Lab session 7: Agents based on Propositional logic

Pierre-Alexandre Murena

May 31, 2024

1 An implementation of propositional logic

In file `PL.py`, you can find an implementation of propositional logic. This code will help you create and manipulate propositional sentences. Propositional sentences are encoded in the `Sentence` class.

Using inheritance, we define several kinds of sentences: *variables*, *negations* (of the form $\neg p$ where p is a sentence), *or* (of the form $p_1 \vee p_2$ with p_1, p_2 two sentences), *and* (of the form $p_1 \wedge p_2$ with p_1, p_2 two sentences), *implies* (of the form $p_1 \rightarrow p_2$ with p_1, p_2 two sentences), *if and only if* (of the form $p_1 \leftrightarrow p_2$ with p_1, p_2 two sentences), *conjunctions* (of the form $p_1 \wedge \ldots \wedge p_n$ with $p_1, \ldots, p_n$ sentences), and *disjunctions* (of the form $p_1 \vee \ldots \vee p_n$ with $p_1, \ldots, p_n$ sentences).

Propositional variables are given a name, which is the symbol that will be used when printing the sentences. This name is passed as input to the constructor:

```
a = sVariable("a")
```

You can then combine variables into sentences. For instance:

```
sOr(a, sNeg(a))
```

encodes the proposition $a \vee \neg a$.

Question 1. Create the proposition $p = (a \wedge (\neg b \rightarrow a))$. For that, first instantiate the propositional variables a and b, and then create p.

The semantic level of propositional logic involves the *intepretation* of sentences. An interpretation maps each propositional variable to a Boolean and computes the truth value of the sentence, using some specific rules. In this implementation, an intepretation is a dictionary mapping each variable to its truth value. For instance:

```
I = { "a":  True, "b":  False }
```

The truth value of a proposition can be computed by calling the method `value`. For instance, with this choice of I:

```
sAnd(a, b).value(I)
```

returns `False`.

Question 2. Compute the truth value of p for this interpretation I.

The *models* of a propositional sentence are the interpretations that make the sentence true. In order to find these models, it is important to know which are the variables involved in the sentence, which is possible using method `variables`. The models of a sentence are obtained using method `get_models`.

Question 3. What are the models of p?

Question 4. Using the methods `variables` and `get_models`, implement `is_tautology`, a method which outputs True if the sentence is a tautology and False otherwise.

Sometimes, we are not interested in *all* models of a sentence, but in knowing whether there is one. Using the `get_models` method can be long in case we have a large number of variables. But even in this case, we can still quickly come up to an interpretation making a sentence true, which indicates that the sentence is *satisfiable*. The method `bruteForceSAT` implements a brute-force approach to finding such an interpretation[1].

Question 5. Use this method to check that $q = a \rightarrow (b \vee \neg a)$ is satisfiable.

2 Enumeration method (Bonus, 2 points)

The enumeration method is used to verify that $\alpha \models \beta$, for two propositional sentences α and β. The method works as follows: It finds all the models of α and checks that they are also models of β.

Question 6. Implement the enumeration method in function `entails(s1, s2)`. The function returns `True` if $s_1 \models s_2$ and `False` otherwise.

Question 7. Use your implementation to check that $(a \vee c) \wedge (b \vee \neg c) \models a \vee b$.

3 Playing Go... No, kidding: Monkey and Banana again

The goal of this exercise is to use the given implementation of propositional logic to solve a problem you know very well now: the monkey and banana problem.

The main task will be to properly describe the problem in the language of propositional logic and to implement it. Once a proper description is done, you will show how solving the problem reduces to SAT solving.

[1]This method is definitely not optimal, but remember that the SAT problem is NP-complete, which means that it is difficult to solve in the worst case even with more sophisticated approaches.

3.1 Describing the environment

We consider an environment of length n. For each position x, we introduce the variable B_x indicating whether the banana is in cell x. In the code, the names of these variables are `B[0]`, ..., `B[n-1]`.

Question 8. In the code, the position of the banana is given by `bananaPosition`. In list `B`, add the sentence(s) indicating the position of the banana.

