Informatics 2D: Tutorial 4

Satisfiability and First-Order Logic

Week 5

1 DPLL algorithm

The DPLL algorithm consists of the following steps:

- Convert proposition to CNF
- Loop through the following steps until a satisfying assignment is found or none is possible:
  - Loop through the following simplifications until the formula can't be simplified anymore:
    - Pure literal heuristic.
    - Unit Clause heuristic.
  - Select a variable and branch the search space into a formula where the variable is true and a formula where the variable is false. (This means that you try the algorithm recursively upon these new formulae, with a satisfying assignment for one of the new formula being a satisfying assignment for the original).

Your lecture notes and R&N chapter 7 section 6 describe the steps in more detail.

Question: Use the DPLL algorithm to show whether the following propositional formulae is satisfiable:

$S_{1,1} \wedge (S_{1,1} \Leftrightarrow W_{1,2} \vee W_{1,1} \vee W_{2,1}) \wedge \neg((W_{1,2} \wedge P_{1,2}) \vee (W_{2,1} \wedge P_{2,1})) \wedge \neg P_{1,1} \wedge \neg((W_{1,1} \wedge W_{2,1}) \vee (W_{1,1} \wedge W_{1,2}))$

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 1 (also in R&N Section 9.2, p.328.)

```
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 1: Unification Algorithm.

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).

4 *More to learn[1]

- 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 than an agent that can process only FOL?

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