Jaden Dawdy

Professor Andersen

Intro to AI, CS457

December 13, 2024

Automated Theorem Prover

The goal of this project is to create a program that efficiently resolves theorems given a knowledge base and a refute query. This project went pretty well and went much less stressful than any of the other projects. This was because I started a week before the due date and worked a few hours each day to make gradual progress leaving me with minimal work left to do when the due date came.

The main struggle of this project was getting everything to work together. I was having several issues with getting RandomResolve and HeuristicResolve to properly work together. The goal was to get the output to be as similar to the example output given as I could and I got very close, but there were just so many issues along the way that really made this project take long. To get the heuristic working as desired it didn't take horribly long. The example given in the assignment description gave me good ideas and helped form the shape of the heuristic resolve that is there now.

The heuristic used involves several steps. First is prioritizing refuted sentences. This is good because it makes sure the steps actually move toward the goal of finding the contradiction over going off in random directions. Second is finding the shortest complementary sentence that can be used. This is important because using longer sentences than necessary can lead to taking more steps than needed. In addition it checks for complimentary predicates to prevent wasting time unnecessarily. Lastly, it

keeps an array of previously used sentences to ensure it doesn't redundantly loop. This was an issue earlier in the process where it would get stuck looping on the seemingly best sentence.

In addition to HeuristicResolve, the Unify and canUnifyParams functions were implemented. canUnifyParams is a helper function that quickly checks if two parameters can be unified. This saves time over Unify which helps save time. The Unify function takes in two sentences and finds predicates in each sentence that have opposite negations. Such as P(x) in sentence 1 and !P(x) in sentence 2. It then creates substitutions for the variables and replaces the variables with the substitutions throughout the sentences. It then unifies the parameters and adds the new sentence to the knowledge base.

Lastly, the Resolve algorithm runs the RandomResolve and HeuristicResolve and prints their ratios similar to the example code given. The only extra spet of effort put into this function was creating separate copies of the knowledge base for the two resolve functions because there was an issue where HeurisiticResolve would reuse sentences generated from RandomResolve.

The only thing left that I would like to improve on in the future is just adding more steps to the heuristic or finding a better one. I also added a limit to the amount of sentences that can be stored because I noticed there was a limit so when a resolution is too long, there is a bus error. I noticed in the example code it would just loop infinitely for the RandomResolve so I wasn't sure what the right direction to go was. So I just implemented it in the way that it seemed to be intended where there was a limit to the number of sentences that can be stored and just let the bus error remain.

Results:

This is what outputs look like for different example files that were given.

jadendawdy@Mac prover % ./prover easy1

```
Current Knowledge Base
0: A(b)
1: !A(Tim)   :from refuted part


Running Random Resolve (press Ctrl-c to cancel)...
   0:                    A(b)
   1:                    !A(Tim)
--
Sentences 0 and 1 Complete the Proof!
RandomResolve: #sent-generated = 1, #steps = 1, time = 3.8e-05


Running Heuristic Resolve...
   1:                    !A(Tim)
   0:                    A(b)
--
Sentences 1 and 0 Complete the Proof!
HeuristicResolve: #sent-generated = 1, #steps = 1, time = 3.3e-05
Heuristic vs Random ratios:  hSteps/rSteps = 1, hTime/rTime = 0.868421
```

Next is for the first example knowledge base from the assignment on canvas:

a) **Every ambassador speaks only to diplomats, and some ambassador speaks to someone, therefore there is a diplomat.**

jadendawdy@Mac prover % ./prover example1.txt

```
Current Knowledge Base
0: !Ambassador(b) !SpeaksTo(b,c) Diplomat(c)
1: Ambassador(John)
2: SpeaksTo(John,Bob)
3: !Diplomat(Bob)   :from refuted part


Running Random Resolve (press Ctrl-c to cancel)...
   1:                    Ambassador(John)
   0:                    !Ambassador(b) !SpeaksTo(b,c) Diplomat(c)
--
   0:                    !Ambassador(b) !SpeaksTo(b,c) