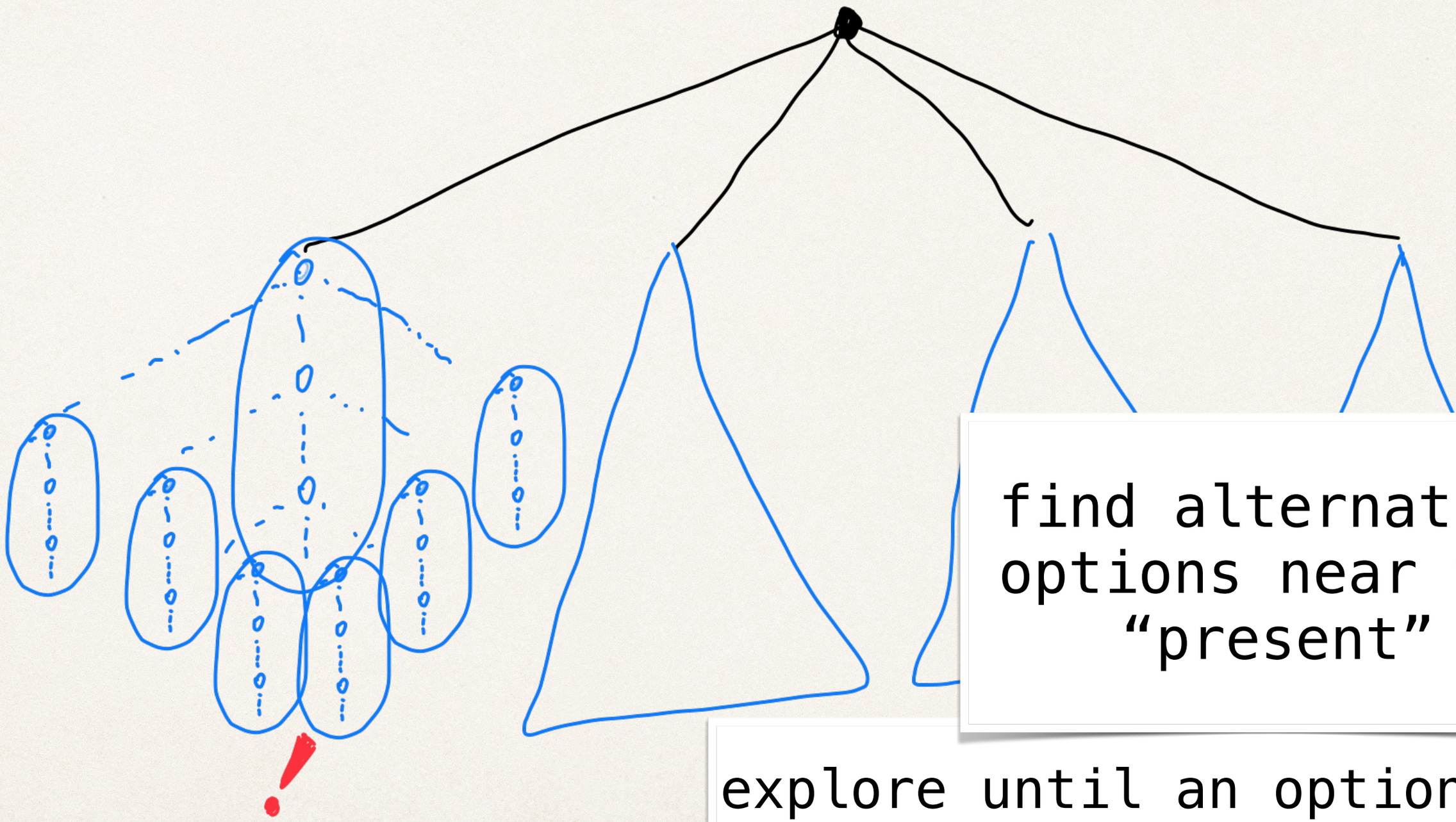


Strategy SOLVE_P —Search Tree



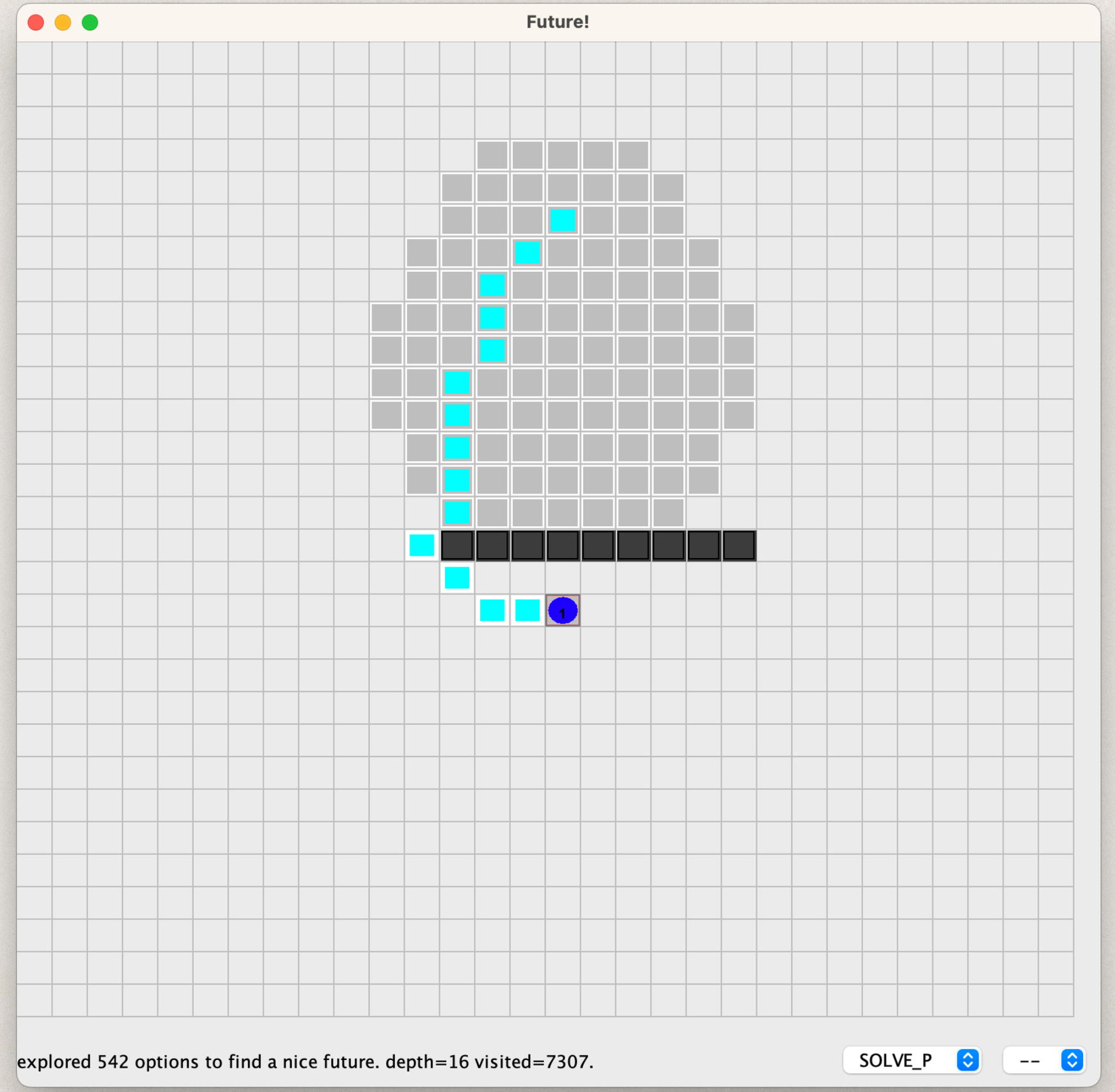
explore until an option leads to a good future

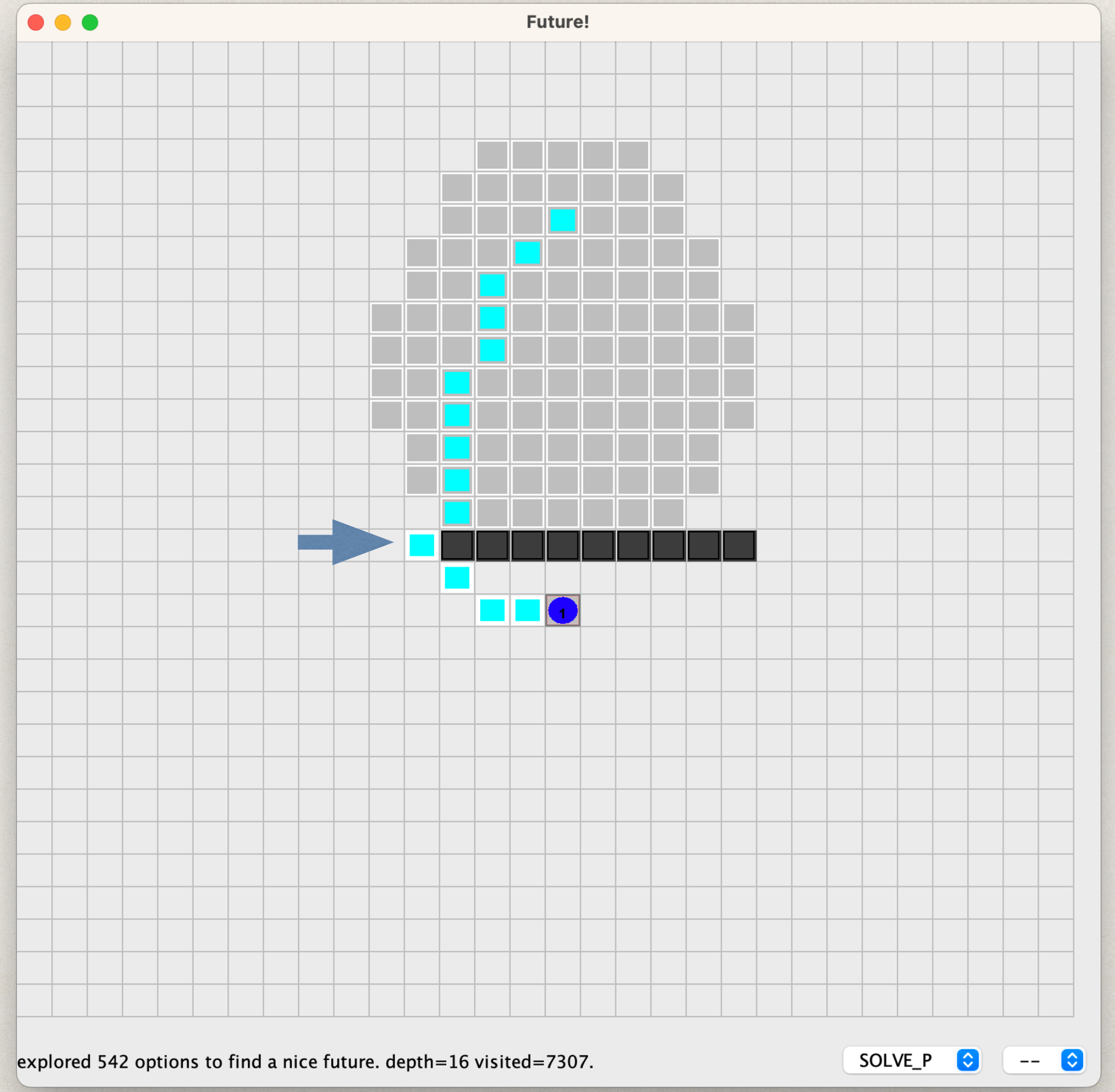
Strategy SOLVE_P —Search Tree

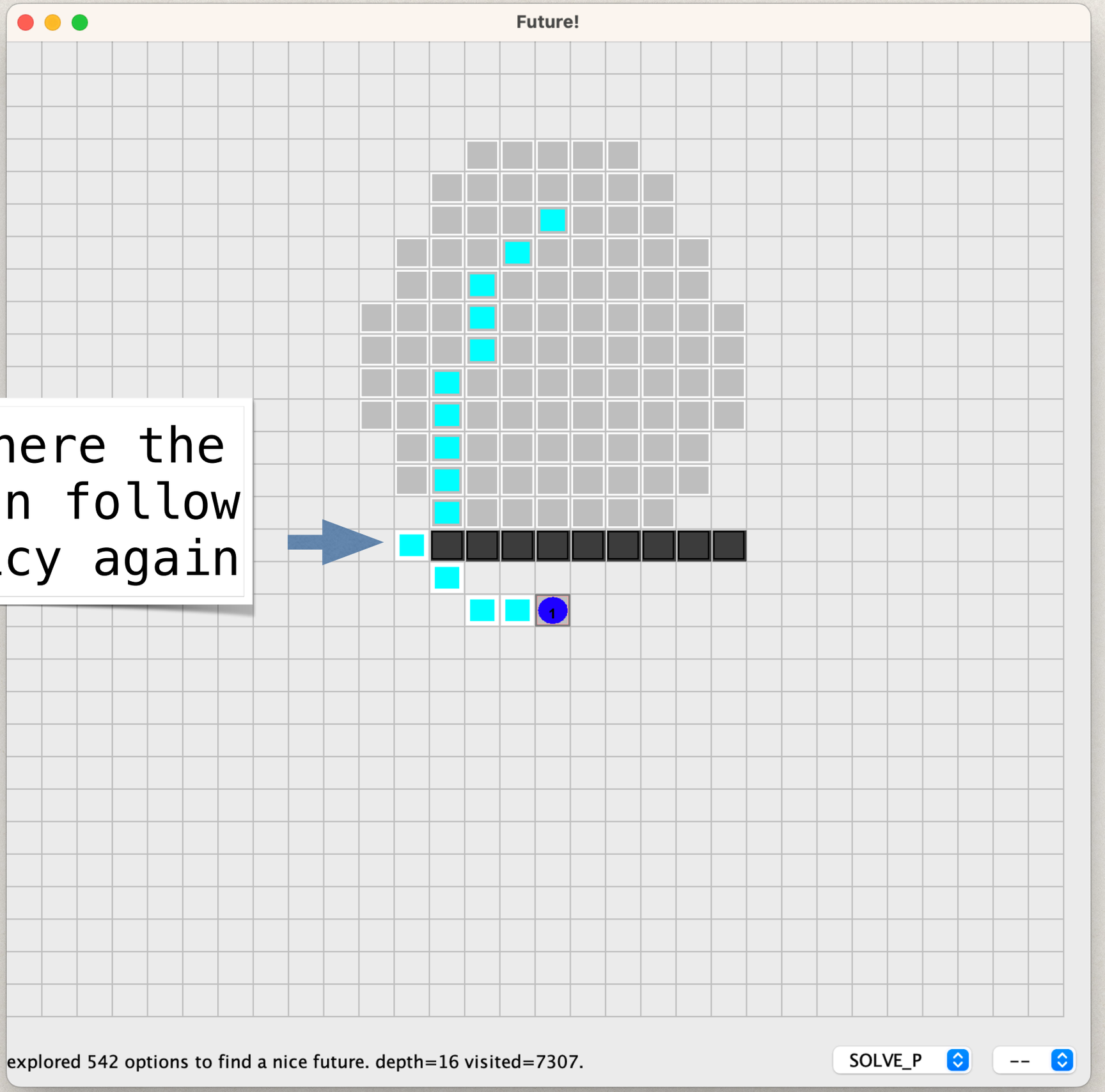


find alternative options near the “present”

