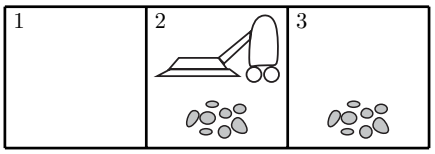
02180 Intro to AI
Exercises for week 4, 26/2-19
SOLUTIONS

Exercise 1

To begin with, consider the deterministic and fully observable vacuum cleaner world with three squares, and the initial state being:



The goal states are the states where all squares are clean.

a. Draw a graph of the state space up to and including distance 2 from the initial state. Don't forget states that *lead* to other reachable states, and not only those that are reachable.

SOLUTION. See Figure 1.

b. How many states are there up to distance 2 from the initial state? How many states are there in total for the problem with 3 squares? How many are goal states? How many of the states are reachable from the initial state?

If there were n squares (horizontally) instead of 3, and there could be dirt in any and all of them in the initial states, how many states are there then in the state space, and how many of those are goal states?

SOLUTION. There are 8 states up to distance 2 from the initial state, found by counting in the drawn state space.

We can represent each state as a quadruple (s_1, s_2, s_3, r) where $s_1, s_2, s_3 \in \{\mathtt{c}, \mathtt{d}\}$ for square one to three being clean or dirty respectively, and $r \in \{1, 2, 3\}$ for the robot being in squares one to three respectively. This gives a total of $2 \cdot 2 \cdot 2 \cdot 3 = 24$ possible states, since no combinations represent invalid states. 3 of these states are goal states, namely $(\mathtt{c}, \mathtt{c}, \mathtt{c}, 1)$, $(\mathtt{c}, \mathtt{c}, \mathtt{c}, 2)$, and $(\mathtt{c}, \mathtt{c}, \mathtt{c}, 3)$. From the initial state, we can never reach a state which has dirt in square 1, but we can reach every other state. This means we can reach all states represented by tuples of the form $(\mathtt{c}, s2, s3, r)$ of which there are $2 \cdot 2 \cdot 3 = 12$.

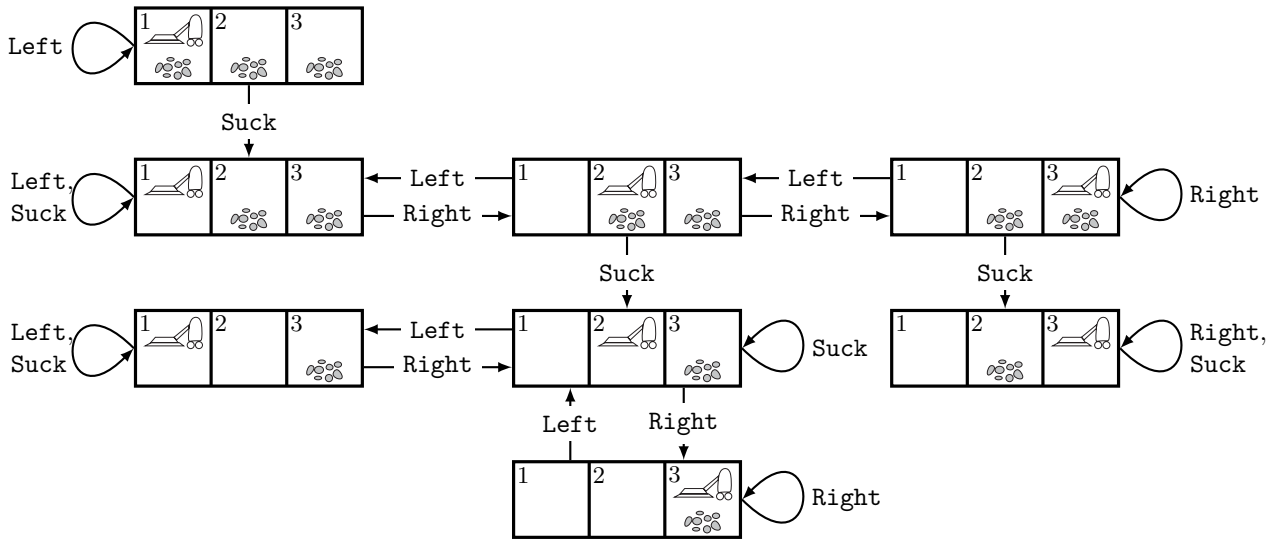


Figure 1: Solution for Exercise 1a.

