Informatics 2D: Tutorial 4

Constraint Satisfaction and First-Order Logic*

Week 5

1 The Crop Allocation Problem

Consider the following problem in bio-dynamic farming (where some crops grow better next to particular crops)[1] for the specific land division shown in Figure 1.

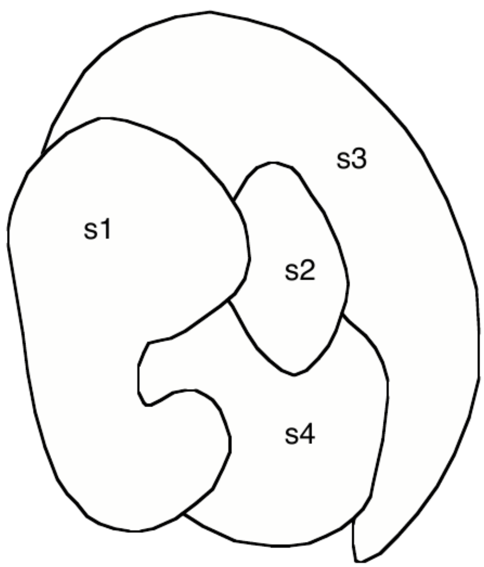

Figure 1: The Bio-Dynamic Farming Problem.

The figure shows the allocation of a piece of land for planting four different crops using the constraints of bio-dynamic farming. In this kind of farming, the idea is that there are groups of crops that develop better if set in particular arrangements. Also the balance of nutrients in the soil is used to decide what to plant where. Here are the constraints according to the current levels of nutrients in the soil:

*Credits: Kobby Nuamah, Michael Rovatsos

[1] Adapted from an original problem set by Mellish & Fisher.

1. Sector 1 (s1) can be planted with one of the following crops: {cabbage, kale, broccoli, cauliflower}
2. Sector 2 (s2) can be planted with one of the following crops: {cabbage, kale, broccoli}
3. Sector 3 (s3) can be planted with one of the following crops: {kale}
4. Sector 4 (s4) can be planted with one of the following crops: {kale, broccoli}

The constraint here is that we do not want two sectors that are adjacent to each other to be planted with the same crops.

How does this look when expressed as a constraint satisfaction problem (CSP)? What are the stages that the AC-3 algorithm goes through in obtaining arc consistency for this example? (see Figure 2 for the AC-3 algorithm)

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  inputs: *csp*, a binary CSP with components (X, D, C)
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add (X_k, X_i) to *queue*
  return *true*

---

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i
  revised $\leftarrow$ *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised $\leftarrow$ *true*
  return *revised*

Figure 2: The AC-3 algorithm.

2 First-Order Logic

Part 1: Represent the following sentences in first-order logic. You will have to define a vocabulary (which should be consistent between sentences).

1. Some students took French in spring 2001.
2. Every student who takes French passes it.
3. Only one student took Greek in spring 2001.
4. The best score in Greek is always higher than the best score in French.
5. There is a male barber who shaves all the men who do not shave themselves.

Part 2: Write down a first-order logic sentence such that every world in which it is true contains exactly one object.

3 Most General Unifier (MGU)

The most general unifier (MGU) is the least constrained substitution that makes two clauses unify with each other. What is the MGU for each pair of clauses below? If there is no MGU, explain why.

The Unify algorithm in figure 3 (also in R&N Section 9.2, p.328.)

1. $p(A, B, B)$ and $p(x, y, z)$
2. $q(y, g(A, B))$ and $q(g(x, x), y)$
3. older(father(y), y) and older(father(x), John)
4. knows(father(y), y) and knows(x, x)

Note that, constants are upper case (e.g. A, B) and variables are lower case (e.g. x, y, z).

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
   inputs: x, a variable, constant, list, or compound
           y, a variable, constant, list, or compound
           θ, the substitution built up so far (optional, defaults to empty)

   if θ = failure then return failure
   else if x = y then return θ
   else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
   else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
   else if COMPOUND?(x) and COMPOUND?(y) then
       return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
   else if LIST?(x) and LIST?(y) then
       return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
   else return failure
---------------------------------------------------------------------
function UNIFY-VAR(var, x, θ) returns a substitution
   inputs: var, a variable
           x, any expression
           θ, the substitution built up so far

   if {var/val} ∈ θ then return UNIFY(val, x, θ)
   else if {x/val} ∈ θ then return UNIFY(var, val, θ)
   else if OCCUR-CHECK?(var, x) then return failure
   else return add {var/x} to θ
```

Figure 3: Unification Algorithm.

4 *More to learn[2]

- Second order logic quantifies also over relations (not only over variables). Does this make a difference? Would an agent that understands second order logic be more useful thanan agent that can process only FOL?

[2]Starred *problems are outside the examinable course content. Feel free to ignore them completely.