3.2 Actions and successor-state axioms

We consider 4 actions: `Left`, `Right`, `Grab` and `Interact`. The first three actions are familiar to you. The `Interact` action corresponds to climbing onto the box if the monkey is on the floor, and going down if the monkey is on the box. In the code, `actionVariables` is a list containing all action variables at all time steps. For instance, `actionVariables[3]["Left"]` returns the propositional variable indicating whether action `Left` is played at $t = 3$.

We describe the impact of the actions using *successor-state axioms*, for each of the fluent variables.

Question 9. What are the fluent variables in the monkey and banana problem?

3.2.1 Box position

The box position is represented by the variables P_x^t, reading "The box is at position x at time t". This variable can be retrieved in the code using `box_positions[t][x]`.

For each time, a successor-state axiomis of the form:

$$P_x^{t+1} \leftrightarrow (P_x^t \wedge \text{condition to stay there}) \vee (P_{x'}^t \wedge \text{condition to move to } x)$$

For instance, the box is located in cell 0 at time $t + 1$ if and only if:

- It was located in cell 0 at time t, and the agent performed `Left`, `Interact` or `Grab`.
- It was located in cell 1 at time t, and the agent performed `Right`.

The corresponding propositional sentence is then:

$$P_0^{t+1} \leftrightarrow (P_0^t \wedge (Left^t \vee Interact^t \vee Grab^t)) \vee (P_1^t \wedge Left^t)$$

All the seccessor-state axioms for the box position are already implemented.

3.2.2 Monkey up

An important parameter to take into account in the problem is whether the monkey is on the box. This is represented by the variables Up^t. They can be retrieved in the list `monkey_up`.

Question 10. Express the conditions for the monkey to be on the box at time $t + 1$, separating the case where the monkey was already up at time t and the case where it was not. Express them in English first, then in propositional logic, writing the successor-state axiom. Implement it.

3.2.3 Grabbing

We need to keep track of the fact that the monkey is grabbing the banana. This is represented by the variables $GrabbingBanana^t$. They can be retrieved in the list `grabbing_banana`.

Question 11. Express the conditions for the monkey to be grabbing the banana at time $t+1$, separating the case where the monkey was already grabbing it at time t[2] and the case where it was not. Express them in English first, then in propositional logic, writing the successor-state axiom. Implement it.

3.3 Constraint axioms

This representation has a main limitation. It does not state some aspects which may seem obvious to us, but are not: the unicity of actions (at each time t, the agent can play one single action) and of positions (the box can be at only one cell at a time).

Question 12. Write the unicity of actions rule in propositional logic.

The rule (as well as the rule for the unicity of positions) is already implemented in the code[3]

3.4 Search problem

Initially, the monkey is on the floor, not holding the banana, and the box is in cell 0.

Question 13. Implement the propositional sentence describing the initial state.

The goal for the monkey is to be grabbing the banana after its last action, at time $T-1$.

Question 14. Implement the propositional sentence describing the goal of the monkey.

The overall problem is obtained by taking the conjunction of all the propositional sentences built so far. It contains the description of the environment, of the actions and their consequences, of the constraints, of the initial state and of the goals of the agent. We call this sentence ϕ.

An interpretation of ϕ necessarily describes the truth value of all propositional variables, in particular of the actions. In other words, an interpretation describes the plan of the agent. An interpretation that does not make ϕ true is not consistent with the problem (either because it does not respect the rules, does not reach the goal etc). Conversely, if one can find an interpretation that makes ϕ true, this interpretation should yield a valid choice of actions.

Question 15. Using `bruteForceSAT`, find an interpretation of ϕ in the case where $n = 2$, the banana is located in 0 and $T = 3$. Observe what happens if you "forget" some constraints.

[2]We assume here that, once it got its banana, the monkey will keep it in hand. After all, it required 7 weeks of work, and at least 5 different implementations to make this poor animal get its banana. It deserves to keep it, don't you think so?

[3]If you are smart and read this before answering the question, here are some bad news: the code is not transparent enough to give you the answer!