If there are n squares, then each square still has two possible values in a state, and the agent can be in n squares. A tuple representation of such states could be $(s_1, s_2, \ldots, s_n, r)$ with values like before, for a total of $2^n \cdot n$ states, and of these only n are goal states.

Exercise 2

Now we move on to consider the *erratic* vacuum cleaner world with 3 squares, where the `Suck` action cleans the square of the robot, but also possibly any of the adjacent squares (e.g. when in square 2, performing a `Suck` action also possibly cleans square 1 and/or 3).

a. Using the tuple representation of states from the lecture, e.g. the initial state is $(c, d, d, 2)$, determine:

   1. RESULTS($(d, c, d, 1)$, `Suck`)
   2. RESULTS($(d, d, d, 1)$, `Suck`)
   3. RESULTS($(d, d, d, 2)$, `Suck`)
   3. RESULTS($(d, c, d, 2)$, `Suck`)
   3. RESULTS($(c, d, d, 3)$, `Suck`)

SOLUTION.

1. RESULTS($(d, c, d, 1)$, `Suck`) $= \{(c, c, d, 1)\}$.
2. RESULTS($(d, d, d, 1)$, `Suck`) $= \{(c, d, d, 1), (c, c, d, 1)\}$.

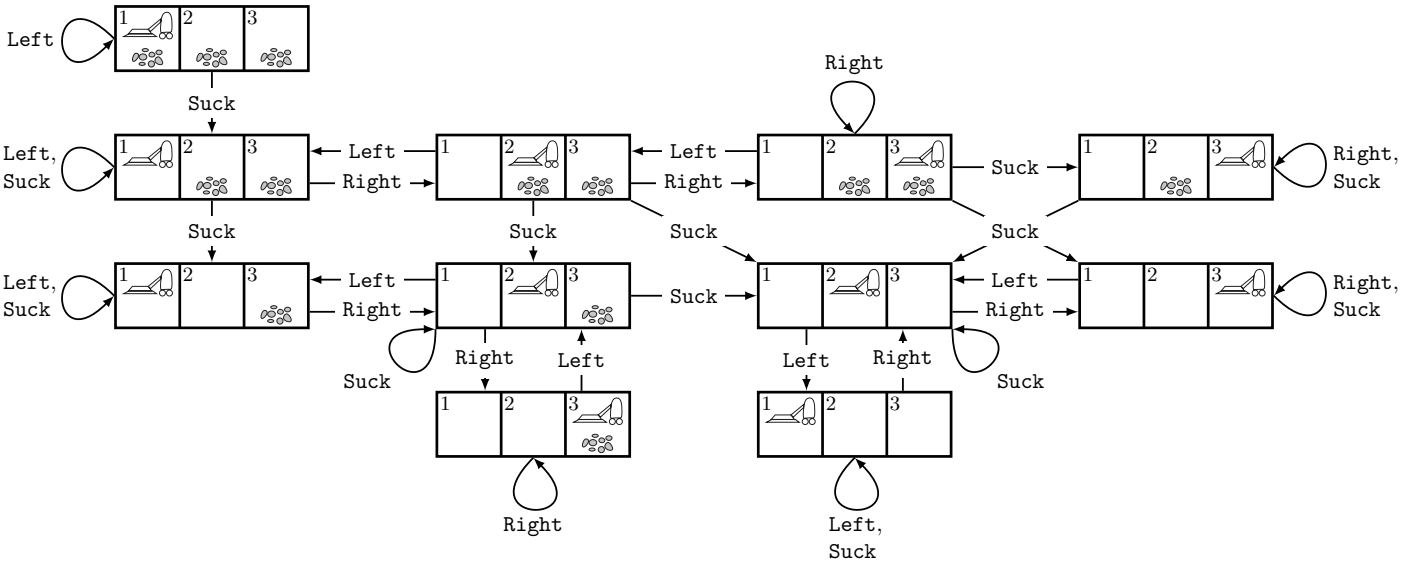


Figure 2: Solution for Exercise 2b.

3. RESULTS$((d, d, d, 2),$ `Suck`$) = \{(d, c, d, 2), (c, c, d, 2), (d, c, c, 2), (c, c, c, 2)\}$.
3. RESULTS$((d, c, d, 2),$ `Suck`$) = \{(d, c, d, 2), (c, c, d, 2), (d, c, c, 2), (c, c, c, 2)\}$.
3. RESULTS$((c, d, d, 3),$ `Suck`$) = \{(c, d, c, 3), (c, c, c, 3)\}$.

b. Draw a graph of the state space up to and including distance 2 from the same initial state as the previous question. Each state can now have several outgoing edges with the same action label leading to different states.

