

# Einführung in Artificial Intelligence SS 2024, 4.0 VU, 192.027

## Exercise Sheet 5 – Planning and Decision Theory

For the discussion part of this exercise, mark and upload your solved exercises in **TUWEL** until Wednesday, June 5, 23:55 CEST. The registration for a solution discussion ends on Friday, June 7, 23:55 CEST. Be sure that you tick only those exercises that you can solve and explain!

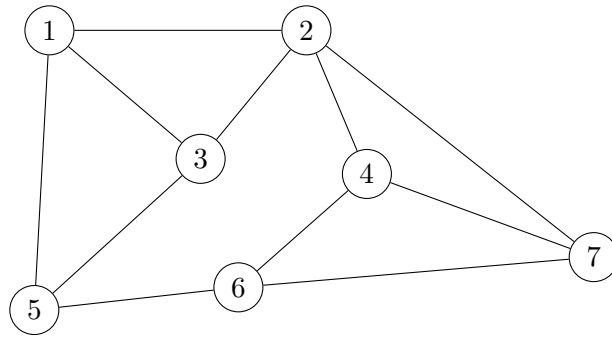
In the discussion, you will be asked questions about your solutions of examples you checked. The discussion will be evaluated with 0–25 points, which are weighted with the fraction of checked examples and rounded to the next integer. There is *no minimum number of points* needed for a positive grade (i.e., you do not need to participate for a positive grade, but you can get at most 85% without doing exercises).

Note, however, that *your registration is binding*. Thus, if you register for a solution discussion, then it is *mandatory* to show up. Not coming to the discussion after registration will lead to a reduction of examination attempts from 4 to 2.

Please ask questions in the **TUWEL** forum or visit our tutors during the tutor hours (see **TUWEL**).

---

**Exercise 5.1:** Consider the 3-colorability problem for the following graph:

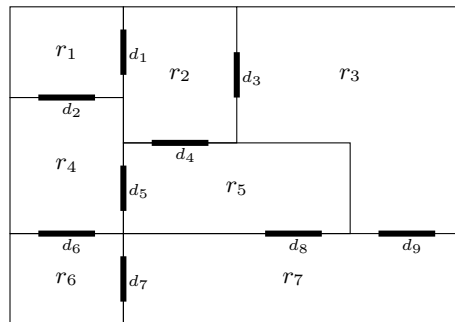


- Perform one step of the backtracking search on the following graph. Use the colors red, green, and blue. Select the first node by using the *degree heuristic*. After that, select the nodes according to the *minimum-remaining-values heuristic* and the *degree heuristic* if the remaining values of two or more nodes are equal. If you still have multiple options after applying both heuristics, select the node with the smallest number. If the step terminates and colors the entire graph, show the coloring it found. If it necessitates a backtrack, state which node could not be colored and stop (i.e., do not backtrack).
- Find a partial coloring of the graph and select one additional node to color so that the *least-constraining-value heuristic* forces you to choose one particular color for that node.

**Exercise 5.2:** Agent 99 is a time-travelling secret agent whose task is to retrieve, in the middle of a night in 1929, a mobile phone at Kurt Gödel's apartment that her colleague from CONTROL had left behind while taking a selfie. The mission is of vital importance, as if she fails, the fabric of space-time will be profoundly damaged. The colleague left her a detailed plan of the house, specifying which doors he has left unlocked, and told her where the phone is located. Our agent must navigate through the house, possibly unlocking any locked doors and retrieve the lost mobile phone. If there is already an unlocked door between two rooms, she may simply pass. She may unlock any locked door if she

is in one of the two rooms connected by it. As she is interested in doing this before the brilliant mind awakes, she needs to make as few moves as possible.

- (a) Design two STRIPS actions, one for crossing from one room to another and one for unlocking a locked door between two rooms. Define the variables to model different aspects of this exercise on your own and describe them in detail.
- (b) You are given the blueprint of the house that Agent 99 received. Rooms are labeled with  $r_1, \dots, r_7$ . Thick lines on the walls indicate a door between two rooms and are labeled with  $d_1, \dots, d_9$ . Agent 99 was told that the doors  $d_2, d_5$ , and  $d_9$  have been left unlocked. Upon arriving in the alternate timeline, she finds herself in the room  $r_1$ , while the phone is in the room  $r_7$ . Formulate the initial state of the given planning setting and use *progression planning* to find a plan to retrieve the phone. What do the goal states look like?