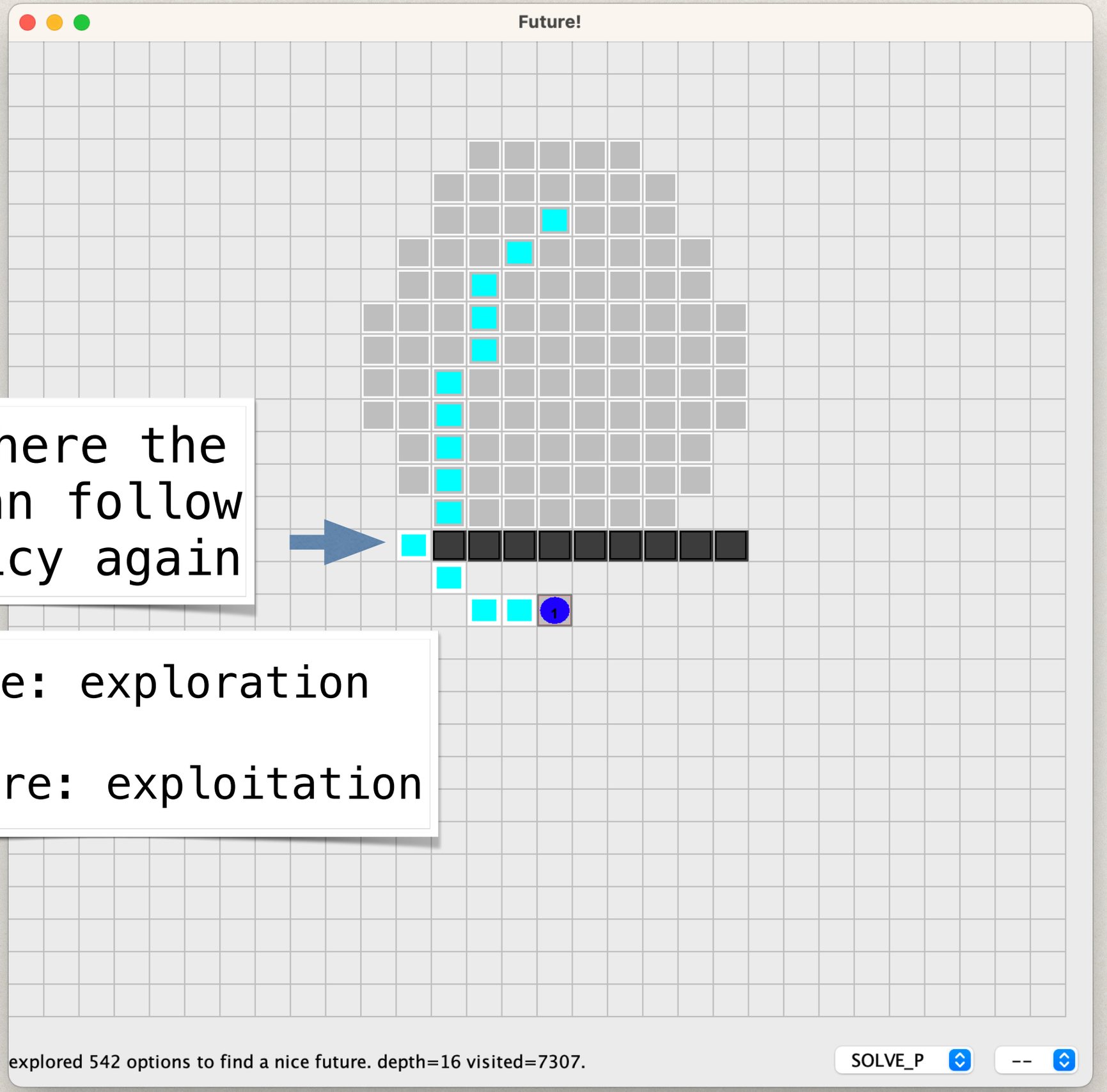
explore until an option leads to a good future


Future!
1
explored 542 options to find a nice future. depth=16 visited=7307.
SOLVE_P
--


Future!
1
explored 542 options to find a nice future. depth=16 visited=7307.
SOLVE_P
--



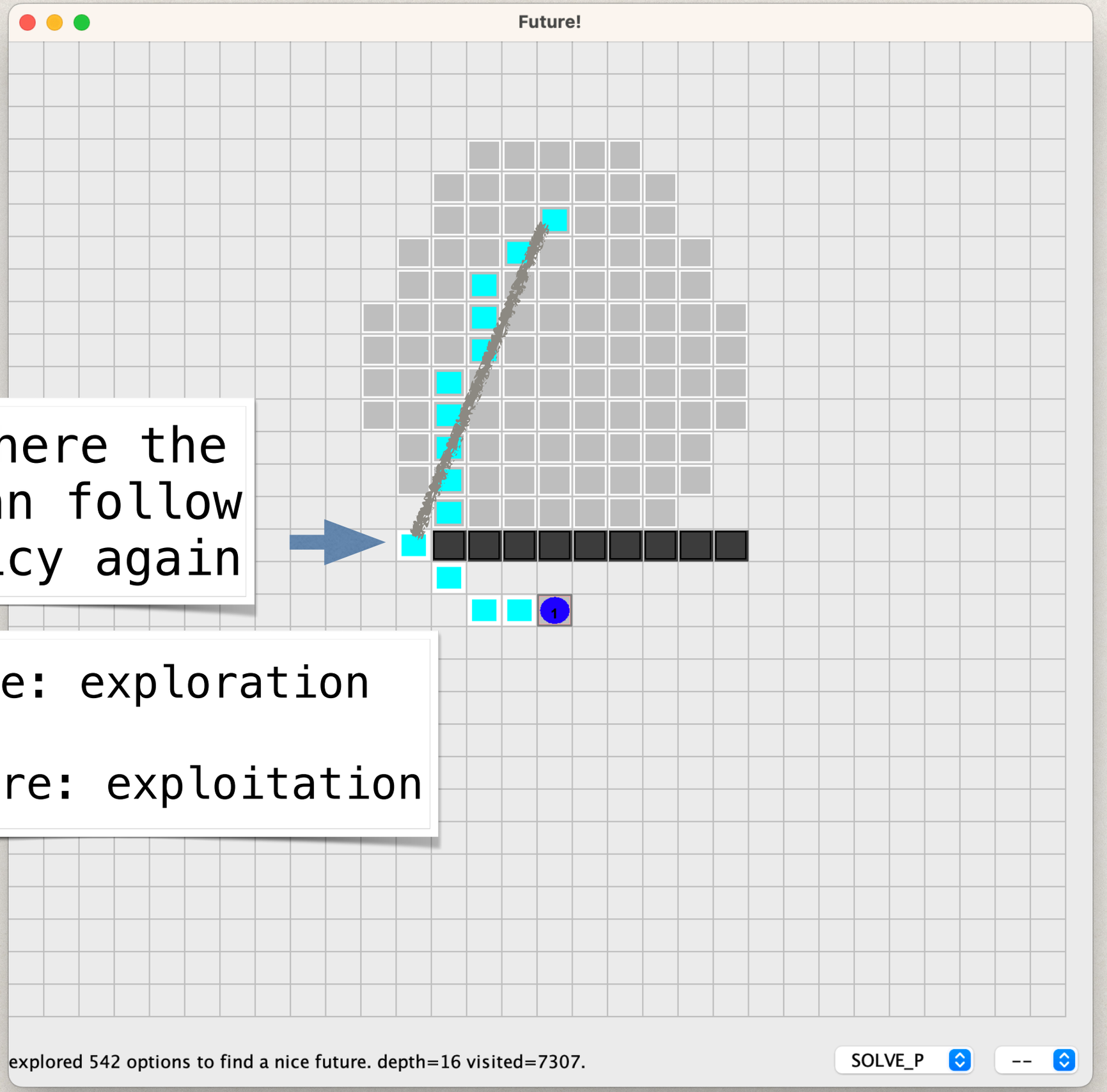
point where the agent can follow its policy again