SOLUTION. See Figure 2.

c. Draw the AND-OR tree up to depth 2 (counting each and-or transition as 1 depth) with the given initial state as root. Mark the OR-nodes as:

- *GOAL*, if they're a goal state;
- *LOOP*, if they occur on the path to the root; and
- *OPEN*, if they're at depth 2 and are neither a goal or a loop state.

SOLUTION. See Figure 3, disregarding the red nodes and edges (at depth 3).

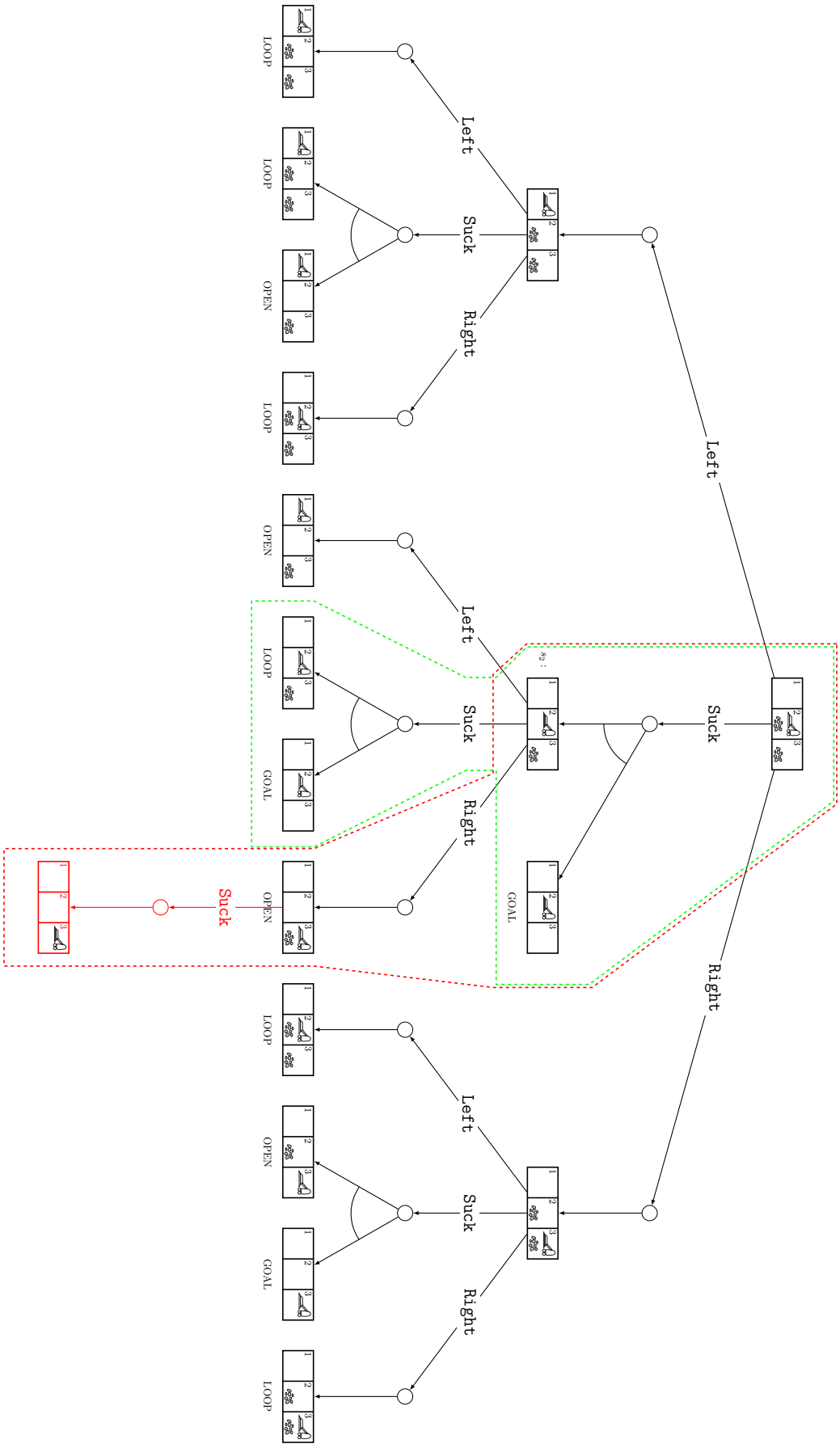


Figure 3: Solution for Exercise 2c and 2e.

d. What is the maximal outgoing degree of any AND-node in the tree? Is there a state from which the AND-node of some action would have a higher out-degree, and if so then what state, action, and degree would that be?

