

CSIS0270 Artificial Intelligence, 2003–2004

Assignment 3

Deadline May 2, 2004, 5:00pm.

This assignment contains both a written part (Questions 1) and some programming parts (Questions 2 and 3). For the written part, hand-in your answers to the assignment box (B3). For the programming parts, hand-in the programs that you write via the hand-in system of the department.

1. **Resolution Proof (20%)** In mathematics, each function (like our First-Order Predicate functions) has a declared **domain** and **co-domain**: for a function f with domain A and co-domain B , the function can be applied to each value in the set A , and the result will always be in the set B . A function is said to be **surjective** if each value in the co-domain is the result of applying f to some value in the domain.
 - a. Express the above criteria of domain and co-domain in first-order logic. In particular, write a sentence that is true whenever F has domain A and co-domain B .
 - b. Express the above criteria of surjectiveness in first-order logic. In particular, make a sentence that is true whenever F is surjective, assuming that A and B are the domain and co-domain of F . Other than the ones already defined in FOL, you may only use the predicate \in for building your sentence. Convert the logic sentence into CNF. Also, write a sentence that is true whenever F is not surjective, and convert that to CNF as well.
 - c. Suppose we have three sets A , B and C , and two functions F and G . The first function has domain A and co-domain B , the second function has domain B and co-domain C . The **composition**, GoF , takes a value in A , apply function F and then function G to get a value in C . So GoF has domain A and co-domain C . Prove the following theorem using First-order logic resolution refutation: “ **G is surjective**” is entailed by **GoF is surjective**. (Note: you may make use of the axioms defining equality.)
2. **Normalization in Prolog (30%)** Write a predicate, `normalize/2`, which takes a propositional logic expression and convert it into CNF clauses. More precisely, `normalize(Expr, CNFList)` is supposed to be called with `Expr` completely instantiated. It is an expression, of one of the following form:
 - `neg(Expr)`: the negation of `Expr`.
 - `and(Expr1, Expr2)`: the conjunction of `Expr1` and `Expr2`, which are themselves expressions.
 - `or(Expr1, Expr2)`: the disjunction of `Expr1` and `Expr2`, which are themselves expressions.
 - `if(Expr1, Expr2)`: the implication with `Expr1` being the premise and `Expr2` being the consequence, which are themselves expressions.
 - `iff(Expr1, Expr2)`: the biconditional of `Expr1` and `Expr2`, which are themselves expressions.
 - All terms (i.e., without opening parentheses after them) is a propositional symbol.

The `CNFList` should be an unbound variable. After execution of `normalize/2`, it will be binded to a list of clauses, each clause is a disjunction represented by a list of positive

or negated propositional symbols, with negation represented by `neg()`. E.g., if we execute `normalize(if(or(neg(a), b), c), CNF)` with `CNF` being unbound, then `CNF` will be bounded to (a particular ordering of) `[[a, c], [neg(b), c]]`. You should remove duplicated clauses or duplicated literals in clauses, although you need not check whether two elements within a clause is exactly opposite to each other (and thus is not useful).

3. **Wumpus World (50%)** In the web page, you can find a program `wumpus.pl`, which implements a cave which an agent is asked to navigate around. You must write an “agent” program to navigate around the cave. The core of the `wumpus` program is in the `do_play/3` predicate. It calls the `a_init/0` procedure of the agent program at the beginning, and calls `a_tell_percepts/1` of the agent to tell the agent what percepts are experienced: the argument is a list of 5 numbers, each being a 0 or 1. They tell whether the breeze, stench, glitter, bump and scream percepts has been sensed. It then calls `a_ask_action/1` to ask the agent for an action, which should bind its variable argument to one of `left`, `right`, `forward`, `grab`, `shoot`, and `climb`. The action is then performed, and the above is repeated until the agent is either killed or get off the cave. The agent should assume that it starts at `room(1,1)` (which is how we write a room: the first is row number, the second is column number), facing East. No other information is provided by the `wumpus` program, so the agent program must derive its own information using the received percepts and the actions it produces.

In the web page, you can find a sample agent program, `random_agent`, which chooses actions randomly. Modify the agent program so that it tries to find the gold, with the following strategy:

- If the gold has been found, the agent should fetch it, return to `room(1,1)` in the fewest number of steps, and run use the `climb` action to exit the cave.
- If there is at least one known safe room that the agent has not yet visited, the agent should visit the one in the fewest number of steps.
- If there is no safe room that the agent has not yet visited, and the agent knows the location of the wumpus, the agent should go to a room neighbouring to the wumpus, turn towards the wumpus room and shoot in the fewest number of steps, attempting to make more rooms safe.
- If all else fails, the agent should return to `room(1,1)` in the fewest number of steps and use the `climb` action to exit the cave.

To automate the process to tell the wumpus about the percepts, the `wumpus` program has a representation. Obviously the agent program is not allowed to take advantage of these information. In general, the `wumpus` program uses predicates which starts with `w_`. When testing your program, the TA will use a modified `wumpus` program which have the names of all these predicates changed. So **your program must not call or create predicates with names started by `w_`**, to avoid conflicting with the changed program of the TA. The `wumpus` program contains a predicate, `w_test`, which can be used to test your program. The TA will of course choose some other configurations to test your agent as well.