**Exercise 5.3:** Delicious homemade pasta tops any store-bought product and can be made with only two ingredients—eggs and flour. Upon learning this, you decide to make yourself some pasta. To get the process started, you need an egg, some flour, and your pasta machine, all of which can be found in your kitchen or pantry. Since you have too much time on your hands, you formalise the ingredient gathering process as follows:

**Action**( $Take(r, o)$ ),  
 Precond :  $contains(r, o) \wedge in(r)$ ,  
 Effect :  $holds(o) \wedge \neg contains(r, o)$

**Action**( $Put(r, o)$ ),  
 Precond :  $holds(o) \wedge in(r)$ ,  
 Effect :  $\neg holds(o) \wedge contains(r, o)$

**Action**( $Move(r_1, r_2)$ ),  
 Precond :  $in(r_1)$ ,  
 Effect :  $\neg in(r_1) \wedge in(r_2)$

The meaning of the predicates is as follows:

- $in(r)$ : you are located in the room  $r$ ,
- $contains(r, o)$ : the room  $r$  contains the object  $o$ ,
- $holds(o)$ : you are holding the object  $o$ .

Furthermore, you specify your initial state as

$$S := \{in(kitchen), contains(kitchen, egg), contains(pantry, flour), contains(pantry, machine), room(kitchen), room(pantry), object(egg), object(flour), object(machine)\}.$$

Finally, the desired state is formalised as

$$G := \{in(kitchen), contains(kitchen, egg), contains(kitchen, flour), contains(kitchen, machine)\}.$$

Use the STRIPS state-space search algorithm starting in  $S$  (i.e., use *progression planning*) to determine the shortest possible plan that achieves the desired goal state  $G$ . You do not have to draw a complete graph. Simply show how the solution is found.

**Exercise 5.4:** Assume there is a lottery with tickets for €5 and there are three possible prizes: €1000 with a probability of 0.05%, €100 with probability 0.1%, and €1 otherwise.

- (a) What is the expected monetary value of a lottery ticket?
- (b) When is it rational to buy a ticket? Give an inequality involving utilities with the following utilities:  $U(S_k) = 0$ ,  $U(S_{k+5}) = 5 \cdot U(S_{k+1})$ ,  $U(S_{k+100}) = 50 \cdot U(S_{k+5})$ , but there is no information about  $U(S_{k+1000})$ . ( $S_n$  denotes the state of possessing  $n$  Euros.)
- (c) Define  $U(S_{k+5})$  and  $U(S_{k+1000})$  such that a rational agent whose utility function satisfies the equations in Subtask (b) chooses to buy a lottery ticket.

**Exercise 5.5:** In 1713, Nicolas Bernoulli investigated a problem, nowadays referred to as the *St. Petersburg paradox*, which works as follows. You have the opportunity to play a game in which a fair coin is tossed repeatedly until it comes up heads. If the first head appears on the  $n$ -th toss, you win  $2^n$  Euros.

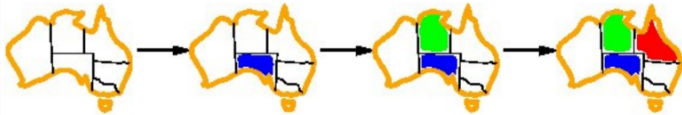
- (a) Show that the expected monetary value of this game is not finite.
- (b) Daniel Bernoulli, the cousin of Nicolas, resolved the apparent paradox in 1738 by suggesting that the utility of money is measured on a logarithmic scale, i.e.,  $U(S_n) = a \log_2 n + b$ , where  $S_n$  ( $n > 0$ ) is the state of having  $n$  Euros and  $a, b$  are constants. What is the expected utility of the game under this assumption? Assume, for simplicity, an initial wealth of 0 Euros and that no stake has to be paid in order to play the game.

## Backtracking search

- ▶ The naive formulation ignored one crucial property of CSPs:
  - Variable assignments are *commutative*, i.e., the order of an assignment of variables does not matter and one reaches the same partial assignment regardless of order.
  - Therefore, CSP search algorithms need only to consider a *single* variable at each node of the search tree!
    - E.g., in the map-colouring problem, initially we may have a choice between  $SA = red$ ,  $SA = green$ , and  $SA = blue$ ,
    - but we would not choose between  $SA = red$  and  $WA = blue$ .
- ▶ With this restriction, we generate only  $d^n$  leaves as expected.
- ▶ Depth-first search for CSPs with single-variable assignments is called *backtracking* search.
  - Backtracking search is the basic uninformed algorithm for CSPs.