SOLUTION. Every AND-node from a `Suck` action in the tree has outgoing degree 2, which is maximal for this tree.

e. Is there a solution in the AND-OR tree? Highlight the solution if it exists in the tree, and otherwise expand more of the tree so that it contains a solution and then highlight that. You can expand in any order you prefer. Write the conditional plan in the language from the slides; use parentheses to resolve ambiguous syntax.

SOLUTION. There is no solution.

It is, however, possible to define so-called *cyclic solutions* that allow loops, but such solutions are only guaranteed to be succesful under some further conditions. First of all, a *fairness assumption* has to be made: If a non-deterministic action has several possible outcomes and is executed an infinite number of times, then eventually every possible outcome will be realised. In this exercise, it would imply that if executing `Suck` an infinite number of times in s_2, eventually the dirt in square 3 will disappear. Under these assumptions, even cyclic plans can be succesful, if the fairness assumption implies that all cycles in the solution will eventually be broken. A cyclic solution is highlighted in the green dashed line, and would have the conditional plan (using the label syntax of R&N):

$$[\texttt{Suck},\ \textbf{if}\ \mathit{state} = s_2\ \textbf{then}\ (L : [\texttt{Suck},\ \textbf{if}\ \mathit{state} = s_2\ \textbf{then}\ L\ \textbf{else}\ \epsilon])\ \textbf{else}\ \epsilon]$$

If we disregard cyclic plans, then we could expand the tree with the red nodes and edges, to get the solution contained in the red dashed line. The solution has the conditional plan:

$$[\texttt{Suck},\ \textbf{if}\ \mathit{state} = s_2\ \textbf{then}\ [\texttt{Right}, \texttt{Suck}]\ \textbf{else}\ \epsilon]$$

f. Does the AND-OR tree correspond to any run of the AND-OR-GRAPH-SEARCH algorithm? If it doesn't, then draw a new AND-OR tree which could result from running the algorithm.

SOLUTION. No. The AND-OR-GRAPH-SEARCH algorithm does a depth-first search in the state space, and thus must exhaust all options in any previous branches before exploring down new branches of the AND-OR search tree. In our AND-OR tree, we have explored a bit down every branch from the root, but not entirely exhausted any, which is not possible in a depth-first search. What we have done could result from a breadth-first exploration of the state space, but never a depth-first exploration like the considered search algorithm.

An AND-OR tree that might result from a run of the algorithm is shown in Figure 4.

Exercise 3

Now consider the deterministic and partially observable *local-sensing* vacuum cleaner world, where the agent knows which square it is in, and senses whether that square is either clean or dirty after each action. The `Suck` action again only cleans the square that the agent is in. Consider the following initial state:

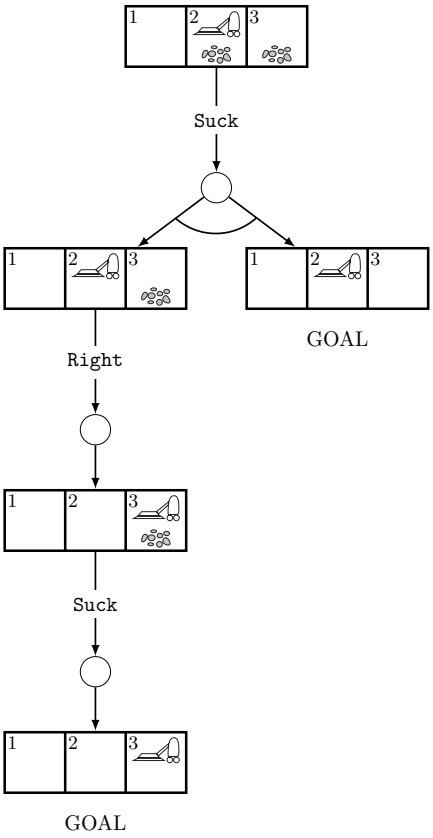
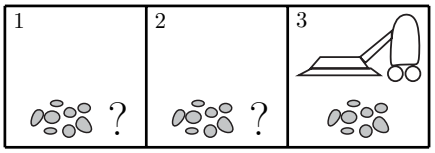


Figure 4: Solution for Exercise 2f.



where the agent knows that there is dirt in square 3, but not whether square 1 or 2 are dirty or clean.

a. What is the initial belief state? You can use the tuple representation for each physical state. How many states are there in the belief state space? If there were n squares, then how many belief states are there?

