AI Project 3: Simple Automated Reasoning System

Roy Howie

April 5, 2017

1 Task

Create an automated reasoning system which resolves queries using either modus ponens or the resolution principle.

2 Algorithm

```
1: procedure ResolveQuery(q)
2:     L ← q                                              ▷ L is a list
3:     while length(L) > 0 do
4:         p ← L[0]
5:         if p = T → F then return success
6:         p_1, ··· , p_n ← resolve p with other queries in L or in the database
7:         Append p_i to L for every i
8:     return failure                                     ▷ L is empty, so the query could not be proven
```

3 Implementation

3.1 Setup

The automated reasoning system was written in `JavaScript`. The demo was written using the `Meteor` framework on the `node.js` runtime. The source code can be found at `http://github.com/royhowie/csc545/` under the `project3/app` directory.

3.2 Rules & Facts

The sample knowledge database consisted of the following rules and facts. Note that the system is capable of reasoning about any set of coherent rules and facts; the demo app provides an interface for changing the rules and facts in the database.

3.2.1 Rules

- $\star$ $lawyer(p) \rightarrow rich(p)$
- $\star$ $rich(p) \rightarrow old(p)$
- $\star$ $poor(p) \wedge grumpy(p) \rightarrow curmudgeon(p)$
- $\star$ $rich(p) \wedge male(p) \rightarrow grumpy(p)$
- $\star$ $wise(p) \wedge old(p) \rightarrow sage(p)$

- $grumpy(p) \land old(p) \rightarrow curmudgeon(p)$

3.2.2 Facts

- $T \rightarrow lawyer(John)$
- $T \rightarrow male(John)$
- $T \rightarrow wise(John)$

4 Demo

A running version of the app can be found at `http://162.243.111.226:4044` or at `royhowie.com:4044`.

5 Conclusion

This automated reasoning system used a combination of modus ponens and the resolution principle to attempt to solve theorems. However, it was limited to a database in normal form. That is, rules could only be of the form $A_1(p) \land \cdots \land A_2(p) \rightarrow B(p)$ and queries could only be of the form $predicate(arg_1, \cdots, arg_n)$.

While an interesting project, life is much more broad. Most rules of life involve disjunction, conjunction, negation, and association. It would be interesting to create an automated reasoning program which at least accepts a ruleset in normal form as opposed to only Horn form.

Furthermore, the logic surrounding binding and unification was not perfect and ultimately had to be removed from the demo. This allowed for queries such as $lawyer(Mary)$ to be returned as T when only $T \rightarrow lawyer(John)$ was in the database. This is clearly nonsense and should be fixed.

The program originally implemented unification properly, except it would occasionally cause the program to become "lost" when the branching factor was high. This was due to a bug with how the bindings were being stored in memory.