Mandatory Exercise - Backwards-chaining

Anders Wind - awis@itu.dk

Task 1

Backwards-chaining is goal driven and can have a better time complexity than forward-chaining. This is due to the fact that Backwards chaining only "searches" the relevant part of the knowledge base, but forward-chaining has to search all different implications of the given facts. Therefore situations where we have a lot of facts that do not have any impact on the query, backwards chaining will be advantageous.

Task 2

I assume that the KB is in CNF.

I note that only one of the symbols in a clause's premise needs to be false for us to say that the clause is false, but only one of the clauses for a propositional symbol needs to be infered for the propositional symbol to be entailed.

```
function PL-BC-Entails? (KB, q) returns true or false
    inputs: KB, the knowledge base, a set of propositional definite clauses
            q, the query, a propositional symbol

    Value-Map ← Check-Clause(KB, q, Map.Empty)
    if q is in Value-Map
        return Value-Map[q]

    return false

function Check-Clause(KB, q, Value-Map) returns map
    inputs: KB, the knowledge base, a set of propositional definite clauses
            q, the query, a propositional symbol
            Value-Map, a map of propositional symbol keys mapping to boolean values, empty to begin with

    if q is in Value-Map
        return Value-Map

    Value-Map.add(q, false)
    for each clause c in KB which implies q
        if c is fact
            Value-Map[q] ← true
            return Value-Map

        else
            result ← true

            Value-Map ← Update-Map(KB, c.PREMISE, Value-Map)
            for each (key, value) in Value-Map where key is in c.PREMISE
                if !value
                    result ← false
                    break

            if result
                Value-Map[q] ← true
                return Value-Map

    return Value-Map


function Update-Map(KB, symbols, Value-Map) returns map
    inputs: KB, the knowledge base, a set of propositional definite clauses
            symbols, a set of propositional symbols
            Value-Map, a map of propositional symbol keys mapping to boolean value

    for each Propositional-Symbol ps in symbols
        Value-Map ← Check-Clause(KB, ps, Value-Map)

    return Value-Map
```

Task 3

Yes but it returns false due to a cyclic case. My algorithm handles cyclic cases by checking if a Symbol has already been already exists in the Value-Map.

This is achieved by adding (q, false) to the Value-Map before recoursively checking the Premise of a clause. Then if q is reached through again in one of the premise symbols(or through any number of recoursions) it will just return the value-map(where the value of the symbol is false) and not try to update again.

Task 4

I assume that it is meant, that time complexity the algorithm runs in based on the amount of propositional symbols. The algorithm runs in linear time since every symbol will only be visited once due to the generation of the value-map and if a symbol already has been entailed it will not be entailed again.