C231: Answer Set Programming

Mark Law

March 4, 2018

This tutorial aims for you to practice using the stable model semantics. Some of these problems require the use of *clingo*. Clingo 5 is installed on the lab machines and can be found at /vol/lab/clingo5/clingo. You should write your ASP programs and run the command `/vol/lab/clingo5/clingo -n 0 {filename}`. Clingo will then compute the Answer Sets of your program. If you wish instead to see the ground program, you should run with the option "$--$`text`". Note that this grounding is sometimes simpler than what we compute as $\mathcal{RG}(P)$, as clingo will simplify things wherever possible.

Question 1

For the following normal logic programs P and sets X:

a) Calculate $\mathcal{RG}(P)$

b) Calculate $(\mathcal{RG}(P))^X$

c) Is X an answer set of P?

d) Write down all answer sets of P (No proof required).

e) Check your answers using Clingo.

f) If X is a model of P but is not an answer set of P, find an unfounded subset of X

i) $X = \left\{ \begin{array}{c} \texttt{heads(c1)}, \texttt{tails(c2)}, \\ \texttt{coin(c1)}, \texttt{coin(c2)} \end{array} \right\}$

$$P = \left\{ \begin{array}{l} \texttt{heads(X)} \leftarrow \texttt{not tails(X)}, \texttt{coin(X)}. \\ \texttt{tails(X)} \leftarrow \texttt{not heads(X)}, \texttt{coin(X)}. \\ \texttt{coin(c1)}. \\ \texttt{coin(c2)}. \end{array} \right.$$

ii) $X = \left\{ \begin{array}{c} \texttt{heads(c1)}, \texttt{heads(c2)}, \\ \texttt{tails(c1)}, \texttt{tails(c2)}, \\ \texttt{coin(c1)}, \texttt{coin(c2)} \end{array} \right\}$

$$P = \left\{ \begin{array}{l} \texttt{heads(X) :- not tails(X)}, \texttt{coin(X)}. \\ \texttt{tails(X) :- not heads(X)}, \texttt{coin(X)}. \\ \texttt{heads(X) :- tails(X)}, \texttt{coin(X)}. \\ \texttt{tails(X) :- heads(X)}, \texttt{coin(X)}. \\ \texttt{coin(c1)}. \\ \texttt{coin(c2)}. \end{array} \right.$$

iii) $X = \{\texttt{p}\}$ and $X = \emptyset$

$$P = \left\{ \begin{array}{l} \texttt{p :- not p.} \\ \texttt{p :- p.} \end{array} \right.$$

iv) $X = \{\texttt{p}, \texttt{q}\}$,

$$P = \left\{ \begin{array}{l} \texttt{p :- q.} \\ \texttt{q :- p.} \end{array} \right.$$

Question 2

A common mistake when writing ASP programs is to write a rule that you know has a finite number of ground instances that can be satisfied, but that causes an infinite relevant grounding. The program P has an infinite grounding. Add single literal to the first rule in the program in order to keep the relevant grounding finite (without changing the intended meaning of the program).

$$P = \left\{ \begin{array}{l} \texttt{alarm(T+1) :- alarm(T),not stop(T).} \\ \texttt{alarm(1).} \\ \texttt{stop(2) :- not stop(9).} \\ \texttt{stop(9) :- not stop(2).} \\ \texttt{time(0..10).} \end{array} \right.$$

The next two questions are reformulations of questions from earlier in the course.

Question 3

Consider the biological process of lactose metabolism by the bacterium E. Coli. The background knowledge includes the following pieces of information:

> E. coli can feed on the sugar lactose (*lact*) if it makes two enzymes permease (*perm*) and galactosidase (*gala*). Enzymes (E) are made if they are coded by a gene (G) that is expressed. Enzyme permease is coded by gene *lac(y)* and enzyme galactosidase is coded by gene *lac(z)*. These genes are part of a cluster of genes (*lac(X)*), that is expressed when the amounts of glucose (*gluc*) are low (*lo*) and lactose (*lact*) are high (*hi*).

1. Define an abductive inference task that explains the lactose metabolism (*feed(lact)*) of E. Coli, by addressing the following three points:

   a Using the signature $\mathcal{L}$ given below, model in ASP the above pieces of information as background knowledge KB:
   $\mathcal{L} = \{feed/1, make/1, code/2, express/1, amt/2, code/2, sugar/1\}$.

   b Define the set A of abducibles as ground instances of predicates *amt* and *sugar*.

   c Define two integrity constraints that state respectively that the amount of a substance (S) may not be both high and low; and that the amount of a substance can only be known if the substance is a sugar.

2. Write an ASP program that models the abductive task constructed in part (1) with the goal `feed(lact)`.

3. Solve the program using clingo to find an abductive solution.

Question 4

Consider the following legal reasoning extract. The background knowledge includes the following pieces of information:

> People *born in* USA are USA *citizen*. People *born outside* USA but *resident of* USA and *naturalised*, are USA *citizen*. People *born outside* USA but *registered* in USA and with USA *citizen* mother, are USA *citizen*. Mary is John's *mother*. Mary is USA *citizen*.

