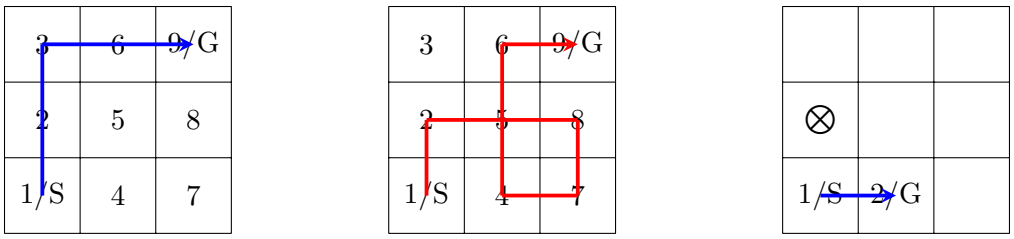
Computational Intelligence WS24/25

Exercise Sheet 1 (Solution) — October 31st, 2024

Thomas Gabor, Maximilian Zorn, Claudia Linnhoff-Popien

1 Simple Grid Environment

Consider the simple grid world example below, where an agent can move on *tiles* but is not allowed to stand still, starting from the initial position (S) with the purpose of reaching the goal tile (G). The agent can move between all numbered tiles that are not *walls* (outside the grid) or *blocked* (⊗). Execution is stopped on reaching the goal tile.



(i) Consider the small grid-world shown in Example 1a. Define a set of possible observations $\mathcal{O}$ and a set of possible actions $\mathcal{A}$ in accordance to the definition given in the lecture. For your definition, what are $|\mathcal{A}|$ and $|\mathcal{O}|$?

Note: Recall that for any finite set $X = \{x_0, ..., x_n\}$ we write $|X| = n$ for the number of elements in X. For infinite sets, we also write $|X| = \infty$.

$$\mathcal{A} = \{\texttt{go\_N}, \texttt{go\_E}, \texttt{go\_S}, \texttt{go\_W}\}$$

$$|\mathcal{A}| = 4$$

$$\mathcal{O} = \{\ (\mathit{tile\_N}, \mathit{tile\_E}, \mathit{tile\_S}, \mathit{tile\_W}, \mathit{current\_tile\_number}, \mathit{done}) : \mathit{tile\_N}, \mathit{tile\_E}, \mathit{tile\_S}, \mathit{tile\_W} \in \{\texttt{wall}, \texttt{blocked}, \texttt{tile}\}, \mathit{current\_tile\_number} \in [1;9] \subset \mathbb{N}, \mathit{done} \in \mathbb{B}\ \}$$

$$|\mathcal{O}| = 3^4 \cdot 9 \cdot 2 = 1458$$

(ii) Similarly, for the grid world in Example 1a, define a goal predicate γ_1 that accepts a policy that (a) finds the way from the start tile (S) to the goal tile (G) in the *optimal amount of steps* and (b) only steps on tile-numbers which are *strictly bigger* than any of the tile-numbers visited before. Then provide a solution path, i.e., π_a, for which $\gamma_1(\pi_a) = \text{True}$. Which of the goal classes we have covered in the lecture could γ_1 fall under?

$$\gamma_1 : \Pi \to \mathbb{B}$$

$$\gamma_1(\pi) \Leftrightarrow_{\text{def}} (\forall t : (o_t.\mathit{done} = \mathit{True} \Leftrightarrow t = 4) \wedge o_t.\mathit{current\_tile\_number} > o_{t-1}.\mathit{current\_tile\_number}).$$

→ Possible trajectories: `[1,2,3,6,9]` or `[1,2,5,8,9]` or `[1,4,7,8,9]` etc.
→ Goal predicates are goal class 1. If we consider a goal predicate that can easily be split up (by writing "$(\forall t \dots) \wedge (\forall t \dots)$", for example), it could be goal class 1.5.

(iii) Now consider the grid world shown in Example 1b. For the given solution trajectory π_b (red line) give $\langle a_t \rangle_{t \in \mathcal{Z}}$ and $\langle o_t \rangle_{t \in \mathcal{Z}}$. What is the response from $\gamma_1(\pi_b)$ in this case?

$$\langle a_0, \dots, a_7 \rangle = \langle \texttt{go\_N}, \texttt{go\_E}, \texttt{go\_E}, \texttt{go\_S}, \texttt{go\_W}, \texttt{go\_N}, \texttt{go\_N}, \texttt{go\_E} \rangle$$

$$\begin{aligned} \langle o_0, \dots, o_8 \rangle = \langle & (\texttt{tile}, \texttt{tile}, \texttt{wall}, \texttt{wall}, 1, \mathit{False}), \\ & (\texttt{tile}, \texttt{tile}, \texttt{tile}, \texttt{wall}, 2, \mathit{False}), \\ & \dots, \\ & (\texttt{wall}, \texttt{wall}, \texttt{tile}, \texttt{tile}, 9, \mathit{True}) \rangle \end{aligned}$$