## Degree heuristic

- ▶ The MRV heuristic does not help at all in choosing the *first* region to colour.
- ▶ In this case, the *degree heuristic* comes in:
  - it selects the variable that is involved in the *largest number of constraints* on other unassigned variables.
- ▶ In the Australia example,  $SA$  is the variable with highest degree, 5.
  - The others have degree 0, 2, or 3.
  - Actually, once  $SA$  is chosen, we can assign the mainland regions clockwise or counterclockwise with a colour different from  $SA$  and the previous region.



Available Colors:   

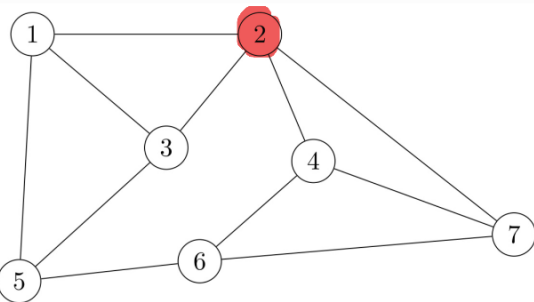
① Initial Node: Use degree heuristic

$$\deg(1) = 3 \quad \deg(4) = 3 \quad \deg(7) = 3$$

$$\deg(2) = 4 \quad \deg(5) = 3$$

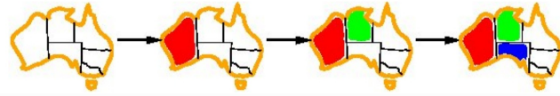
$$\deg(3) = 3 \quad \deg(6) = 3$$

We choose node 2 because it has the highest number of constraints (edges)



## Minimum-remaining-values heuristic

- ▶ The *minimum-remaining-values (MRV)* heuristic:
  - choose the variable with the fewest legal values.
- ▶ If there is a variable  $X$  with 0 legal values remaining, the MRV heuristic will select  $X$  and failure will be detected immediately
  - avoiding pointless searches through further unassigned variables.
- ▶ E.g., in the Australia example, after the assignments for  $WA = red$  and  $NT = green$ , there is only one possible value for  $SA$ .
  - It makes sense to assign  $SA = blue$  next rather than assigning  $Q$ .
  - Actually, after  $SA$  is assigned, the choices for  $Q$ ,  $NSW$ , and  $V$  are all forced.



② Second Node: Use minimum-remaining values heuristic

• calculate how many possible colors a node could have

• select node with minimal possible colors

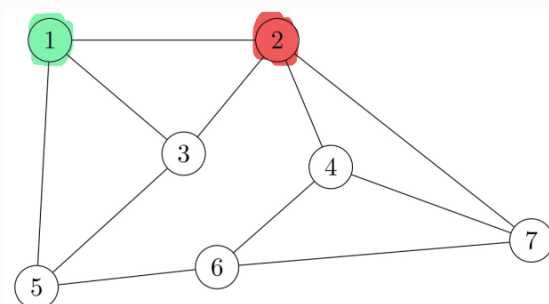
2 possible colors: 1, 3, 4, 7

3 possible colors: 5, 6

• If more than one node found: Additionally select by degree heuristic

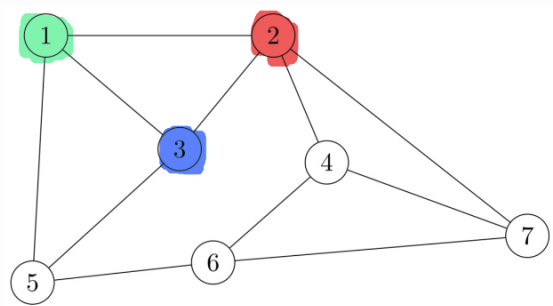
$$\deg(1) = 3 \quad \deg(3) = 3 \quad \deg(4) = 3 \quad \deg(7) = 3$$

• nodes  $\{1, 3, 4, 7\}$  are all valid options. We choose node 1, because it is the smallest number

