Sonal Unadkat CMPT 310: Assignment 3 Report Fall 2016

For this assignment, we were asked to implement a form of Logical Reasoning using a Knowledge- Based Agent (KB). Using Backward Chaining, the goal was to be able to ask a query, and to conclude whether or not the query entails were based on the information given to the KB. To implement this program, we must be able to not only "Ask" the KB the query but we must be able to give the KB some prior knowledge to be able to solve the queries.

We gave the KB information by reading a file with the queries written in a specific format. If the format was correct, we would know the not only the query could be "Ask" but we can also "Tell" the remaining queries needed to the KB. Each query is stored as a collection of atoms. Each atom contains a truth or false value, and it is assumed to be in the form of a Horn Clause. If there is only one atom in the query it is considered as true and is therefore a fact. If there are other atoms in the query it will store them as the body of the query. After we have given the KB all the information it needs, we can then "Ask" the KB to see if our query entails.

When we "Ask" the KB a query, we initialize the backward chaining process. This is done by using recursion and 2 important queues. The first queue we create is an Entailed Queue, which will keep track of all the atoms that have already been proven to be true. An atom is true, if and only if it is a fact or it consist of atoms which are proved true from being derived of facts. We use this Queue to keep track of all the atoms that we essentially know are true. The Check queue are all the atoms in the body of our query that have proven to be true. These two lists are passed through each phase of the recursion.

How the algorithm works, is by taking our query and iterating through all the clauses in the KB that relate to our goal query. Once we find a related query, we can analyze it by trying to solve each of the atoms in that related query's body. If our query that we are analyzing, is a fact, it can be added onto our entailed list directly. If not, we will then recursively solve for each atom in the body of the query. Every time we reach a fact we will add it into the entailed list and return true. Every time we solve for an atom in the body of the query, we will add it into our check list. After analyzing all atoms, if the size of our Check queue is equal to the total number of atoms of our body of the query we are analyzing, then we can conclude that, that query is entailed. What this translates to, is that we can solve for every atom in the body of our query, and therefore we can conclude that this query can be added into our entailed list and return true. If we do not get an equal number then, we can return false, and continue on finding the next query that pertains to our goal query in the KB for analysis. If we reach the end of all the queries in our KB, we can conclude that our goal cannot be entailed and therefore we can return false back to the user program.

When implementing backward chaining, it is important to remember that Horn clauses that have multiple atoms in their body can be broken down into smaller Horn Clauses.

For example (from Assignment Guidelines) $p \wedge q \Rightarrow p$ can be turned into $q \Rightarrow p$ and $p \Rightarrow q$.

In my implementation of an algorithm, we solve each clause by breaking it into smaller atoms. This is done while we recursively solve for each atom of the body of the query. The reason why our KB will not just store them as smaller Horn Clauses, is because our KB needs the original query to related back to. If

we recall back to our Check queue, which keeps track that all the atoms that consisted in the body of our query that are entailed, we can compare it back to the original query and derive a conclusion.