Assignment 1

Duc Minh Le

ID: 3561723

Athabasca University

COMP 456 – Artificial Intelligence

Dr. Sidi Esmahi

Steve Leung

# Question 1

There are several limitations of expert systems in modeling human performance:

1. Difficulty in capturing “deep” knowledge, there are potential narrowness of knowledge in expert systems. E.g., MYCIN lacks real knowledge about human physiology.
2. Lack of robustness and flexibility. Expert systems generally lack the ability to further examine, coming up with strategies when presented with a problem that can’t be solved immediately.
3. Inability to provide deep explanation. Because expert systems lack deep knowledge of their problem domains, their explanations are generally restricted to a description of the steps they took in finding a
4. Difficulties in verification. Though the correctness of any large computer system is difficult to prove, expert systems are particularly difficult to verify
5. Little learning from experience. A completed expert usually will not improve without further update and maintenance from its programmers.

# Question 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | Q | ¬P | ¬Q | ¬P → ¬Q | P ∨ ¬Q |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |

From the truth table we can see that there is an equivalence between (¬P → ¬Q) and (P ∨ ¬Q)

# Question 3

The 2 expressions written in list syntax:

* lives\_in(X athabasca\_town works\_in(X athabasca\_university))
* lives\_in(karen Y works\_in(X Z))

Unification:

1. unify( lives\_in(X athabasca\_town works\_in(X athabasca\_university)), lives\_in(karen Y works\_in(X Z)) ), recursively unify the elements of each expression.
2. unify( X karen ), success, returns the substitution {karen/X}
3. unify( lives\_in(athabasca\_town works\_in(karen athabasca\_university)), lives\_in(Y works\_in(karen Z)) ), recursively unify the elements of each expression.
4. unify( athabasca\_town Y ), success, returns the substitution {athabasca\_town /Y}
5. unify( lives\_in(works\_in(karen athabasca\_university)), lives\_in(works\_in(karen Z)) )
6. unify( works\_in(karen athabasca\_university), works\_in(karen Z) ), recursively unify the elements of each expression
7. unify( karen, karen ), success, returns the empty substitution {}
8. unify( athabasca\_university, Z ), success, returns the substitution { athabasca\_university/Z }
9. unify( works\_in(), works\_in() ), success, returns the empty substitution {}
10. unify( lives\_in(), lives\_in() ), success, returns the empty substitution {}
11. unify( (), () ), success, returns the empty substitution {}

Thus the 2 expressions can be unified, and the most general unifier would be:

lives\_in( X, Y, works\_in( X, Z ) )

# Question 4

Written in **a1q4.pl**

# Question 5

Written in **a1q5.pl**