CMPT 310 - Artificial Intelligence Survey

Assignment 3

Due date: Nov 21, 2016 — J.P. Delgrande
10 marks — Nov 2, 2016

Important Note: Students must work individually on this, and other CMPT 310, assignments. You may not discuss the specific questions in this assignment, nor their solutions with any other student. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the general concepts involved in the questions in the context of completely different problems. If you are in doubt as to what constitutes acceptable discussion, please ask!

Assume that a knowledge base KB consists of a set of rules of the form $a_1 \wedge \cdots \wedge a_n \Rightarrow a$ where $n \geq 0$, a is an atom, and each a_i is an atom. For a rule $r = a_1 \wedge \cdots \wedge a_n \Rightarrow a$, $body(r) = \{a_1, \ldots, a_n\}$ and $head(r) = a$, If $i = 0$ then the rule is a *fact* and is just written as a. Here is an example of a set of rules:

Example 1:

p
$p \Rightarrow q$
d
$q \wedge j \Rightarrow g$
$f \Rightarrow e$
$f \wedge d \Rightarrow c$
$d \wedge g \Rightarrow c$
$e \wedge d \Rightarrow c$
$j \Rightarrow b$
$b \wedge c \Rightarrow a$
j

For this assignment you will implement, using some imperative langauge (C, Python, Java, etc.), a backward chaining algorithm for such rules. Here is pseudo code to give an outline of what is required for the algorithm.

Backward Chaining Procedure: The argument to *solve* is a list of goals that have yet to be solved. If the user's query is q, then *solve* will initially be called with argument (q).

```
solve(goals):
    if goals = () then return true
    let a = first(goals); goals = rest(goals)
    for each r ∈ rules where head(r) = a
        if solve(append(body(r), goals)) = true
            return(true)
    return(fail)
```

Notes:

1. Your program should read in a set of rules (possibly at the command line, in firing up your program). It should then process a set of queries as entered by the user.

2. Your program should accept the following syntax for a rules file. Each line will contain one rule. A rule $a_1 \wedge \cdots \wedge a_n \Rightarrow a$ will be written as $a \;\; a_1 \;\; \ldots \;\; a_n$. So the head of the rule comes first, followed by the body.

   Example 2:

The rules:	p $p \Rightarrow q$ $p \wedge q \Rightarrow r$ $s \Rightarrow r$	would be expressed:	p $q\ p$ $r\ p\ q$ $r\ s$

3. You must run your program on the two examples, as given, with query a in the first case and r in the second. In addition if necessary, submit further test cases to provide further evidence that your program is working as expected.

4. The pseudo-code only determines whether a proof exists or not. Your program should output helpful, readable, diagnostics, indicating how an atom was derived and when a chain of reasoning fails or succeeds.

5. Documentation should briefy describe what you have done, as well as discuss features and limitations of your program. For example you might want to consider how your program would behave if it had the rule $p \wedge q \Rightarrow p$ or the pair of rules, $q \Rightarrow p$ and $p \Rightarrow q$. (Note: You are not expected to necessarily handle cases such as these, but you are expected to know about them and to document them.)