point where the agent can follow its policy again

before: exploration

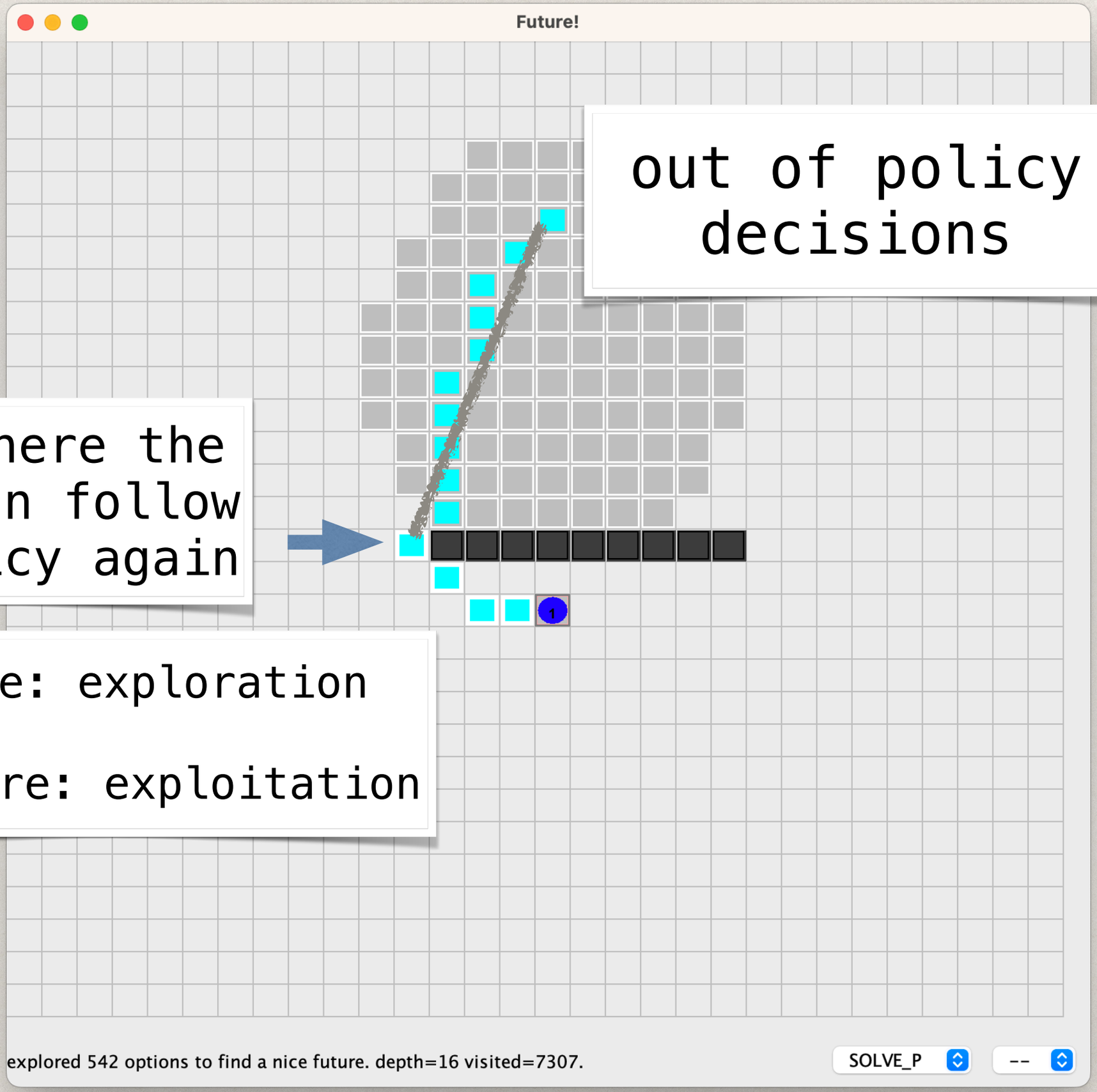
from here: exploitation



point where the agent can follow its policy again

before: exploration

from here: exploitation



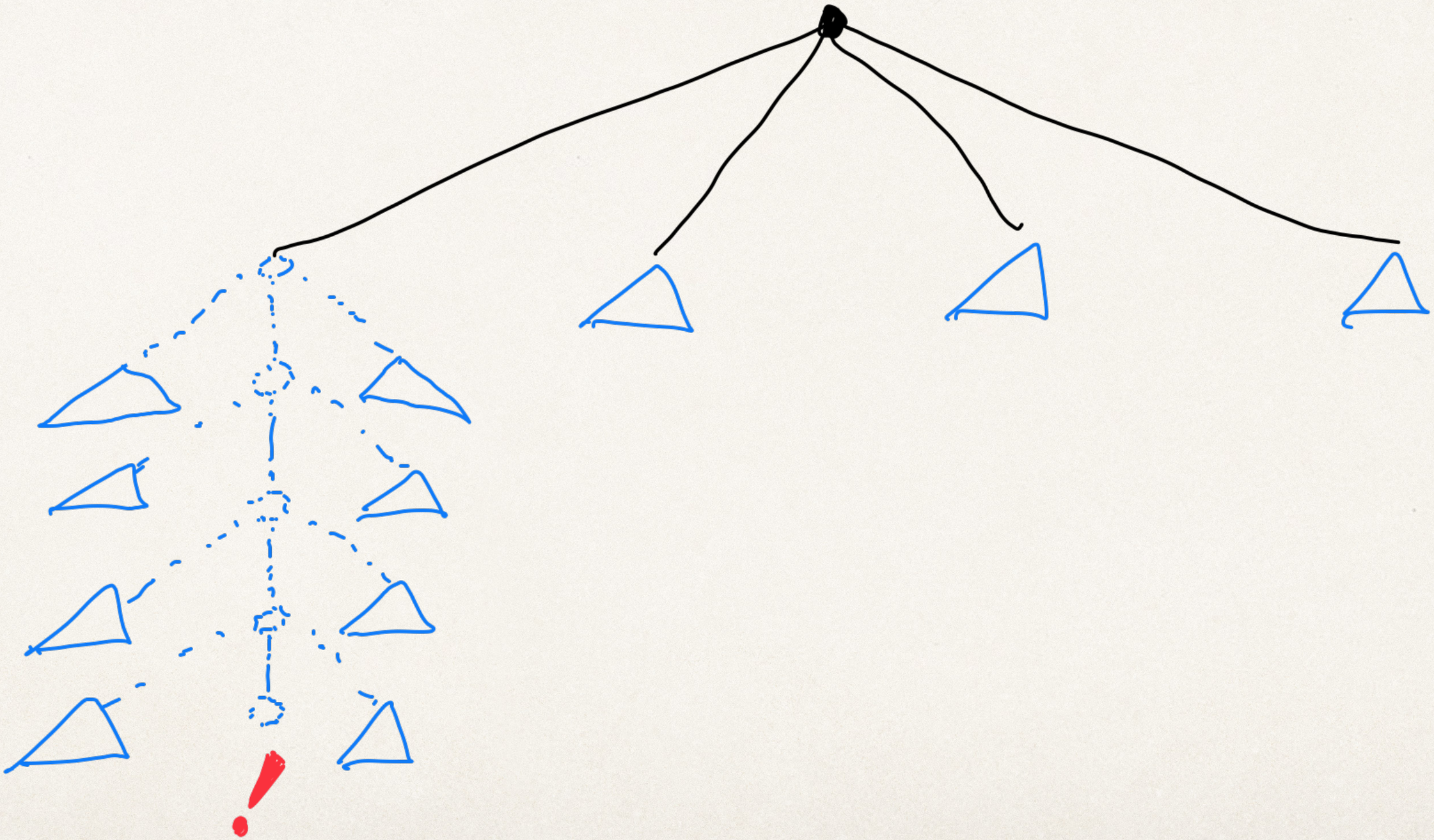
out of policy decisions

point where the agent can follow its policy again

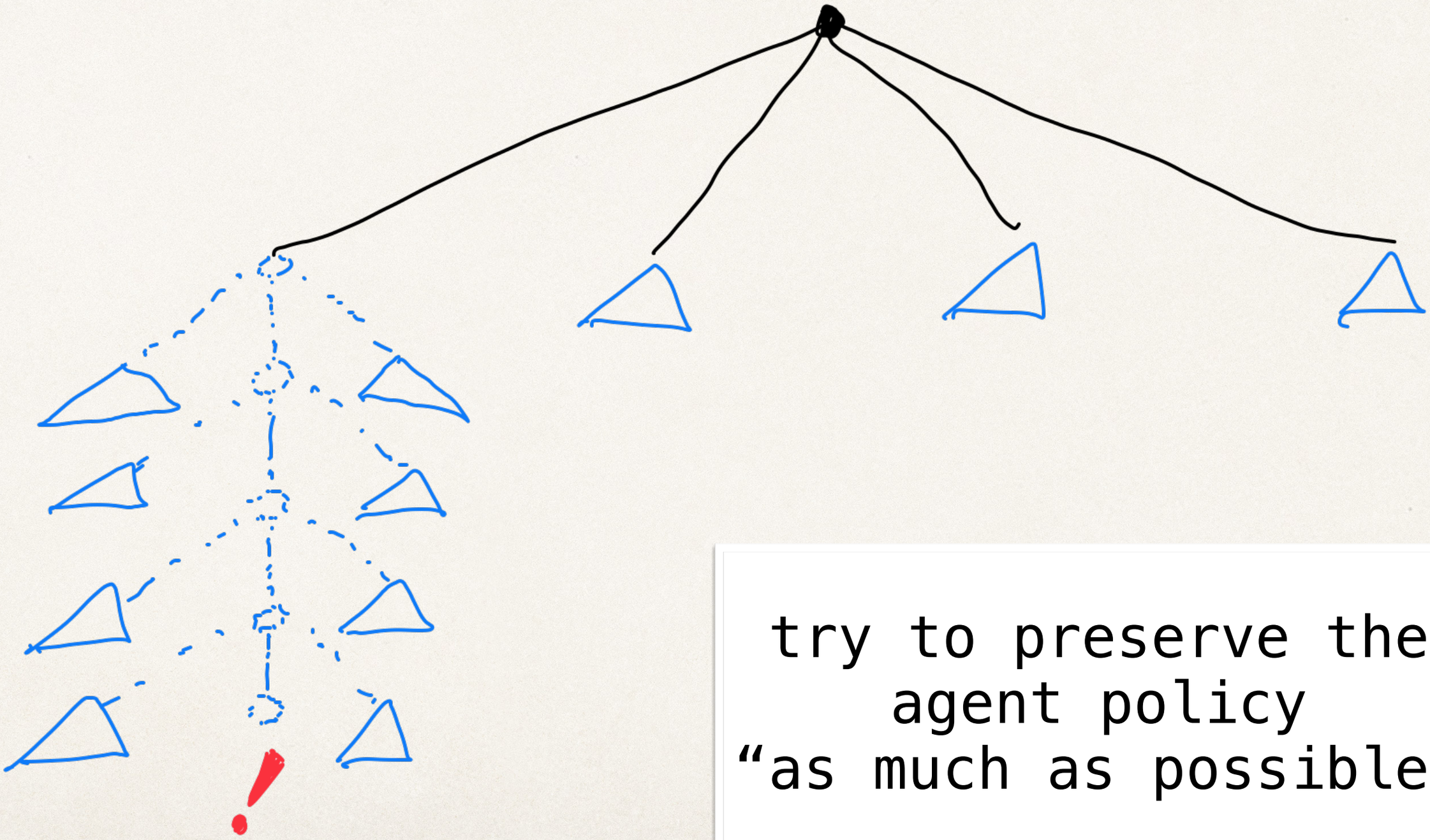
before: exploration

from here: exploitation

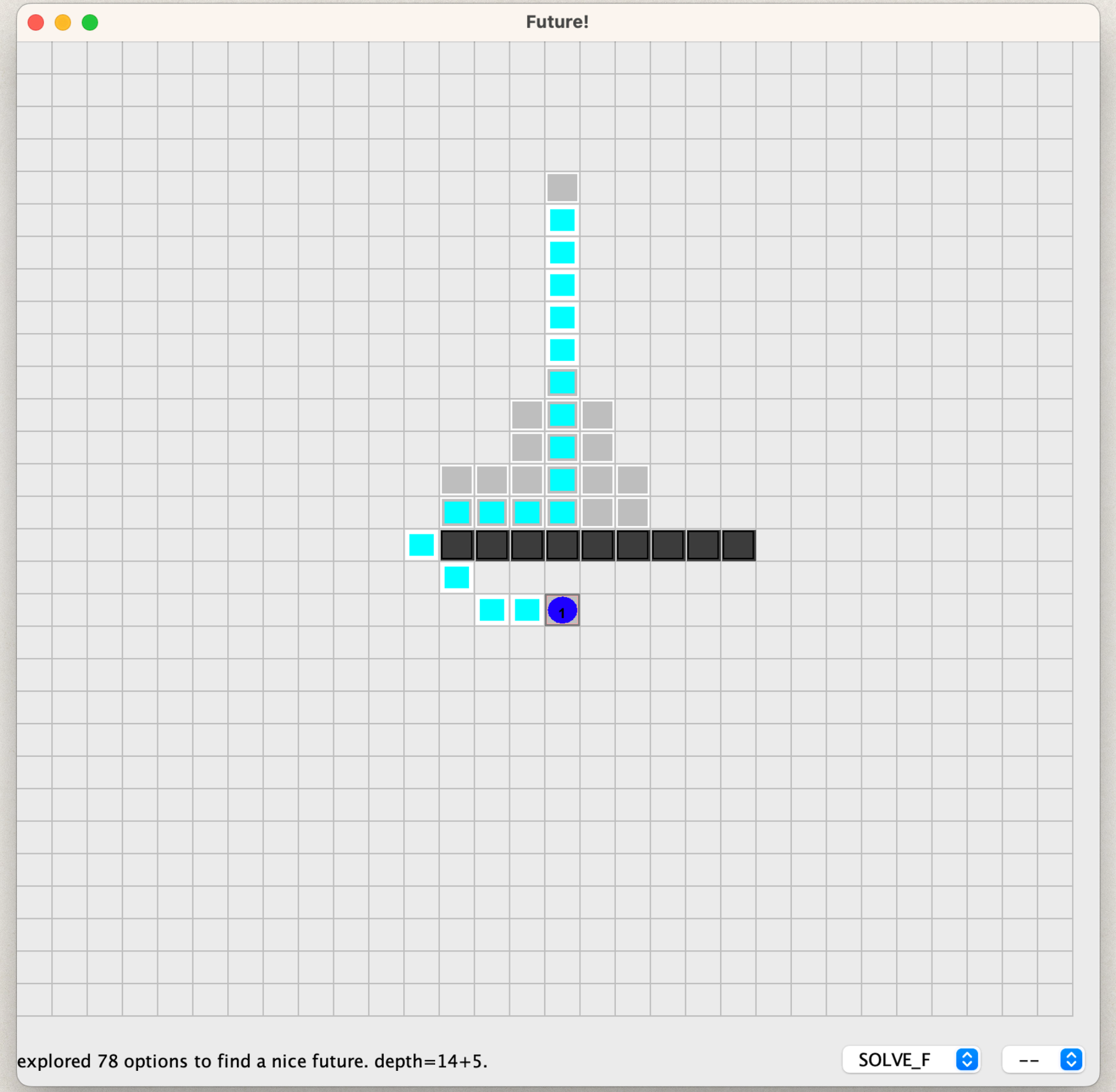
Strategy SOLVE_F

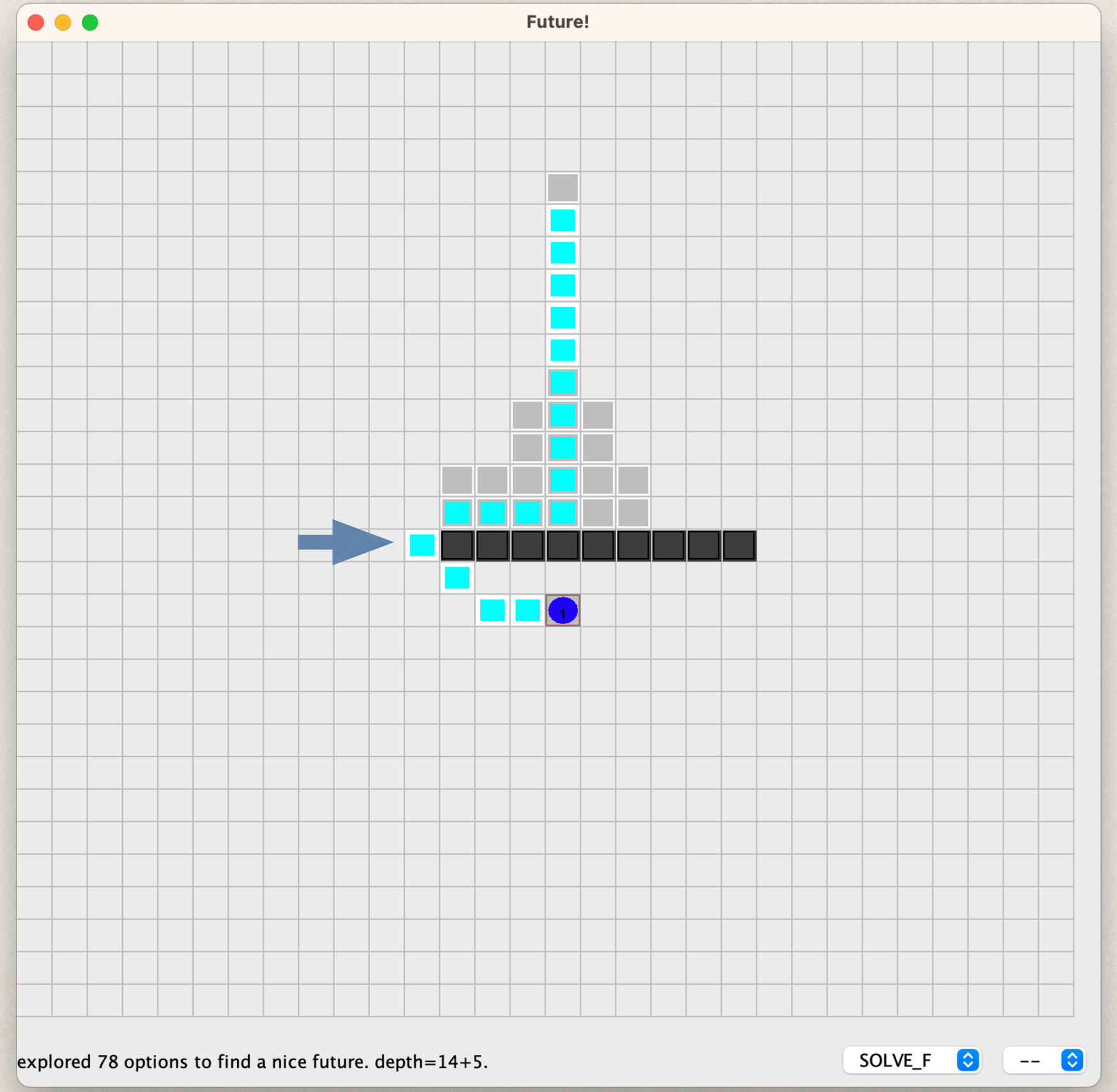


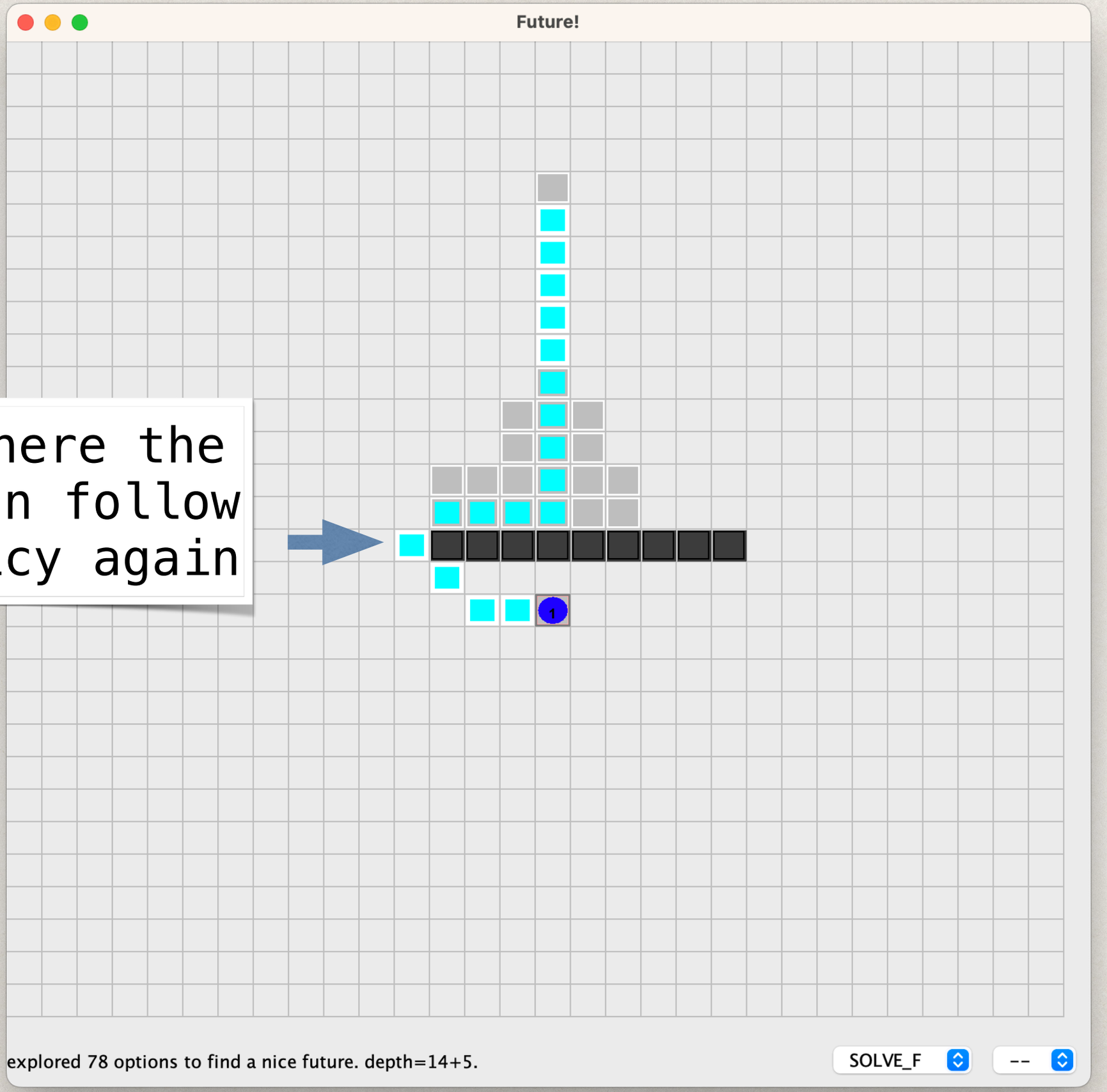
Strategy SOLVE_F



try to preserve the
agent policy
“as much as possible”