SOLUTION. The initial belief state is $\{(c, c, d, 3), (c, d, d, 3), (d, c, d, 3), (d, d, d, 3)\}$, encoding that the agent is in square 3 and that square 3 is dirty (consistent between all the physical states in the belief state), but that the remaining two squares can be in any configuration of clean and/or dirty (i.e. their state is unknown).

Given that there are 24 physical states, and that a belief state is any set of physical states, we get that there are $2^{24} = 16,777,216$ possible belief states.

When there are n squares, we found that there were $2^n \cdot n$ physical states, which in turn means there are $2^{2^n \cdot n}$ possible belief states, a double-exponential number of belief states in terms of n. For $n = 10$, we have $2^{2^{10} \cdot 10} = 2^{1024 \cdot 10} = 2^{10240} = 10^{\log 2 \cdot 10240} \approx 10^{3083}$.

b. We will use the notation $X@Y$ where X is `clean` or `dirty` and Y is 1, 2, or 3 to denote percepts for the squares being clean or dirty. Hence, `dirty`@2 is the percept that square 2 is dirty. Now define the following physical states:

s_0 : $(c, d, c, 3)$

s_1 : $(c, d, c, 2)$

s_2 : $(c, d, d, 3)$

s_3 : $(c, c, c, 3)$

s_4 : $(c, c, d, 3)$

s_5 : $(d, c, d, 3)$

s_6 : $(d, d, d, 3)$

s_7 : $(d, c, c, 3)$

Using the introduced percept notation, determine the following:

- PERCEPT(s_0)
- PERCEPT(s_1)
- PERCEPT(s_7)
- POSSIBLE-PERCEPTS($\{s_0, s_2, s_3, s_4\}$)
- UPDATE($\{s_0, s_2, s_3, s_4\}$, `dirty`@3)
- RESULTS($\{s_2, s_4, s_5, s_6\}$, `Left`)

SOLUTION.

- PERCEPT(s_0) = `clean@3`.
- PERCEPT(s_1) = `dirty@2`.
- PERCEPT(s_7) = `clean@3`.
- POSSIBLE-PERCEPTS($\{s_0, s_2, s_3, s_4\}$) = {`clean@3`, `dirty@3`}.
- UPDATE($\{s_0, s_2, s_3, s_4\}$, `dirty`@3) = $\{s_2, s_4\}$.
- RESULTS($\{s_2, s_4, s_5, s_6\}$, `Left`) = $\{\{(c, c, d, 2), (d, c, d, 2)\}, \{(c, d, d, 2), (d, d, d, 2)\}\}$.

d. Assume the modelling change so that the robot has a vague sensor which can only determine which of the following two is the case: 1) there is dirt in the current square or an adjacent square; 2) the current square and all adjacent squares are clean. We can model this new situation with only two percepts, `dirty` and `clean`. Explain why. Then determine the same function values as in b, except the percept `dirty@3` is replaced by `dirty`.

SOLUTION. The percept `dirty` covers case 1, where the robot senses dirt in the current or an adjacent square. The percept `clean` covers case 2, where no dirt is sensed (in the current location or any adjacent square).

- PERCEPT(s_0) = `dirty`.
- PERCEPT(s_1) = `dirty`.
- PERCEPT(s_7) = `clean`.
- POSSIBLE-PERCEPTS($\{s_0, s_2, s_3, s_4\}$) = {`clean`, `dirty`}.
- UPDATE($\{s_0, s_2, s_3, s_4\}$, `dirty`) = $\{s_0, s_2, s_4\}$.
- RESULTS($\{s_2, s_4, s_5, s_6\}$, `Left`) = $\{\{(c, c, d, 2), (d, c, d, 2), (c, d, d, 2), (d, d, d, 2)\}\}$.