Session 3 – Solutions

Exercise 1.

1. ```
   print(onemodel(T,S) ~= nil)
   ```

   Or just use `print(sat(T,S))`

2. Optimal model expansion.

   ```
   term colAmount : V {
     #{c: ? a[Area]: Coloring(a)=c}
   }

   procedure main() {
     stdoptions.nbmodels = 1
     sol = minimize(T,S,colAmount)[1]
     print(value(colAmount,sol))
   }
   ```

3. Model expansion

   ```
   procedure main() {
     stdoptions.nbmodels = 2
     models = modelexpand(T,S)
     print(models[1] ~= nil and models[2] == nil)
   }
   ```

4. New theory + model expansion

   ```
   theory Corner5 : V {
     solution(1,1) = 5.
   }

   procedure main() {
     NewTheory = merge(T,Corner5)
     printmodels(allmodels(S,NewTheory))
   }
   ```

5. Make vocabulary and theory that defines what a route is between two stations. Then make term stating the length of the route between the given stations. Then optimal model expansion will provide you the answer.
   ```

6. Make a query.
7. Define new theory + optimal propagate.

```
theory Row3NoDouble5 : V {
  #{c[column] : solution(3,c) = 5} <= 1.
}

procedure main() {
  consequences = optimalpropagate(Row3NoDouble5,S)
  // extract the actual consequences into separate
      partial structures
}
```

8. Add the negation to the initial partial solution, and apply the `unsatstructure` procedure to retrieve the answer.

```
procedure minimumExplanation(partialS,consequence,rules
    ) {
  negativeCons = // negation of consequence, use
      makefalse procedure
  Sunsat = merge(partialS,negativeCons)
  return unsatstructure(rules,Sunsat)
}

procedure main() {
  consequences = optimalPropagate(Row3NoDouble5,S)
  for consequence in consequences do
    minimumExplanation(S,consequence,Row3NoDouble5)
  end
}
```

9. We optimal propagate first with a single theory. If no new consequences are found, try with combining two theories, etc. Some parts are left in pseudo code since they would require some boilerplate code.

```
procedure main() {
  partialS = copy(initialS)
  n = 1
  while partialS is not complete do
    for subset Ts of size n of our theories do
      Tnew = merge(Ts)
      cons = optimalpropagate(Tnew,partialS)
      if cons ~= partialS then
        for all consequence in cons do
          print(minimumExplanation(partialS,
              consequence, Tnew))
        end
        partialS = cons
        n = 0
```

```
          break
        else
      end
      n = n + 1
    end
  }
```

Note that the ordering of the rules is not taken into account yet. One could do this in the for loop over the subsets of the theories by prefering first theories with a combined difficulty as low as possible.

10. By splitting the theory into small theories that are easy to understand by humans you can use the above procedure. To see such procedure in action for Einstein puzzle, take a look at `https://bartbog.github.io/zebra/origin/`.

Exercise 2. For queries solving the questions, see solution `.idp` file on Toledo

1. True
2. True
3. Strange question; $M100$ has no prerequisites, hence trivially true. He instructs each of the 0 courses.
4. False
5. Note: John is not an element of the domain of the database, a database will answer: false, but in logic the truth value of this sentence will be undefined. Note: in the IDP system you cannot write a query like this (since John is not in the vocabulary)
6. False
7. True

Exercise 3. For queries solving the questions, see solution `.idp` file on Toledo

Exercise 4. See `.idp` file on Toledo.

Exercise 5. UNA: 3, 4, 5
DCA: 1, 3
In this case, UNA is satisfied if no more than one arrow arrives at a node, and DCA is satisfied if every node can be reached from 0.

Exercise 6.

UNA(τ):

$$\forall a, x : \texttt{Nil} \neq \texttt{Cons}(a, x).$$
$$\forall a, x, b, y : \texttt{Cons}(a, x) = \texttt{Cons}(b, y) \Rightarrow a = b \wedge x = y.$$

DCA(τ): We introduce a new auxiliary predicate $U/1$ (see definition 2.5.5. in the course notes (p.93)). DCA(τ) can then be expressed as follows:

$$
\begin{aligned}
&\{\\
&\quad U(Nil).\\
&\quad \forall h[A], l[list] : U(Cons(h,l)) \leftarrow U(l).\\
&\}\\
&\forall l[list] : U(l).
\end{aligned}
$$

Exercise 7.

```
vocabulary V {
  type Dir
  North : Dir
  West : Dir
  South : Dir
  East : Dir
}

theory T : V {
  ! x[Dir] : x = North | x = West | x = South | x = East.
  North ~= West.
  North ~= South.
  North ~= East.
  West ~= South.
  West ~= East.
  South ~= East.
}
```