

# Programing Directions for Assignment 2

CSS486/2506 - Knowledge Representation and Reasoning

Fall 2007

## 1 Platforms to be used

The preferred programming languages for the course are the following two:

1. **Prolog:** your program has to run under SWI-Prolog, which is available for Linux and Windows at <http://www.swi-prolog.org/> together with full documentation.  
You can run SWI-Prolog in CDF (execute `pl`) and CSLAB (execute `/usr/bin/swipl` on the application machines).
2. **Scheme:** your program has to run under MIT Scheme. You can download a copy from <http://www.gnu.org/software/mit-scheme/> for Linux or Windows.  
MIT Scheme is installed in both CS and CDF. Just type `scheme` to run it.

## 2 Input Format

This section discusses the input format for the program you must write for question 5.

Primitive concepts, roles, and operators are represented with atom names in *lowercase* such as `thing`, `child` and `adult` respectively. Complex concepts are built using lists where the first element in the list is either atom `all`, `exists`, `at-most` or `and`. Notice that all these atoms are in lowercase because atoms in capital letters are variables in Prolog.

The input file will contain exactly two lines, each one defining a concept. 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.

The following are three example files in the Scheme format.

file1.txt:

```
(and (all employee canadian))
(all employee (and american canadian))
```

file2.txt:

```
(exists 1 teacher)
(and (exists 2 teacher) (all teacher male))
```

file3.txt:

```
(and (all cousin (exists 0 friend)) (all employee female))
(and (at-most 0 employee) (all friend (at-most 3 cousin)))
```

The following are three example files in the Prolog format (note the dot at the end of each line).

file1.txt:

```
[and, [all, employee, canadian]].
[all, employee, [and, american, canadian]].
```

```
file2.txt:
[exists,1,teacher].
[and,[exists,2,teacher],[all,teacher,male]].
```

```
file3.txt:
[and,[all,cousin,[exists,0,friend]],[all,employee,female]].
[and,[at-most,0,employee],[all,friend,[at-most,3,cousin]]].
```

### 3 What your program should provide

If you use Prolog or Scheme we provide sample files (`subsume.pl` and `subsume.scm` respectively) which already have the code necessary for reading two concepts from a file. Both programs define a top-level procedure called `subsume` whose only argument is the name of a file containing a set of clauses.

Procedure `subsume` reads the clauses in the file. The two concepts read are passed as arguments to `is_subsumed`. It is **this** procedure, `is_subsumed`, that you have to implement by modifying the sample file.

The program must print which concept subsumes which (i.e. “the first concept subsumes the second”), or a message indicating that no concept subsumes the other.

Finally, if you are using Java or C, your program itself should be named `subsume` and it should take as its only argument the name of the file where it should read the set of clauses. A call to your program should have the following form: `subsume <filename>`. No templates for reading input files will be provided for those languages.

Recall that you should explicitly say in your documentation the exact command used for compiling your program in either CDF or CS.

### 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 /DOT/ toronto.edu` as an attachment file. Note that code received *after the due date* will count as a late assignment.

**No matter what platform you choose, remember that your program has to run error-free in CDF or CS.** So, if you worked elsewhere, we recommend you try your program in either of these machines before handing in the code.