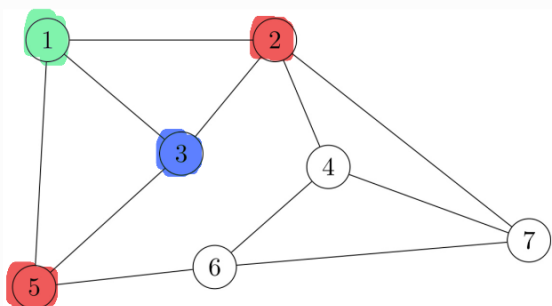




③ Node with lowest possible colors: 3

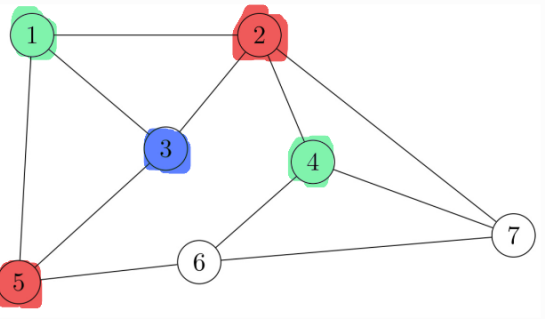
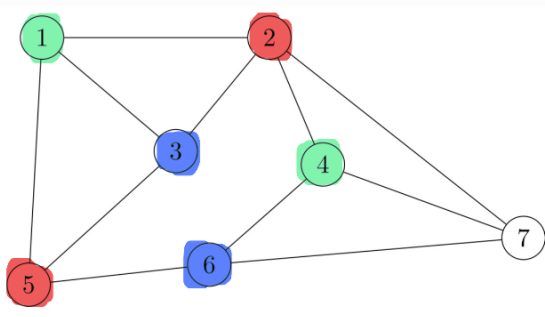


④ Node with lowest possible colors: 5



⑤  
 {4, 6, 7} each have 2 possible values.  
 Choose by degree:  $\text{deg}(4, 6, 7) = 3$   
 Valid options = {4, 6, 7}  
 We choose node 4 because it has the smallest number

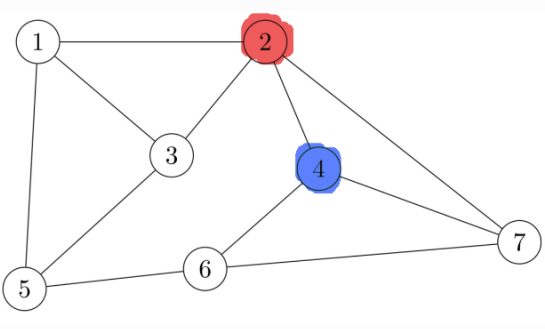
⑥ Node with only 1 possible color: 6



⑦ Because node 7 has no possible coloring, we cannot complete the graph. Here we would backtrack to the "conflict set" (the set of variables that caused the failure)

5.2

Available Colors: ■ ■ ■



Selected Node = 6

• We now use the least-constraining-value heuristic to determine the color of node 6

Possible colors(6) = { ■, ■ }

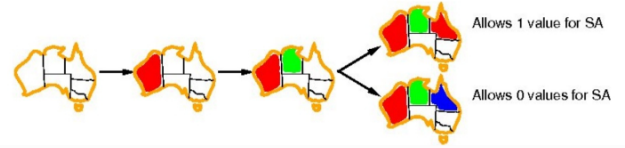
• If we choose Node 6 = ■ we have the following options for neighboring values:

Possible colors(5) = { ■, ■ }



Possible colors(7) = { }


Least-constraining-value heuristic

- Once a variable has been selected, to decide on the order in which to examine its values, the least-constraining-value heuristic can be effective:
  - it prefers a value that rules out the fewest choices for the neighbouring variables in the constraint graph.
- In the Australia example, suppose we have the partial assignment WA = red and NT = green, and our next choice is for Q.
  - Blue would be a bad choice, because it eliminates the last legal value for Q's neighbour SA.
  - The least-constraining-value heuristic thus prefers red to blue.




• If we choose Node 6 =  we have the following options for neighboring values:

Possible colors(5) = { ,  }

Possible colors(7) = {  }

=> the heuristic forces us to pick Node 6 = , because it leaves more options for neighboring nodes. This allows 3 possible values for neighbor nodes.

Node 6 =  is more restricting because it allows only 2 possible values, with Node 7 being unassignable.