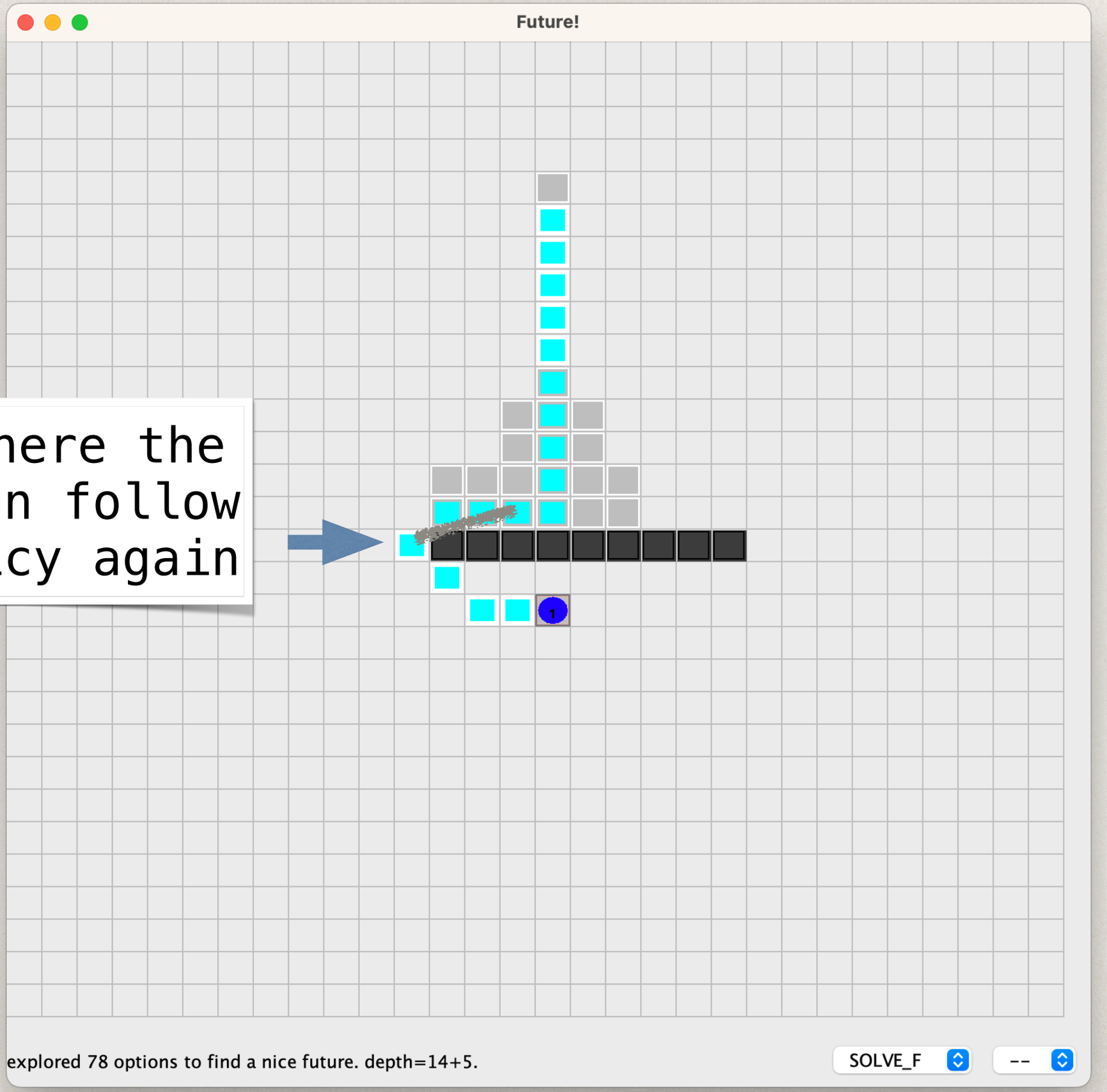

Future!
1
explored 78 options to find a nice future. depth=14+5.
SOLVE_F
--

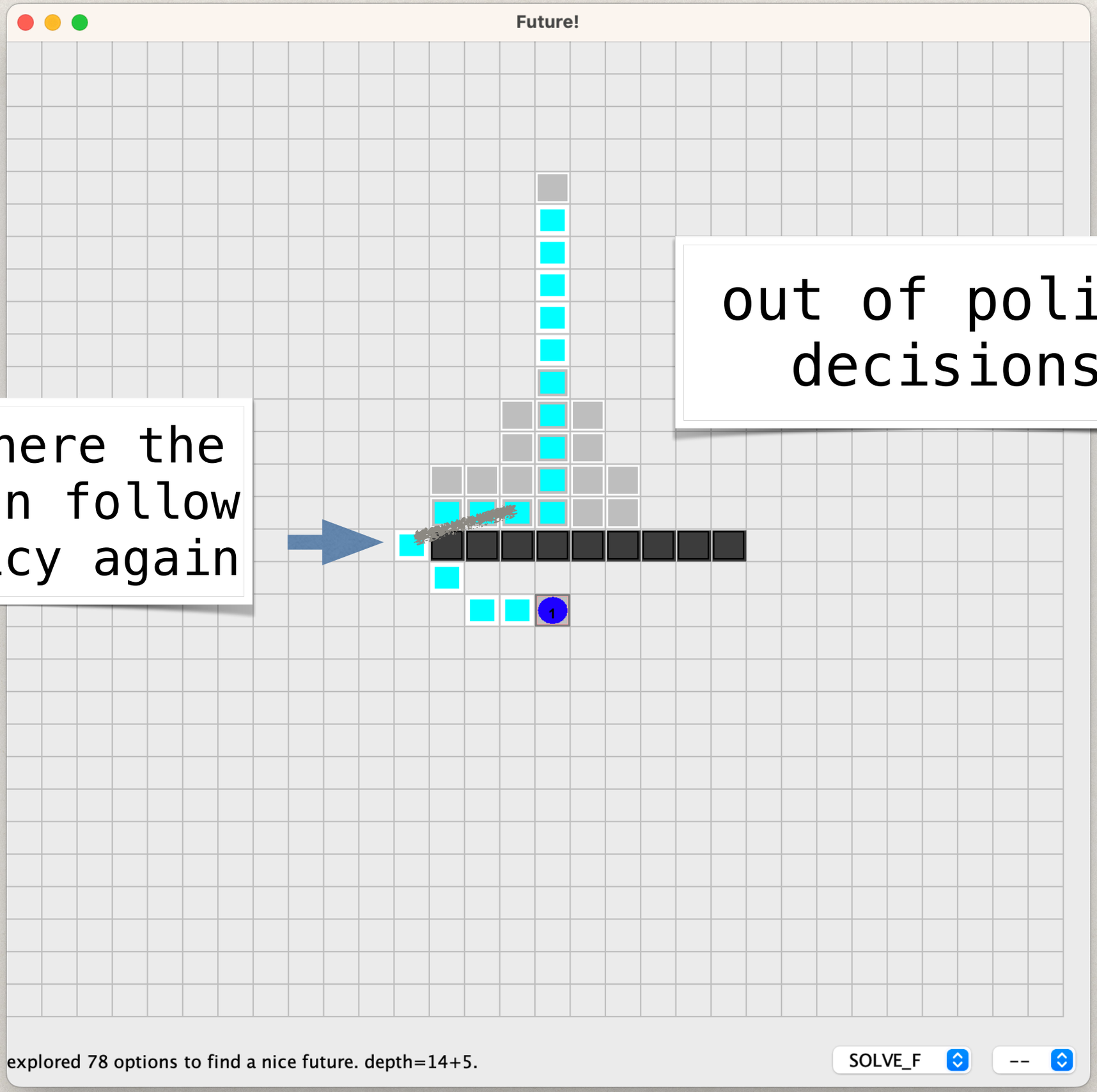

Future!
1
explored 78 options to find a nice future. depth=14+5.
SOLVE_F
--



point where the agent can follow its policy again

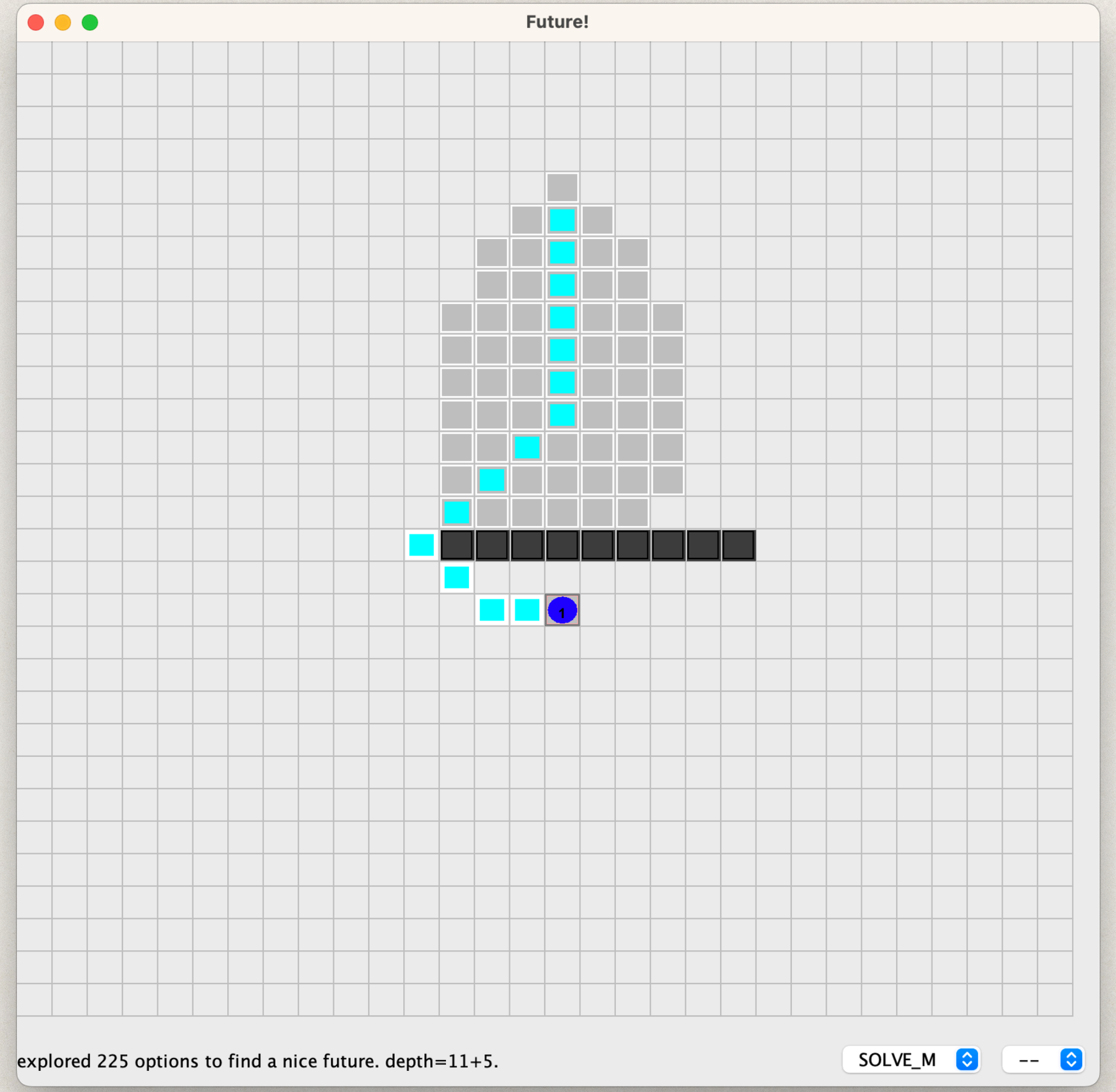
point where the agent can follow its policy again

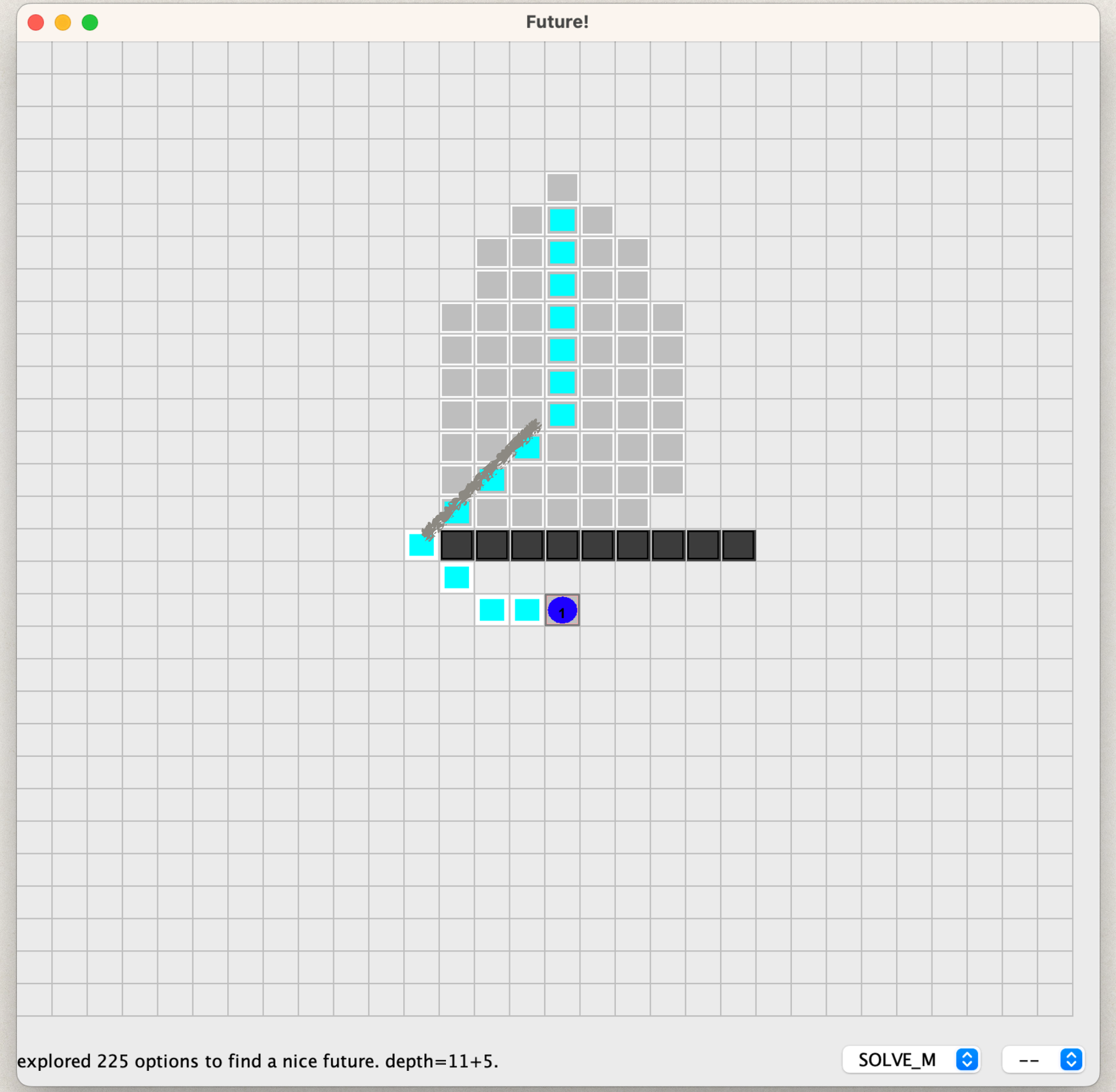