1. Define an abductive inference task that explains the possibilities of *John be USA citizen*, by addressing the following three points:

   a Using the signature $\mathcal{L}$ given below, model in ASP the above pieces of information as background knowledge KB:
   $\mathcal{L} = \{citizen/2, mother/2, bornIn//2, bornOut/2, resident/2,$
   $naturalised/2, registered/2\}$.

   b Define the set A of abducibles as ground instances of the predicates *bornIn*, *bornOut*, *resident*, *naturalised*, *registered*.

   c Define the integrity constraint that states that John is not USA *resident*.

2. Write an ASP program that models the abductive task constructed in part (1) with the goal `citizen(john, usa)`.

3. Solve the program using clingo to find an abductive solution.

Question 5

A graph G is said to be Hamiltonian if there is a cycle that passes through every node exactly once. Using only normal rules, choice rules and hard constraints, write a program P such that P combined with facts describing the nodes and edges in any graph G is satisfiable if and only if G is Hamiltonian.

a) *i*) Write a rule that states that any edge in the graph may or may not be in the Hamilton cycle (defining the predicate `in/2`).

ii) Define a predicate `reach/1` such that `reach(X)` holds if and only if the node X is *reachable* from the first node (i.e. if there is a path from `1` to `X`).

iii) Using only constraints, complete the program. You should test it with a few different graphs. You can use the not equals operator (`!=`). Note that this operator is evaluated at ground time, so no rule occurs in the grounding such that `t != t` occurs in the body, for any term `t`.

b) Check whether the graphs represented by the following sets of facts are Hamiltonian. For those that are, compute the reduct wrt the relevant answer set.

i)
```
node(1..4).
edge(1,2).
edge(2,3).
edge(3,2).
edge(2,4).
edge(4,1)
```

ii)
```
node(1..4).
edge(1,2).
edge(2,3).
edge(3,4).
edge(4,1)
```

iii)
```
node(1..4).
edge(1,2).
edge(2,3).
edge(3,4).
```

iv)
```
node(1..4).
edge(1,2).
edge(2,1).
edge(3,4).
edge(4,3).
```

Question 6

Construct the required reducts to check whether each interpretation X is an answer set of each program P.

a) $P =$
```
p;q.
p :- not p.
q :- not q.
```

b) $P =$
```
p;q.
p :- q.
 :- not q.
```


i) $X = \emptyset$

ii) $X = \{\texttt{p}\}$

iii) $X = \{\texttt{q}\}$

iv) $X = \{\texttt{p, q}\}$

Question 7

Reconsider the normal logic programs from Question 1:

a) Calculate $\mathcal{RG}(P)^X_{FLP}$

b) Is X an answer set of P?

Question 8

Choice rules are often defined by translation to counting aggregates and disjunction. A choice rule $\texttt{lb}\{\texttt{h}_1,\ldots,\texttt{h}_\texttt{m}\}\texttt{ub :- b}_1,\ldots,\texttt{b}_\texttt{n}$ becomes the rules:

$$\begin{array}{c} \texttt{h}_1 \lor \widehat{\texttt{h}_1} \texttt{ :- } \texttt{b}_1,\ldots,\texttt{b}_\texttt{n}. \\ \ldots \\ \texttt{h}_\texttt{m} \lor \widehat{\texttt{h}_\texttt{m}} \texttt{ :- } \texttt{b}_1,\ldots,\texttt{b}_\texttt{n}. \\ \texttt{:- } \texttt{b}_1,\ldots,\texttt{b}_\texttt{n}, \texttt{not lb } \#\texttt{count}\{\texttt{h}_1 : \texttt{h}_1,\ldots,\texttt{h}_\texttt{m} : \texttt{h}_\texttt{m}\}\texttt{ub}. \end{array}$$

(where each $\widehat{\texttt{h}_\texttt{i}}$ is a new atom that does not occur in the rest of the program)
The answer sets of P are then the answer sets of the translated program (with all the new atoms removed).

Consider the program $P = \left\{ \begin{array}{l} \texttt{1\{p;q\}2 :- r.} \\ \texttt{0\{r\}1.} \end{array} \right\}$.

a) For each X below, compute the "direct" reduct for choice rules given in unit 8 and determine whether X is an answer set of P.

b) Use the transformation above to translate the choice rules into an ASP program with disjunction and aggregates. Then for each X below that you determined was an answer set in part (a), give an X' that is an answer set of your translation such that $X' \cap HB_P = X$. By computing the FLP reduct of your translation, show that X' is indeed an answer set of the translation.

i) $\emptyset$

ii) {r}

iii) {p}

iv) {r, p}

v) {r, p, q}

Question 9

Consider following set of facts that define the structure of a sudoku grid:

```
block(1..9).            row(1..9).              column(1..9).
possible_value(1..9).

in_block(1,1..3,1..3).  in_block(2,1..3,4..6).  in_block(3,1..3,7..9).
in_block(4,4..6,1..3).  in_block(5,4..6,4..6).  in_block(6,4..6,7..9).
in_block(7,7..9,1..3).  in_block(8,7..9,4..6).  in_block(9,7..9,7..9).
```