5.2a

Move ( $r_1, r_2, d$ )

Precondition:  $At(r_1) \wedge r_1 \neq r_2 \wedge Linked(r_1, d) \wedge Linked(r_2, d)$

Effect:  $\neg At(r_1) \wedge At(r_2)$

Unlock ( $r, d$ )

Precondition:  $At(r) \wedge Locked(d) \wedge Linked(r, d)$

Effect:  $Unlocked(d) \wedge \neg Locked(d)$

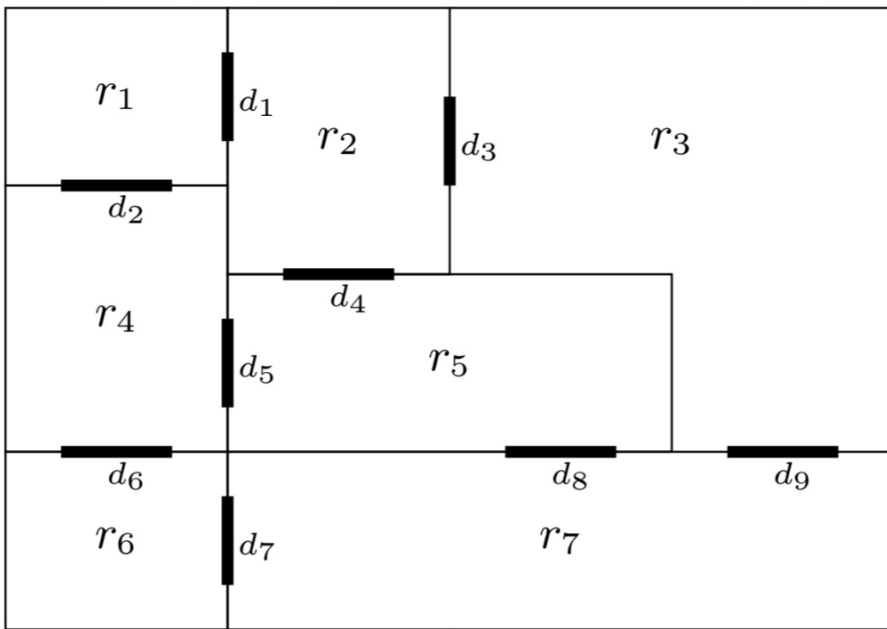
$At(r)$ ... Agent is at room  $r$

$Linked(r, d)$ ... Room  $r$  is connected to door  $d$

$Locked(d)$ ... Door  $d$  is locked

$Unlocked(d)$ ... Door  $d$  is unlocked

5.2b



start =  $r_1$

goal =  $r_7$

unlocked =  $\{d_2, d_5, d_9\}$

locked =  $\{d_1, d_3, d_4, d_6, d_7, d_8\}$

Use Progression Planning: Forward search (from initial state to goal)

We assign each action the value = 1

Because each action has the same value, BFS/A\* will find an optimal solution.

Preface:

The following holds true in every state:

$Linked(r_1, d_1)$   $Linked(r_4, d_2)$   $Linked(r_6, d_6)$

$Linked(r_1, d_2)$   $Linked(r_4, d_5)$   $Linked(r_6, d_7)$

$Linked(r_2, d_1)$   $Linked(r_4, d_6)$   $Linked(r_7, d_3)$

$Linked(r_2, d_3)$   $Linked(r_5, d_4)$   $Linked(r_7, d_8)$

$Linked(r_2, d_4)$   $Linked(r_5, d_5)$   $Linked(r_7, d_9)$

$Linked(r_3, d_3)$   $Linked(r_5, d_8)$

For better readability

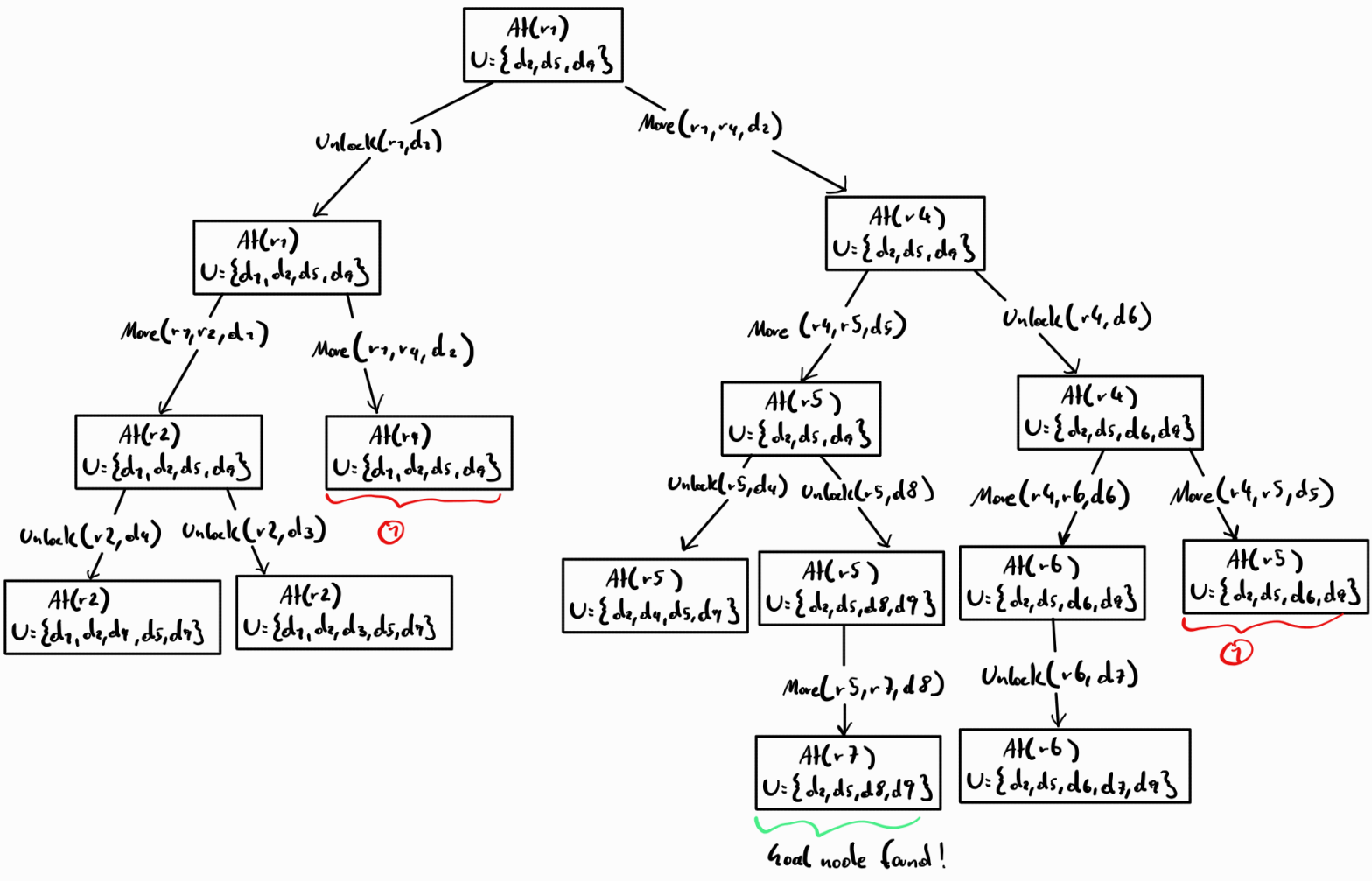
$Unlocked(d_2) \wedge Unlocked(d_5) \wedge Unlocked(d_9)$

