Homework 3

CS 411 - Artificial Intelligence I - Fall 2019

Matteo Corain 650088272

September 19, 2019

1 Question 1

1.1 Atomic representation

An *atomic representation* of an environment is characterized by the fact that each possible state in which the environment can be is described by a single item (for example, a name that uniquely identifies the particular state), without any internal structure. Externally, we may know how states are connected to each other and how to transition from a state to another, but we do not have any information on how the state is internally structured; in other words, each state is represented as a "black box", with the only property of being uniquely identifiable.

Atomic representations are very simple to manage, but they have a very limited expressiveness. For this reason, hey cannot be applied in complex environments, in which we have to account for multiple factors; in fact, in those case we would need to keep track of a new state for each minimal variation of any of those, making the representation of the state space very convoluted.

1.2 Factored representation

By using a *factored representation*, each state of the environment is characterized by a set of *variables*, each with its own *value*. In this way, the environment is described at a much finer level of detail, splitting up the characterization of the different, independent factors that identify its state into a set of distinct properties. Additionally, different states to share the same values for certain variables (we can tell that two states are somehow similar, while in the previous case we can only say that they are different).

Factored representations are more expressive than atomic representations, making them much more natural, convenient and concise for the usage with large state spaces. Moreover, they can be useful to represent the uncertainty on the current state that is characteristic of partially observable environments: unobserved variables may in fact simply treated as "null" values and disregarded for the selection of the next action. On the other hand, they require additional information to be known from the environment: if we represent an environment in a factored way, in fact, we must be able to list all the relevant variables and the way they are captured and used upfront.

1.3 Structured representation

In a *structured representation*, the environment is described in terms of *objects* interacting with each other; those interactions between objects are represented by means of *relationships*. Object relationships are dynamic: they do not need to be identified upfront, but may be added or removed on-the-fly when needed.

Structured representations are even more expressive than factored representations since they can take describe the dynamicity of the environment structure in a very concise way. On the other hand, this kind of representation may become very complex, requiring agents to perform more expensive computations to make an efficient use of the collected information.

2 Question 2

2.1 Atomic representation

If we represent the robot's environment using the atomic approach, we first need to identify each possible state of the world and associate it with a name. For example, the state represented in figure may be associated with identifier s_x:

$$s_x = \{\text{agent in square 5, stars in locations 2, 4, 11}\}$$

Supposing that the state of the environment depends on 13 independent factors, namely the position of the robot and the presence or absence of a star in each of the 12 locations, the total number of states that we have to identify for this kind of environment is equal to:

$$n_s = 12 * 2^{12} = 49152$$

This derives from the fact that we have 12 possible locations for the robot and 2^{12} possible arrangements of stars on the terrain. Once that states have been defined, we need to define the relationships among them, which describe how the agent is able to transition from a state to another. For example, state s_x is connected to the following set of states:

- $s_y = \{\text{agent in square 1, stars in locations 2, 4, 11}\}$, to which the agent can transition by taking the *Up* action;
- $s_z = \{\text{agent in square 6, stars in locations 2, 4, 11}\}$, to which the agent can transition by taking the *Right* action;
- $s_w = \{\text{agent in square 9, stars in locations 2, 4, 11}\}$, to which the agent can transition by taking the *Down* action.

It is important to point out that the agent, in this case, has no notion of the different factors that characterize the environment; it is only able to determine somehow what state it is in. Given the size of the state space, this representation is not very convenient as it would generate a very large network of relationships that the agent would need to store somehow to be used for the selection of the action to take. Nonetheless, it is very simple to use: if the agent knows what the current state is (or infer it from its percepts), then it can simply understand what actions it can take to transition towards the one that yields the best expected reward (just following the tree of relationships).

2.2 Factored representation

If we represent the robot's environment using the factored approach, we have to identify a set of properties that are able to describe completely all the characteristics of the environment that are relevant to the agent to decide what action to take. For this kind of environment, those properties are, as already mentioned, the *location* of the robot and the *position* of the stars. A complete, factored representation of the robot environment can therefore be seen as a data structure with 13 distinct fields:

- A *robot-location* field, allowing for integer values from 1 to 12, which identifies the location of the robot;
- Twelve *star-in-location-x* fields (with x varying from 1 to 12), allowing for boolean values *true* or *false*, which characterize the presence or absence of a star in the considered location.

In this representation, each distinct state is identified by a unique combination of values assigned to the 13 variables. For example, the current state is characterized by:

- *Robot-location = 5*;
- *Star-in-location-2 = true*, *star-in-location-4 = true*, *star-in-location-11 = true*;
- Remaining fields set to *false*.

After having defined the structure of the environment, we have to specify how the different actions of the agent affect the representation of the state; in general, each action will have an effect only on a subset of the variables of interest. These effects are not codified, as in the atomic case, in a state graph that the robot can traverse; instead, the agent function should implement some form of reasoning to analyze the possibilities of acting to modify the state in the desired way. In this case, the robot may modify the state of the environment in the following ways:

- By taking one of the *Up*, *Left*, *Down*, *Right* actions, which modify the current state by setting the *robot-location* variable to the identifier of the square the robot is moving to;
- By taking the *Pick* action, which modifies the current state by setting to *false* the *star-in-location-x* variable relative to the position of the picked star.

This representation is way more manageable (in terms of space) with respect to the state space graph that is generated from an atomic representation; in fact, the agent has to memorize only the thirteen variables of interest to fully determine the state of the environment. As a drawback, this representation requires more thinking from the robot to decide what the next action to take is going to be, since it needs to analyze all the different characteristics of the environment. Moreover, exactly as in the atomic case, this representation of the state is very specific to the particular environment we are considering; for example, if we moved the robot in an environment using a different grid of locations, then this representation would not have been appropriate.

2.3 Structured representation

If we represent the robot's environment using the structured approach, we have to identify a set of objects and the relationships using which they interact with each other. In this environment, the following objects may be identified:

- The *robot*, which represents the intelligent agent;
- The *stars*, which represent the objects the agent is willing to pick up;
- The *squares*, which represent a unit of location.

The three entities interact with each other using a complex set of relationships; those include:

- The *located-in(robot, square)* relationship, used to identify the current location of the robot;
- The *positioned-on(star, square)* relationship, used to identify the current position of a star.

Using this representation, each state is uniquely identified by a specific set of objects connected by a specific set of relationships. For example, in the presented case, the current state is described by the following set of objects and relationships:

- A single *robot* object, three *star* objects and twelve *square* objects;
- A *located-in(robot, square-5)* and three *positioned-on(star-1, square-2)*, *positioned-on(star-2, square-4)* and *positioned-on(star-3, square-11)* relationships.

With this kind of representation, the actions the agent can take are used to modify the relationships among the different objects, or possibly to create or destroy new objects. In the presented case, the robot may act in the following ways:

- By taking one of the *Up*, *Left*, *Down*, *Right* actions, which modify the current state by deleting the current *located-in(robot, square)* relationship and creating a new one with the square the robot is moving to;
- By taking the *Pick* action, which modifies the current state by deleting the *positioned-on(star, square)* relationship of the picked star.

This kind of representation is the most general of all, since the objects and the relationships among objects can be dynamically modified if needed; for example, this representation is easily adaptable to the case in which the robot that is moved into an environment with a different size, or in which it can collect different kinds of objects. It is also very compact (we just need to store the current set of objects and relationships), extensible (it is straightforward to add new objects or relationships) and readable (relationships are expressed in quasi-natural language). On the other hand, it is the most difficult to process, since the information the robot may gather at different points of time are complex and changing, requiring proper interpretation.