Download the file `sudoku.lp` from CATE and write the following rules, so that it has a single answer set, containing the solution of the sudoku problem. You should check this using clingo 5 (the grounding of the program is too big to work with by hand!).

i) Write a disjunctive rule that says that every cell takes a value between 1 and 9 (using the predicate `value/3`, where the 3 arguments are `row`, `column` and `value`).

ii) Write a constraint using a counting aggregate that expresses that no block can contain the same value more than once.

iii) Write a constraint using a counting aggregate that expresses that no row can contain the same value more than once.

iv) Write a constraint using a counting aggregate that expresses that no column can contain the same value more than once.

Question 10

Consider the magic squares problem. You are given an `n` $\times$ `n` grid and you must place the numbers `1` to `n` in the grid, such that each number appears exactly once and all of the rows, columns and the two diagonals sum to the same number.

Complete the program "`magic_squares.lp`" so that when it is combined with a fact `size(n)`, it computes the magic squares of size `n` $\times$ `n`.

1. First, you should complete the program using a single choice rule, and a single constraint containing two aggregates (expressing that no two lines can sum to different integers).

2. Now try to rewrite your constraint using just one aggregate atom (hint: $X \neq Y$ if and only if $X - Y \neq 0$).

3. Use clingo to compute the size groundings of these two programs with the grid size 3 (run "`clingo --text file.lp | wc -l` " to pipe the grounding to word count and count the number of lines) – interrupt the first program if it takes more than a few seconds! What do you notice about the size of each grounding?

Question 11

For each of the abductive tasks in Questions 3 and 4, reconsider the ASP program you wrote. Add weak constraints to each so that the shortest abductive solution is preferred.

Question 12

Translate the following user journey preferences into weak constraints:

1. My highest priority is safety. I would like to avoid walking through areas with a high crime rating (4 or higher).
2. My next priority is money. I would like to spend as little as possible.
3. Finally, and least importantly, I would prefer to walk as short a distance as possible.

You should use the language:

$$\{\texttt{crime\_rating(Leg,CR)},\ \texttt{cost(Leg,C)},\ \texttt{distance(Leg,D)},\ \texttt{CR1} < \texttt{CR2},\ \texttt{mode\_of\_transport(Leg,M)}\}.$$

Now, for each of the following pairs of interpretations, compute the scores at each priority level and decide which interpretation is preferred.

a)

$$\text{i)} \left\{ \begin{array}{l} \texttt{mode\_of\_transport(1,walk)}, \\ \texttt{distance(1,100)}, \\ \texttt{crime\_rating(1,5)}, \\ \texttt{cost(1,0)}, \\ \texttt{mode\_of\_transport(2,bus)} \\ \texttt{distance(2,3000)}, \\ \texttt{crime\_rating(2,2)}, \\ \texttt{cost(2,2)} \end{array} \right\} \qquad \text{ii)} \left\{ \begin{array}{l} \texttt{mode\_of\_transport(1,bus)}, \\ \texttt{distance(1,100)}, \\ \texttt{crime\_rating(1,5)}, \\ \texttt{cost(1,2)}, \\ \texttt{mode\_of\_transport(2,bus)} \\ \texttt{distance(2,3000)}, \\ \texttt{crime\_rating(2,2)}, \\ \texttt{cost(2,2)} \end{array} \right\}$$

b)

$$\text{i)} \left\{ \begin{array}{l} \texttt{mode\_of\_transport(1,walk)}, \\ \texttt{distance(1,2000)}, \\ \texttt{crime\_rating(1,5)}, \\ \texttt{cost(1,0)}, \\ \texttt{mode\_of\_transport(2,bus)} \\ \texttt{distance(2,3000)}, \\ \texttt{crime\_rating(2,2)}, \\ \texttt{cost(2,2)} \end{array} \right\} \qquad \text{ii)} \left\{ \begin{array}{l} \texttt{mode\_of\_transport(1,walk)}, \\ \texttt{distance(1,100)}, \\ \texttt{crime\_rating(1,5)}, \\ \texttt{cost(1,0)}, \\ \texttt{mode\_of\_transport(2,bus)} \\ \texttt{distance(2,4000)}, \\ \texttt{crime\_rating(2,2)}, \\ \texttt{cost(2,2)}, \\ \texttt{mode\_of\_transport(3,walk)}, \\ \texttt{distance(3,200)}, \\ \texttt{crime\_rating(3,3)}, \\ \texttt{cost(3,0)} \end{array} \right\}$$

Question 13

Express of the following statements as a constraint containing an aggregate:

1. People should not consume more than 14 units of alcohol a week. You should use the language {`consume(Person, Drink, Units, Week)`, `person(Person)`, `week(Week)`}.
2. At least two rooms per building should contain fire extinguishers. You should use the language {`building(Building)`, `room(Building, Room)`, `in(Room, Object)`, `fire_extinguisher(Object)`}.