will be written as

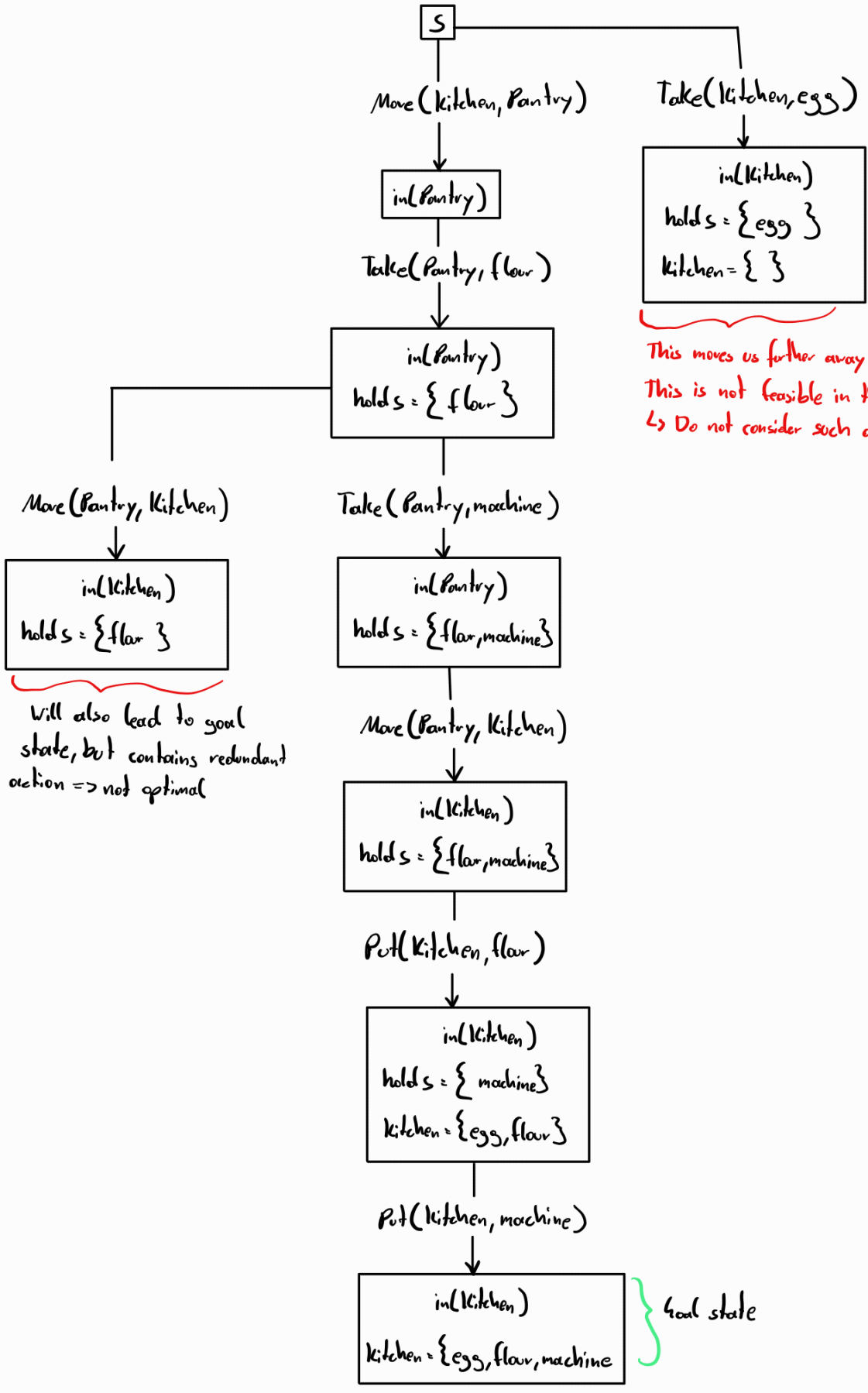
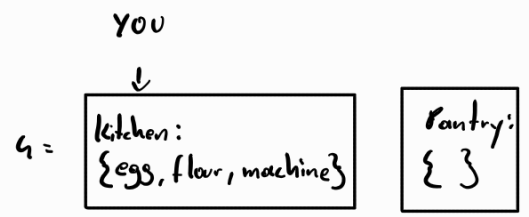
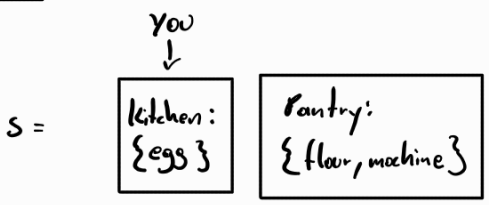
$U = \{d_2, d_5, d_9\}$

① Node does not need expansion as it was already seen with a lower cost. (i.e. this path cannot be optimal because doors were opened that will not be used)

Forward Search BFS:



5.3



This moves us further away from the goal state  
 This is not feasible in this environment  
 ↳ Do not consider such actions

Will also lead to goal state, but contains redundant action => not optimal

Goal state

### 5.4a

$$P(0,0005) = 1000 \text{ €}$$

$$P(0,001) = 100 \text{ €}$$

$$P(0,9985) = 1 \text{ €}$$

$$E = 1000 \cdot 0,0005 + 100 \cdot 0,001 + 1 \cdot 0,9985$$

$$E = 1,5985 \text{ €}$$

The intrinsic value of the ticket is 1,5985 €

But it costs 5 € so the expected value comes to:

$$1,5985 \text{ €} - 5 \text{ €} = -3,4015 \text{ €}$$

### Expected Utility

- ▶ The *expected utility* of an action  $a$  given evidence  $e$ , denoted  $EU(a|e)$ , is the average utility value of the outcomes, weighted by the probability that the outcome occurs:

$$EU(a|e) = \sum_{s'} P(\text{RESULT}(a) = s' | a, e) U(s')$$

- ▶ *Principle of maximum expected utility* (MEU):
  - a rational agent should choose the action that maximises the agent's expected utility:

$$\text{action} = \underset{a}{\text{argmax}} EU(a|e).$$

### 5.4b

$$U(S_{k+1000}) = ?$$

$$U(S_{k+100}) = 50 \cdot U(S_{k+5})$$

$$U(S_{k+5}) = 5 \cdot U(S_{k+1})$$

$$U(S_{k+1}) = ?$$

$$U(S_k) = 0$$

$$E(\text{buy}) = 0,0005 \cdot U(S_{k+1000}) + 0,001 \cdot \underbrace{U(S_{k+100})}_{50 \cdot \underbrace{U(S_{k+5})}_{5 \cdot U(S_{k+1})}} + 0,9985 \cdot U(S_{k+1})$$

$$E(\text{buy}) = 0,0005 \cdot U(S_{k+1000}) + 0,001 \cdot 50 \cdot 5 \cdot U(S_{k+1}) + 0,9985 \cdot U(S_{k+1})$$

$$E(\text{buy}) = 0,0005 \cdot U(S_{k+1000}) + 0,25 \cdot U(S_{k+1}) + 0,9985 \cdot U(S_{k+1})$$

$$E(\text{buy}) = 0,0005 \cdot U(S_{k+1000}) + 1,2485 \cdot U(S_{k+1})$$

$$E(\text{cost}) = U(S_{k+5}) = 5 \cdot U(S_{k+1})$$

The ticket is worth buying, if the expected value is larger than the cost of buying a ticket

$$E(\text{buy}) > E(\text{cost}) \Rightarrow \text{must be true}$$

$$0,0005 \cdot U(S_{k+1000}) + 1,2485 \cdot U(S_{k+1}) > 5 \cdot U(S_{k+1}) \quad | -1,2485 \cdot U(S_{k+1})$$

$$0,0005 \cdot U(S_{k+1000}) > 3,7515 \cdot U(S_{k+1}) \quad | : 0,0005$$

$$U(S_{k+1000}) > 7503 \cdot U(S_{k+1})$$

5.4c

Define  $U(S_{k+1}) = 1, U(S_{k+5}) = 5$

Solve  $U(S_{k+1000}) > 7503 \cdot U(S_{k+1})$

$$U(S_{k+1000}) > 7503$$

So any solution in  $\{U(S_{k+5}) = 5, U(S_{k+1000}) > 7503\}$  is valid.

For example  $U(S_{k+5}) = 5, U(S_{k+100}) = 50 \cdot U(S_{k+5}) = 250, U(S_{k+1000}) = 37 \cdot U(S_{k+100})$

5.5a

Basically roulette strategy. Because there is no "green", you break even.

$$\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \cdot 2^i = \sum_{i=0}^{\infty} \left(\frac{1}{2} \cdot 2\right)^i = \sum_{i=0}^{\infty} 1^i = \sum_{i=0}^{\infty} 1$$

5.5b

$$U(S_n) = a \log_2 b + b \mid S_n (n > 0)$$

$$EU(U) = \sum_{i=0}^{\infty} \underbrace{\left(\frac{1}{2}\right)^i}_{\text{Monetary reward}} \cdot 2^i$$

$$U = \sum_{i=0}^{\infty} (a \log_2(2^i) + b) \cdot \left(\frac{1}{2}\right)^i$$

$$U = \sum_{i=1}^{\infty} (ai + b) \cdot \left(\frac{1}{2}\right)^i = a \sum_{i=0}^{\infty} i \cdot \left(\frac{1}{2}\right)^i + b \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i - \left(\frac{1}{2}\right)^0$$

$$= a \underbrace{\sum_{i=0}^{\infty} i \cdot \left(\frac{1}{2}\right)^i}_2 + b \underbrace{\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i}_2 - 1$$

$$= 2a + b$$

Geometrische Reihe

$$\sum_{k=0}^{\infty} q^k = \frac{1}{1-q}$$

Sum of infinite geometric series

$$\sum_{i=0}^{\infty} ar^n = \frac{a}{1-r}$$