→ $\gamma_1(\pi_b) = \mathit{False}$ (failing both criteria)

(iv) Recall that in the lecture we are covering simple (policy) search approaches, one of them being *random search* (see Algorithm 1 below) with n attempts, i.e., $\rho(n)$. Let us assume that random sampling $\pi \sim \Pi$ can only produce policies that always execute *valid* actions, i.e., never produce sequences of actions that would lead the agent to run into a wall or blocked field. What is the probability of the policy π_b being found by $\rho(8)$ in the grid world Example 1b? Briefly explain your answer.

Algorithm 1 (random search (policy)). *Let $\mathcal{A}$ be a set of actions. Let $\mathcal{O}$ be a set of observations. Let $\Gamma \subseteq (\mathcal{O} \to \mathcal{A}) \to \mathbb{B}$ be a space of goal predicates on policy functions. Let $\gamma \in \Gamma$ be a goal predicate. We assume that the policy space $\Pi \subseteq \mathcal{O} \to \mathcal{A}$ can be sampled from, i.e., $\pi \sim \Pi$ returns a random element from Π. Random search for n samples is given via the function*

$$\rho(n) = \begin{cases} \emptyset & \text{if } n = 0, \\ \pi & \text{if } n > 0 \text{ and } \gamma(\pi) \text{ where } \pi \sim \Pi, \\ \rho(n-1) & \text{otherwise.} \end{cases}$$

Hint: You can assume that the probability that a random policy $\pi \sim \Pi$ will execute *valid* action a at time step t from the set of *valid* actions A_t at time step t is given via $\Pr(\pi(o_t) = a \mid \pi \sim \Pi) = \frac{1}{|A_t|}$.

The probability for reproducing π_b exactly via $\rho(1)$ is the product of each taken transitional probability for each action, i.e.:

$$\begin{aligned} &\Pr\left(a = \langle \texttt{go\_N}, \texttt{go\_E}, \texttt{go\_E}, \texttt{go\_S}, \texttt{go\_W}, \texttt{go\_N}, \texttt{go\_N}, \texttt{go\_E} \rangle \mid \pi \sim \Pi\right) \\ &= 1/2 \cdot 1/3 \cdot 1/4 \cdot 1/3 \cdot 1/2 \cdot 1/3 \cdot 1/4 \cdot 1/3 \\ &= 1/5184 \quad \approx \quad 0.000192 \end{aligned}$$

We then compute chance of 1 success in 8 trials, finding the exact policy that would sequentially chose like the policy π_b, at least once, with the complementary probability:

$$\implies \Pr\left(\rho(8) = a\right) \approx 0.00154 \qquad \text{(according to Wolfram Alpha)}$$

(v) Finally, provide a tile-numbering and goal-position in the Template 1c such that $\gamma_1(\rho(4))$ is *guaranteed* to be *True* and explain why. You may set one blocking tile $\bigotimes$ (on any tile except the initial position (S)) that cannot be traversed.

Example for *guaranteed* policy on Template 1c see above. (Alternative: *Blocked* to the right and 2/(G) on tile above.)
This placement reduces the possible actions from $\Pr\left(a \sim \{\texttt{go\_N}, \texttt{go\_E}\}\right) = 1/2$ to $\Pr\left(a \sim \{\texttt{go\_E}\}\right) = 1/1$.
Any other combination risks the agent potentially going back and forth between two tiles indefinitely.

2 Squirrel Environment

For scientific purposes, we want to deploy a *SquirrelBot*, i.e., a small robotic *agent* that is able to drive across soil and dig for nuts. It can observe its exact location $p \in \mathcal{L}$ with $\mathcal{L} = [0; 100] \times [0; 100] \subset \mathbb{R}^2$ on a continuous 2D plane representing the accessible soil. In the same plane it can also observe a marked target location $g \in \mathcal{L}$ that it wants to navigate to. The value of g is provided by a *MemoryAgent* that tries to remember all locations where nuts are buried, but to the *SquirrelBot* that location g (like its own location p) is just part of its observation. The *SquirrelBot* can execute an action a of the form $a = (\delta x, \delta y, dig) \in \mathbb{R} \times \mathbb{R} \times \mathbb{B}$ once per time step. The action is resolved by the environment by updating the robot's own location by $\delta x, \delta y$ and then digging at the new location iff $dig = True$. However, all actions that attempt to drive a distance greater than 1 (i.e., $\sqrt{(\delta x)^2 + (\delta y)^2} > 1$) per time step are completely ignored by the environment.

