Artificial Intelligence II – Assignment 5

Solve the following problems. (200 points)

Question 1. (30 points) Consider the missionaries and cannibals problem. In the class, we use the constraint

:- time(T), holds(at(I,J,L),T), I>0, J>I .

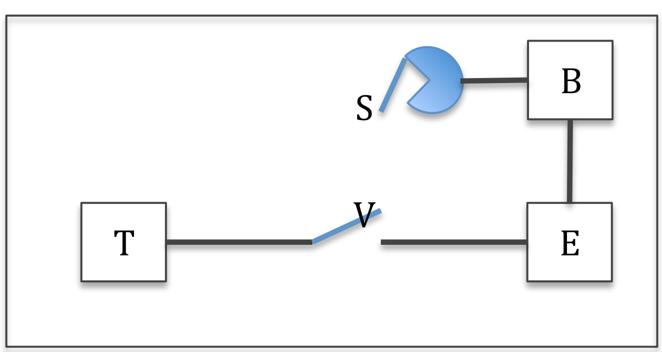
to eliminate answer sets for which there are time steps in which cannibals outnumber missionaries on a bank. In this question, you are asked to **compile** this information into the action precondition. Download the program from the class website and

- delete this constraint from the program
- modify the remaining of the program so that it still gives the right answer. (**Hint**: you should change the preconditions of the actions so that the constraint is enforced!)

Question 2. (120 points) Consider again the missionaries and cannibals problem. The paper "Missionaries and Cannibals in the Causal Calculator" by Vladimir Lifschitz presented several variations of this problem (posted on the class website). Develop answer set programming representation for at least 4 variations to the problem.

Submit your description for each variation that you select together with the code to solve the problem. You can use the code provided in the class as the initial code. Your submission should be succinct and clearly state what the code can do. For example, a variation is given in Section 7, which says ''Boat can carry three'' can be described by the first sentence of the section "The boat can carry three. Then four pairs can cross.'' It means that your program needs to be able to generate a plan so that four missionaries and four cannibals can cross the river.

Question 3. (50 points) Consider a very simplified schematic representation of a car given in the following picture.



S denotes the switch. B denotes the battery, E the engine, T the tank, and V the valve of the tank. We know the following:

- The valve can be opened or closed (in our car, it is always closed!).
- The engine is on or off.
- The engine can have gasoline or not.
- The car is running or not running.
- The engine is damaged or not damaged.
- The switch is at the on or off position.
- The switch can be toggled between the on and the off position by turning the switch on or off. It means that if we turn the switch off (or on) and the switch is at the on (or off) position, it will be at the off the switch (or on) position.
- We do not turn the switch on or off when the car is running.
- When the valve is closed then gasoline is supplied from the tank to the engine. We assume that the tank always has gasoline.
- When the switch is at the on position, the engine is on.
- When the switch is at the off position, the engine is off.
- When the engine is on and the valve is closed then the car will run.
- When the engine is on and the valve is open then the car will be damaged.
- When the engine is damaged then the car will not run.

Write an ASP program that allows us to reason about the status of the car. The suggested set of fluents and actions are:

- <u>*Fluents*</u>:
  - switch_on, switch_off,
  - valve_opened, valve_closed
  - car_running
  - engine_on, engine_off
  - gasoline_available (for the engine)
  - damaged (the engine is damaged)
- <u>*Actions*</u>:
  - turn_switch_on, turn_switch_off
  - open_valve, close_valve

Make sure that you distinguish between static causal laws and direct effects of actions. An example is ``closing the valve causes the valve to be in the *closed* state and an indirect effect of this action is that the engine has gasoline!''

Given the initial state as in the picture (the valve is open and the switch is at the off position) and assume that the engine is not damaged, the engine does not have gasoline. What happens if we execute the following sequence of actions

- close_valve then turn_switch_on
- turn_switch_on then close_valve then turn_switch_on

Make sure that your program yields correct answers to the above questions.

Bonus Question. This bonus question is worth 10 points of the midterm.

The new **clingo** system can interact with Python and includes a number of interesting features. One of this is to allow the interleaving between model computation and program updating. One of the examples can be found in the source package (**clingo-4.5.3-cource.tar.gz**), folder examples/clingo/robots. If you run **python visualize.py** you will see the interaction. The goal of this second question is to create a program that can do the following:

Given two programs P and Q.
Your program needs to execute the following loop:

do
compute an answer set X of program P
add X to Q
compute an answer set of the program X ∪ Q
until there is no more answer set of P

Example:

P = {a :- not b.   b:- not a}
Q = {c :- a.   d :- b. }

First iteration: X = {a}, output {a,c} as answer set of X ∪ Q.
Second iteration: X = {b}, output {b,d} as answer set of X ∪ Q.
Then stop!