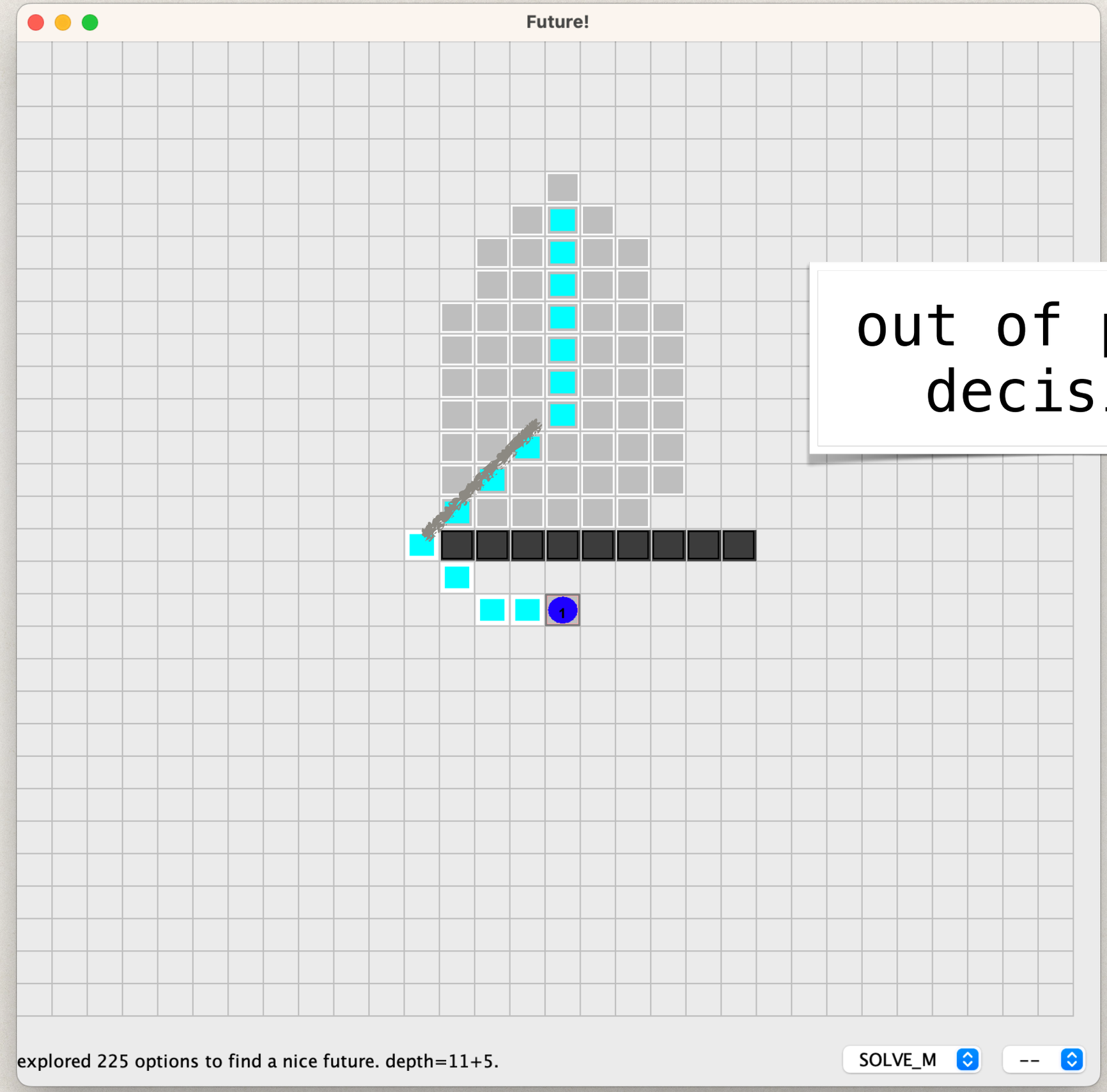


out of policy decisions

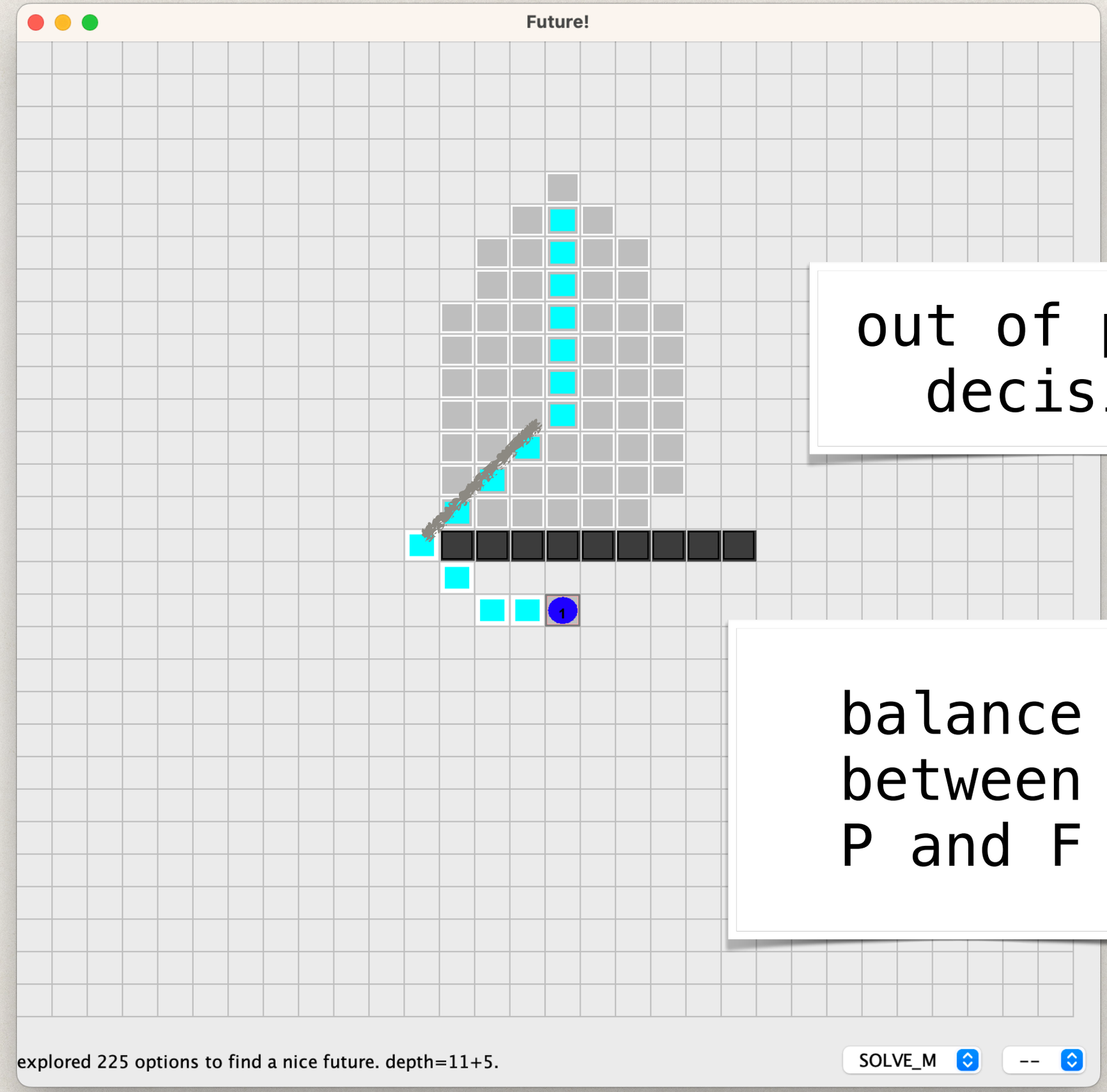
point where the agent can follow its policy again


Future!
1
explored 225 options to find a nice future. depth=11+5.
SOLVE_M
--


Future!
1
explored 225 options to find a nice future. depth=11+5.
SOLVE_M
--



out of policy decisions



out of policy decisions

balance between P and F

version 1.1

updated HERE

Scenario line

little "detour"