(i) Assume that the complete state of the system is given by the position p_t of the *SquirrelBot* at time step t, the position of the marked target location g_t, and a flag dug_t marking if the *SquirrelBot* attempted to dig (after driving) at time step t, i.e., the whole system generates a sequence of states $\langle s_t \rangle_{0 \leq t \leq T}$ for some fixed maximum episode length $T \in \mathbb{N}$ and $s_t \in \mathcal{L} \times \mathcal{L} \times \mathbb{B}$. Give a goal predicate $\gamma_1 : \langle \mathcal{L} \times \mathcal{L} \times \mathbb{B} \rangle \to \mathbb{B}$ so that $\gamma_1(\langle p_t, g_t, dug_t \rangle_{0 \leq t \leq T})$ holds iff the agent has at one point in time attempted to dig at a location nearer than 1 to the target location g.

Note: The *state* reflects all information contained in both the agent and the environment, as opposed to the observations which may only contain parts of it. Since there is nothing more we could know about the policy, we can thus also define goal predicates to be evaluated on that system state.
Hint: You can use the function $\texttt{dist} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ to compute the Euclidean distance between two points in $\mathcal{L}$, i.e., $\texttt{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

$$\gamma_1(\langle p_t, g_t, dug_t \rangle_{0 \leq t \leq T}) \iff \exists t : \texttt{dist}(p_t, g_t) < 1 \wedge dug_t = True$$

(ii) Given a *SquirrelBot* with the same information (p_t, g_t, dug_t), now also give a goal predicate $\gamma_2 : \langle \mathcal{L} \times \mathcal{L} \times \mathbb{B} \rangle \to \mathbb{B}$ so that $\gamma_2(\langle p_t, g_t, dug_t \rangle_{0 \leq t \leq T})$ holds iff the agent has, presumably running out of robot patience, attempted to dig at every second step for exactly 10 steps, coming closer to the goal each time and reaching the goal by digging (within a distance of less than 0.01) on the final, 10th of these steps, thus completing the trajectory.

$$\begin{aligned}\gamma_2(\langle p_t, g_t, dug_t\rangle_{0\le t\le T}) \iff & \texttt{dist}(p_T, g_T) < 0.01 \\ & \wedge\, dug_T = \mathit{True} \\ & \wedge\, \forall t' \in \{T-8, T-6, T-4, T-2\} : \\ & \quad \texttt{dist}(p_{t'}, g_{t'}) > \texttt{dist}(p_{t'+2}, g_{t'+2}) \\ & \quad \wedge\, dug_{t'} = \mathit{True}\end{aligned}$$

(iii) Assume that the whole plane of soil is without obstacles and thus easily navigable for the *SquirrelBot*. Give a policy π that always fulfills the goal predicate γ_1 eventually regardless of the initial state. Also give π's type signature.
Hint: You do not need to construct the fastest such policy.

$$\pi : \mathcal{L} \times \mathcal{L} \to \mathbb{R} \times \mathbb{R} \times \mathbb{B}$$

$$\pi\big((x_p, y_p), (x_g, y_g)\big) = \begin{cases} (+0.1, 0, \mathit{False}) & \text{if } x_g - x_p > 0.1, \\ (-0.1, 0, \mathit{False}) & \text{else if } x_g - x_p < -0.1, \\ (0, +0.1, \mathit{False}) & \text{else if } y_g - y_p > 0.1, \\ (0, -0.1, \mathit{False}) & \text{else if } y_g - y_p < -0.1, \\ (0, 0, \mathit{True}) & \text{otherwise.} \end{cases}$$

3 Running Example: Vacuum World

For the various classes of the goal hierachy it may be intuitively helpful to code the running examples along as we develop them formally.

(i) Implement in `Python` a simple *Vacuum World* as we have seen it in the lecture. The implementation should adequately model the concepts of *observations* and *actions*, which a *policy* π should be able to exert. A randomly acting policy will suffice initially.

(ii) In the lecture we have discussed the following *goal predicate* γ:

γ should hold iff the agent does not execute the same action for all observations,

or more formally:

$$\gamma(\pi) \iff \neg\exists a \in \mathcal{A} : \forall o \in O : \pi(o) = a.$$

In your `Python` implementation, collect a number of actions and observations for a policy of your choice, then implement and verify the goal predicate γ.