

# Programing Directions for Assignment 2

CSS486/2506 - Knowledge Representation and Reasoning

Fall 2008

## 1 Platforms to be used

The platforms to be used are the same as the ones used for assignment 1. Please, refer to the guidelines for assignment 1.

## 2 Input Format

In exercise 4, you are asked to write a program to calculate the *difference* between two concepts  $d_1$  and  $d_2$ . Here, we discuss the input format for your program.

First, concepts, roles, and primitive descriptions are represented with atom names in *lowercase letters*. Complex descriptions are built using lists where the first element is either atom `all` or atom `and`. Notice that all these atoms are in small letter because atoms in capital letters are variables in Prolog.

An input file will consist of two concepts  $d_1$  and  $d_2$ . Each description is represented as a list with atoms and operators `AND`, `ALL` in lowercase (e.g., `[and,p,[and,p,r]]` would be a legal description representation). Examples of legal input files for both Prolog and Scheme are shown in figure 1.

If you do not use Prolog or Scheme you should choose one of the two representations and state which one you use in your documentation.

## 3 What your program should provide

If you use Prolog or Scheme we provide template files (`compDiff.pl` and `compDiff.scm` respectively) which already have the code necessary for reading a concept-description pair from a file. These two files define a top-level procedure called `computeDiff` whose only argument is the name of a file containing a set of clauses.

Procedure `computeDiff` reads the pair of concepts and calls `difference` with the pair. Procedure `difference` takes the pair of concepts in question, and when the first subsumes the second, returns its difference. In Prolog, `difference/3` will return the difference in the third argument; whereas, in Scheme, `difference` will just take two arguments and evaluate into the difference. **It is this procedure `difference` that you have to implement (by modifying the template file)**. Finally, `computeDiff` will print out the result returned by your `difference` procedure.

For the example given in figure 1, the prolog call:

```
?- difference([and,p,[and,q,r]], [and,[and,q,t],[and,p,s],r], D).
```

should succeed with computed answer `D = [and,t,s]`.

Similarly, in Scheme, an evaluation of

```
(difference '(and p (and q r)) '(and (and q t) (and p s) r))
```

```
(and p (and q r))  
(and (and q t) (and p s) r)
```

(a) Scheme representation

```
[and,p,[and,q,r]].  
[and,[and,q,t],[and,p,s],r].
```

(b) Prolog representation

Figure 1: Two files containing a two concepts.

should evaluate to `(and t s)`.

If you are a language other than Prolog or Scheme, your program itself should be named `compDiff` and either take a filename or two concept-descriptions as arguments. Remember to **explicitly** state in your documentation the *exact commands* used for compiling and running your program in either CDF or CS.

Finally, you should *explicitly* provide in your documentation interesting pair of concepts with which you have tested your program.

## 4 How to submit your program

You should submit a printed copy of your program together with its documentation with your assignment in class. Moreover, you should submit your code to `jabaier /at/ cs.toronto.edu` as an attachment file. Note that code received *after the due date* will count as a late assignment.