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

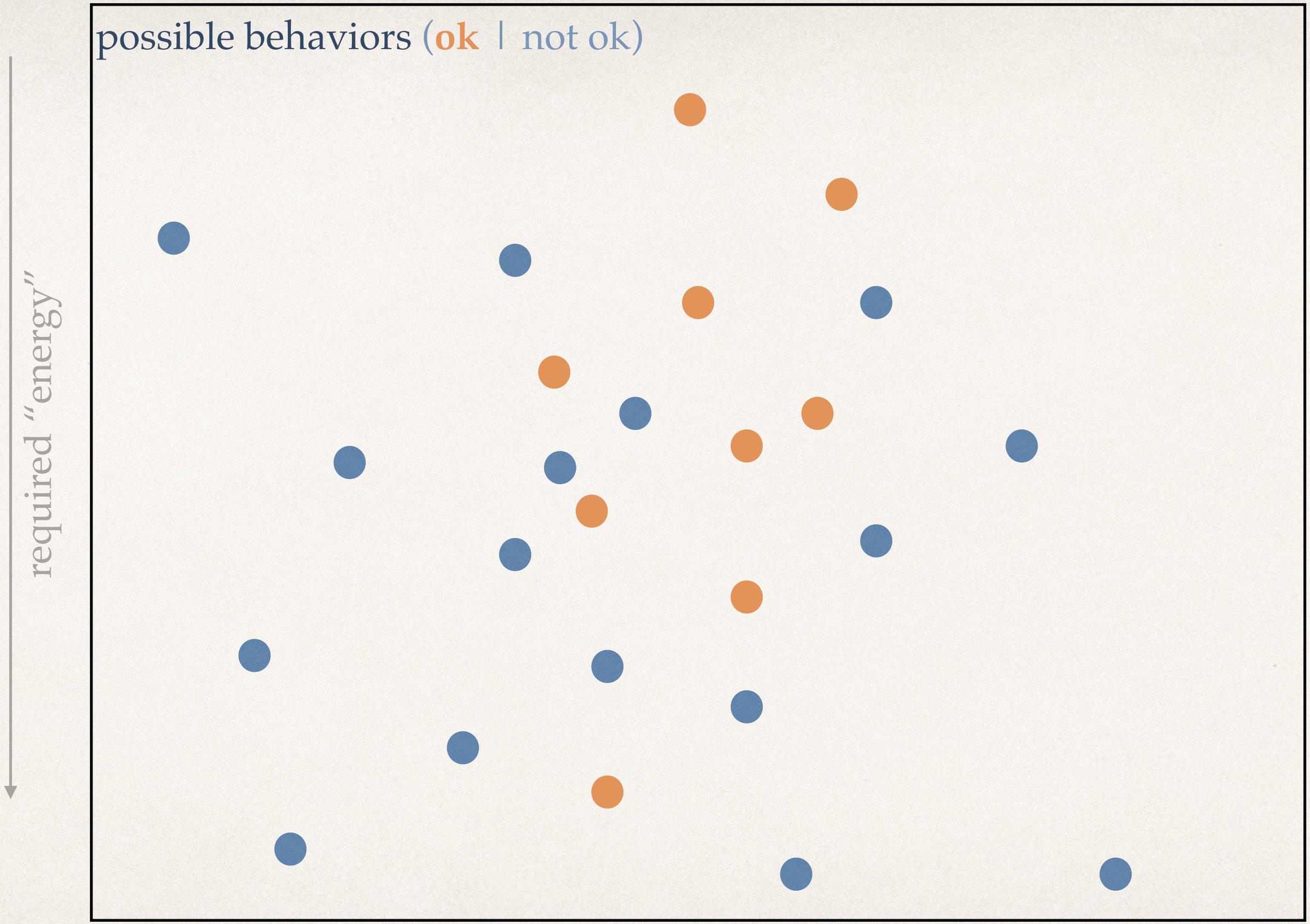
- move the agent out of its policy
  (while exploiting its policy as much as possible)

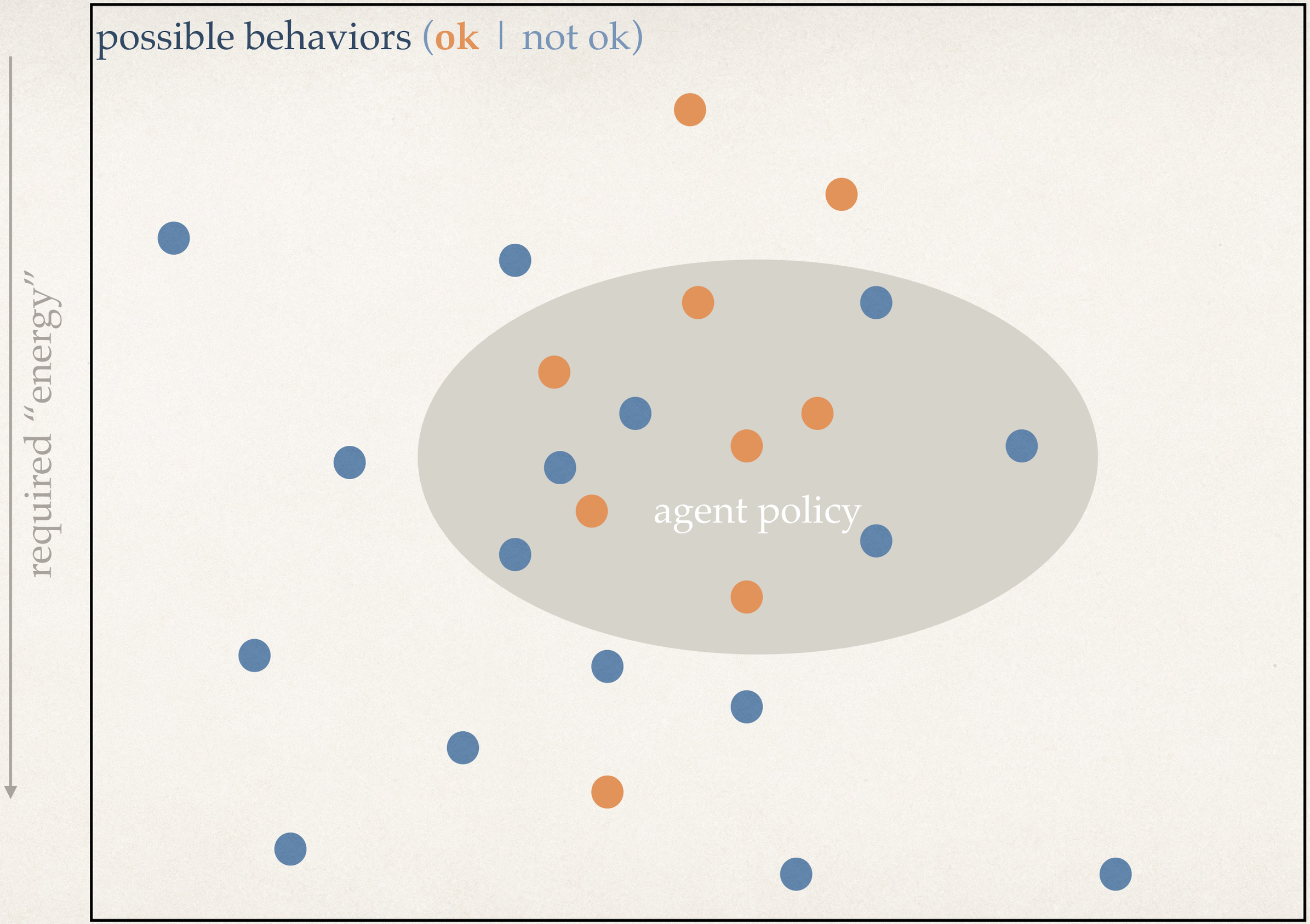
- SOLVE_P: change option as **soon** as possible
  + shorter plan
  - more exploration

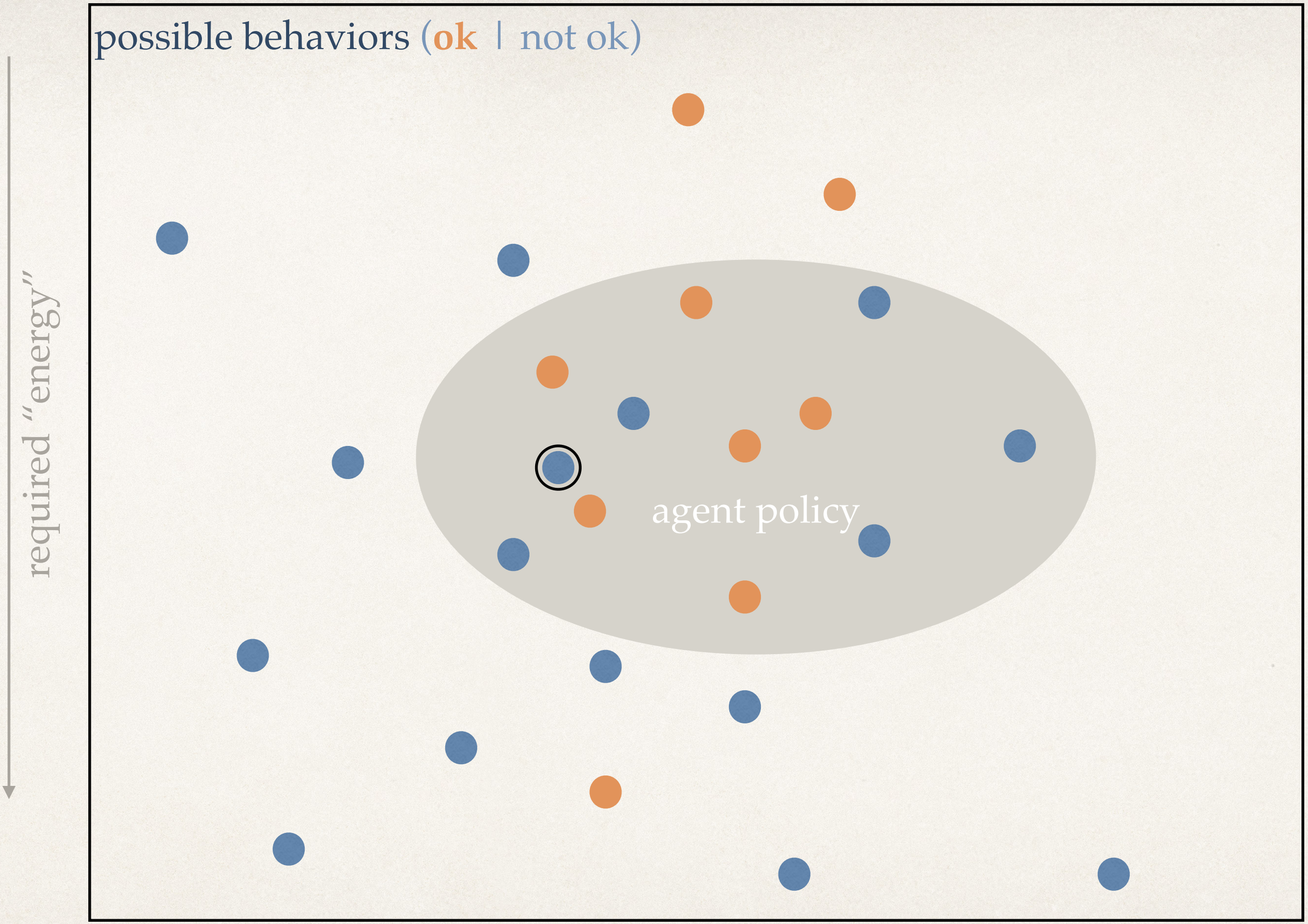
- SOLVE_F: change option as **latte** as possible
  + less exploration
  - not so good plan

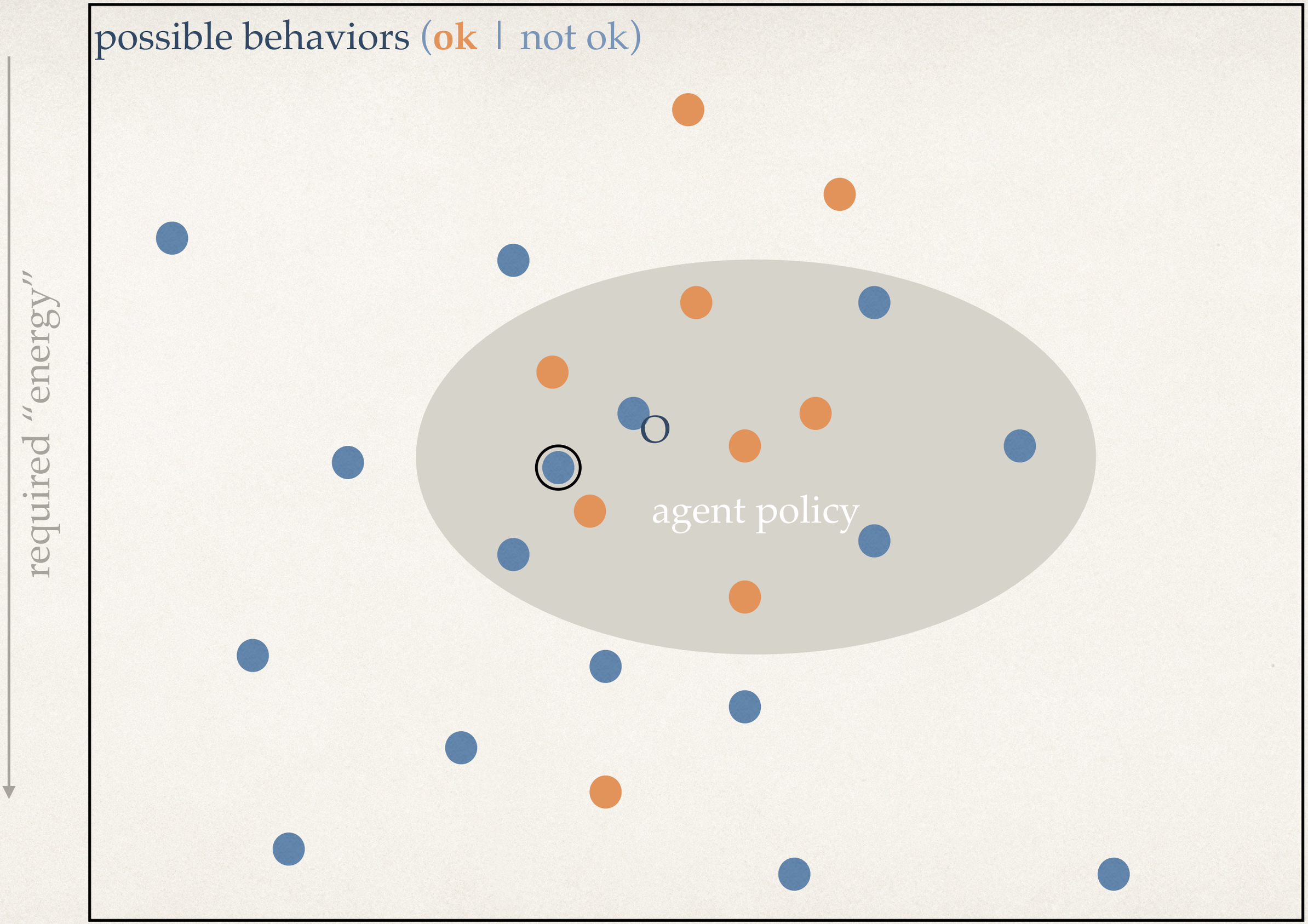
About the Agent Program

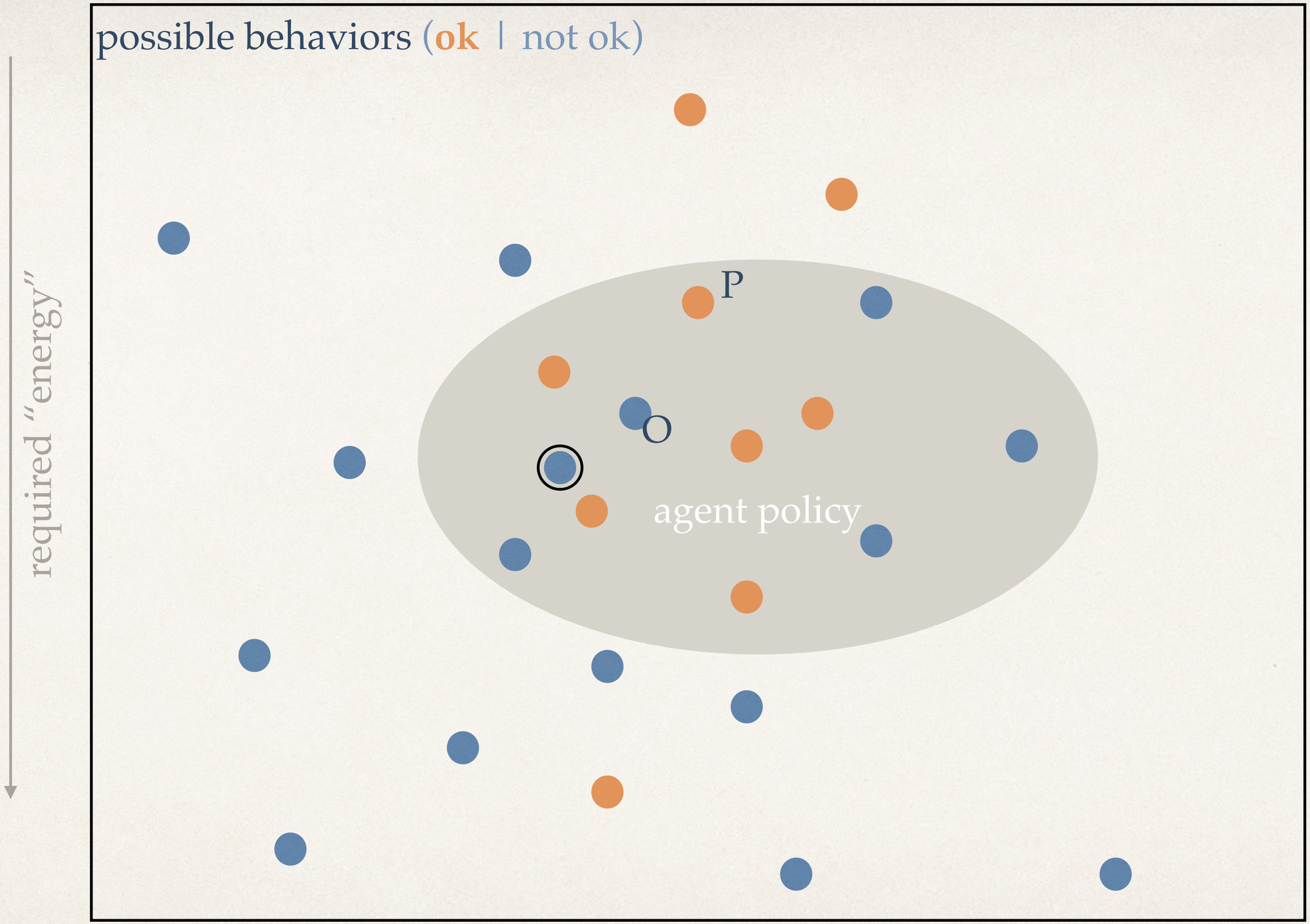
- the agent program (that defines its policy) is the (how-to) **knowledge** the programmer gives to the agent
- without it, we (may) have tradicional search (to consider this **K** distinguishes Jason(S) from search)
- if **K** is very very bad designed, we (may) have tradicional search [if, at least,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

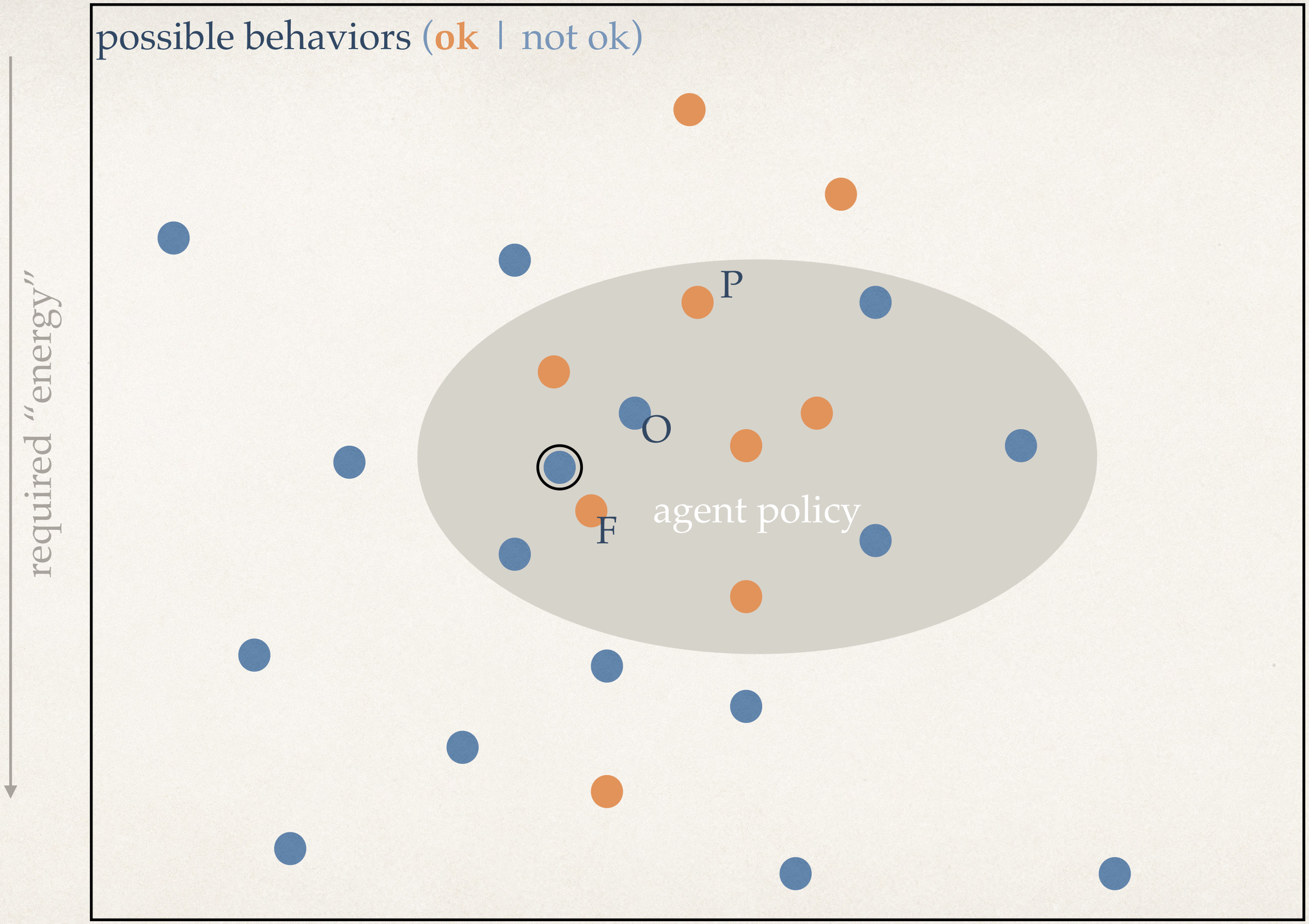

possible behaviors (ok | not ok)
required "energy"

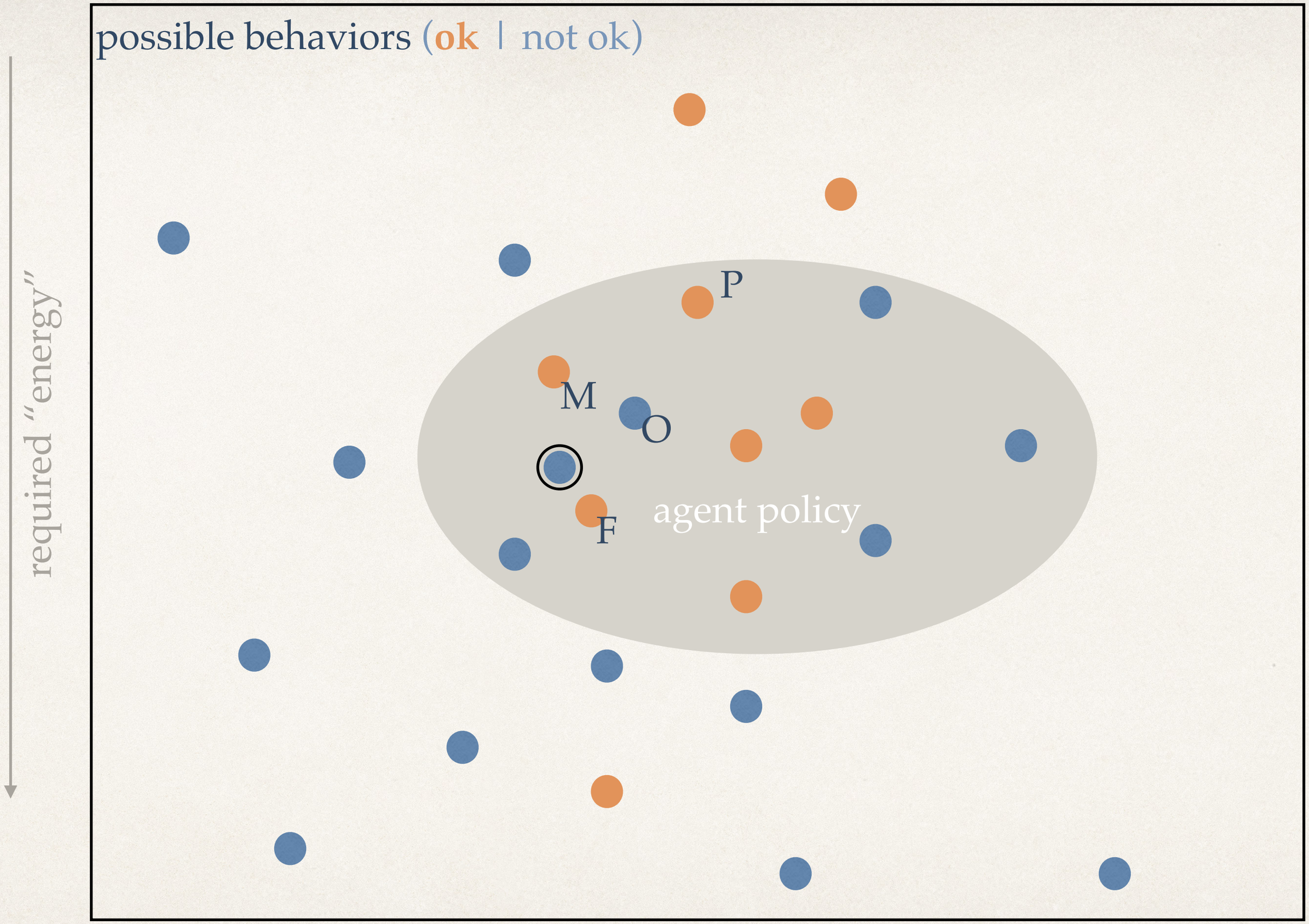

possible behaviors (ok | not ok)
agent policy
required "energy"

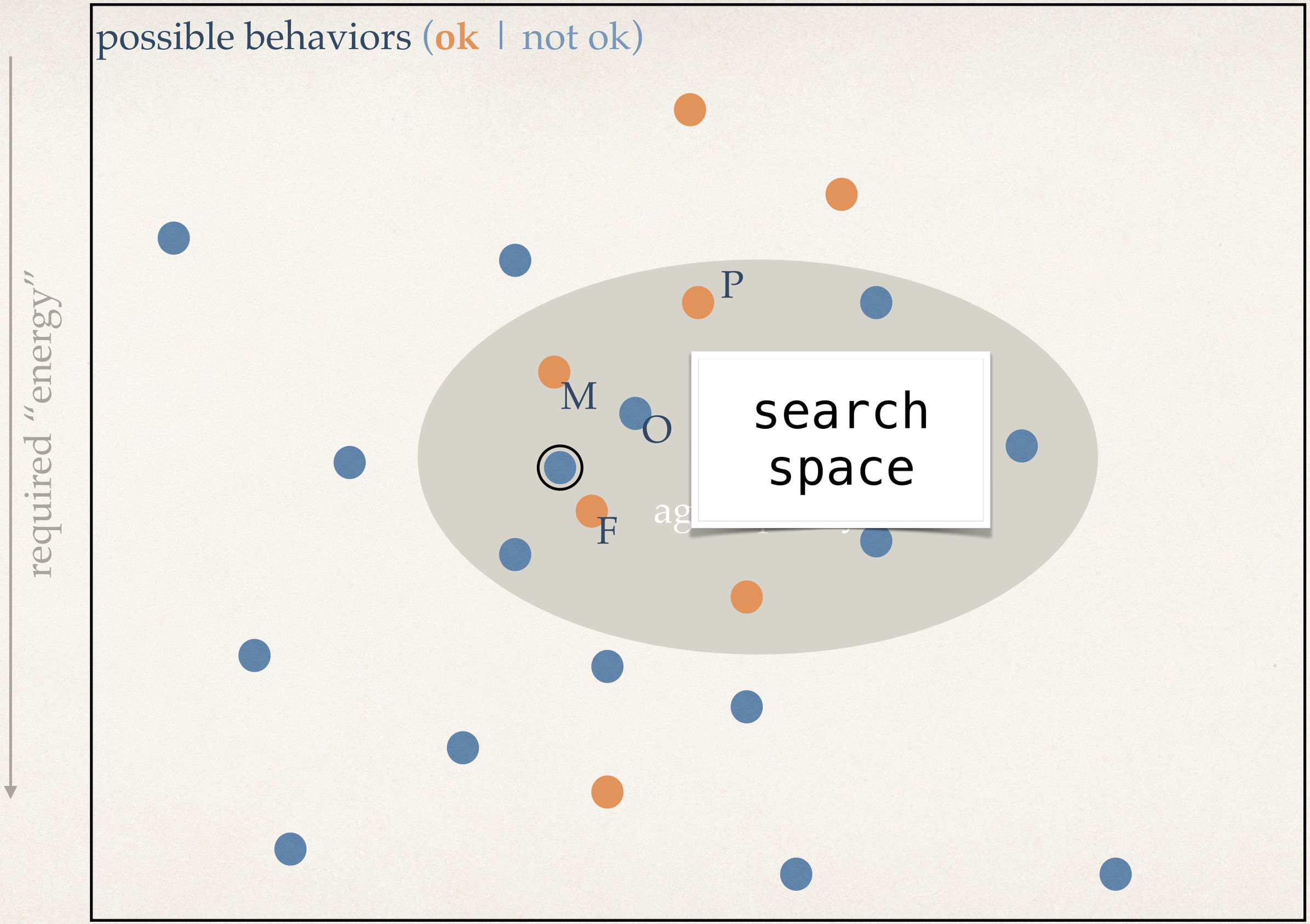

possible behaviors (ok | not ok)
agent policy
required "energy"


possible behaviors (ok | not ok)
agent policy
required "energy"


possible behaviors (ok | not ok)
P
O
agent policy
required "energy"


possible behaviors (ok | not ok)
P
O
F
agent policy
required "energy"


possible behaviors (ok | not ok)
P
M
O
F
agent policy
required "energy"


possible behaviors (ok | not ok)
P
M
O
F
search space
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

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

environment transitions

$$e : S \times A^n \to S$$

agent i policy (as ordered options)

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

future state at k

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

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

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

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

envi

e

agen

options)

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

future state at k

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

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

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

fully observable

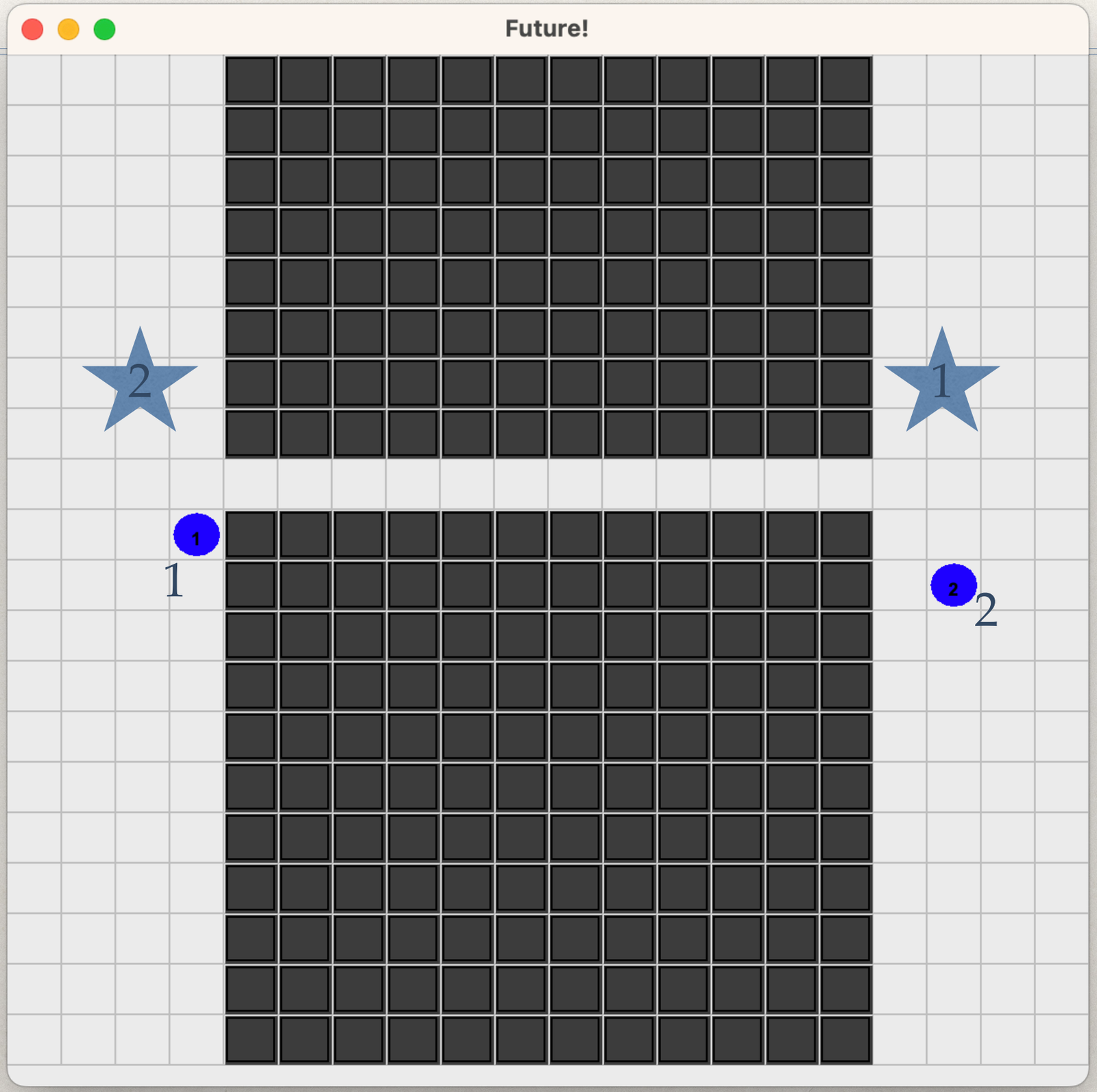
deterministic

multi-agent

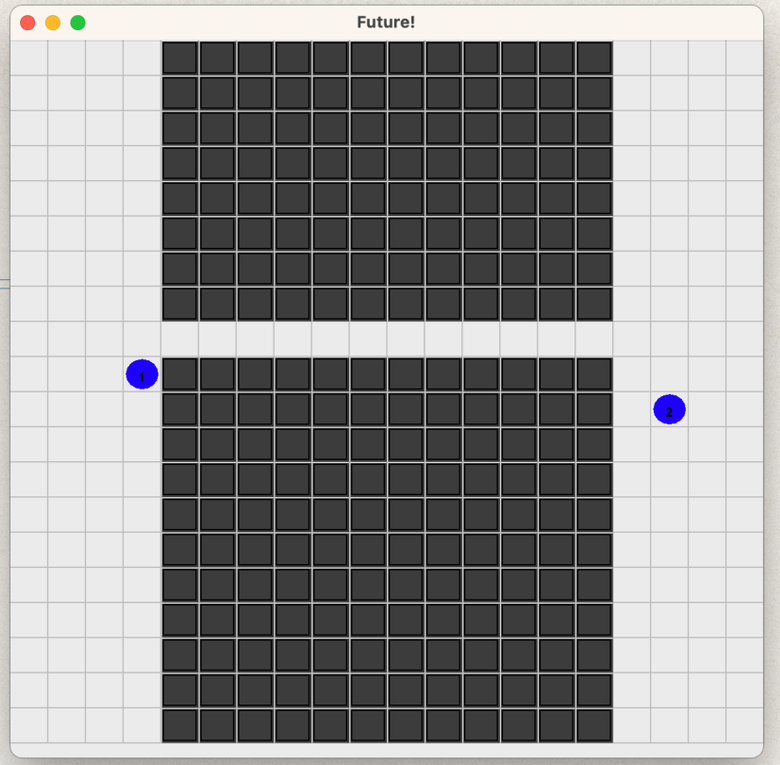
static

discret

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	∞	∞	∞ (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 placed (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

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

$e(s, a_1, a_2, \ldots, s')$ is the probability of s′ from s doing a

agent i policy (as ordered options)

$\pi_i : S \rightarrow A^n$

state and certainty in the future

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

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

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

fully observable

stochastic
dynamic
environment

deterministic
agent

multi-agent

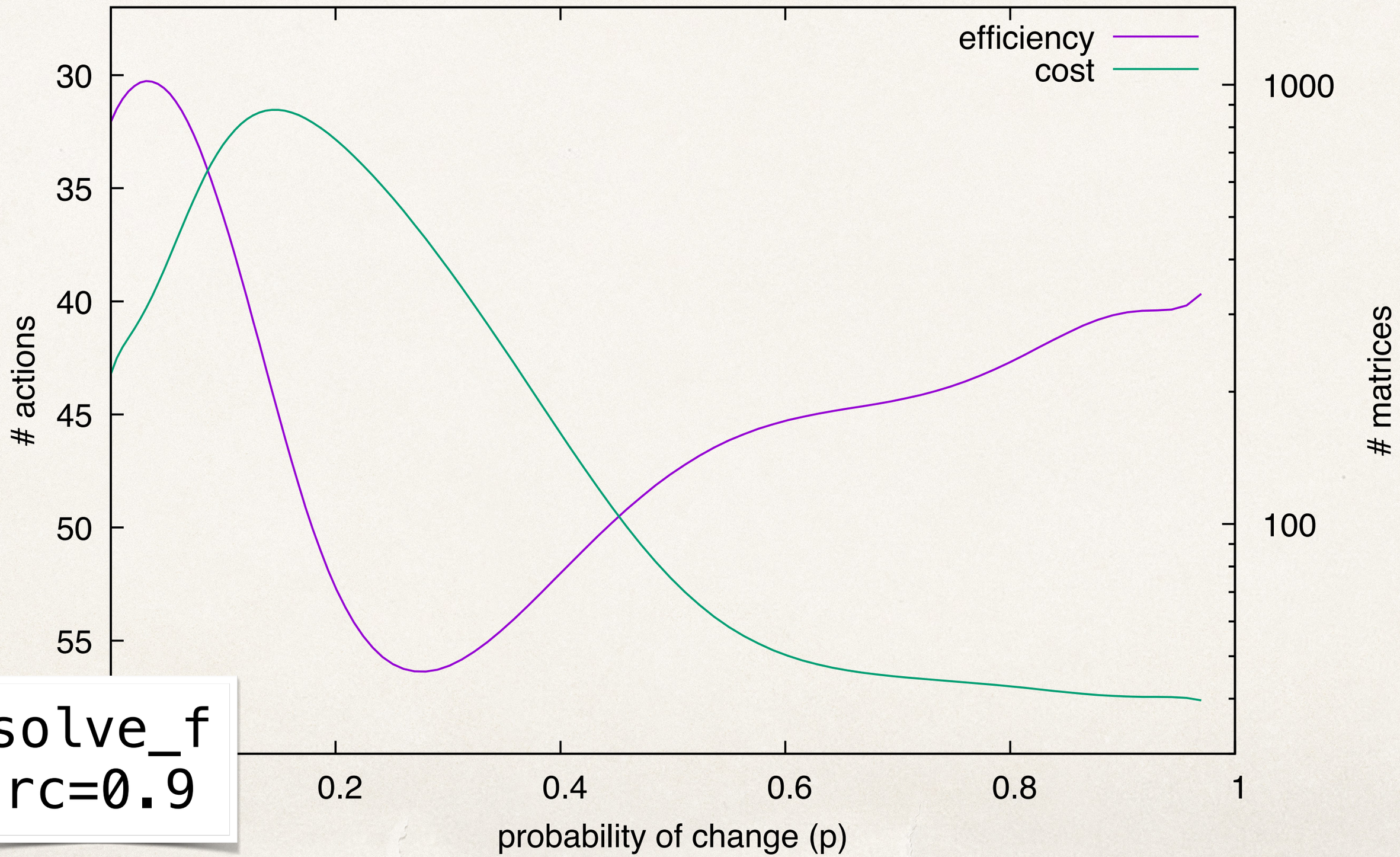
discret

Evaluation

- Measure:
  - efficiency (steps to achieve a goal),
  - (computational) cost,
  - steps in policy, …
- Varies
  - uncertainty of the environment
  - how far to look ahead
  - strategies (solve p, m, f)

Preliminary results

updated
HERE




```
solve_f
rc=0.9
```

Next Steps

- Relax assumptions (deterministic environment, …)
- More scenarios (more complex and realistic)
- Running in background
- Opportunistic options
  oracle: “your policy is ok, but you could do better”
- Consider backtrack (options in the past)
- Other types of problem (internal state, norm violation)
- Shared matrix
- Related work

What to Measure?

- time & memory
- reactiveness
- efficiency & efficacy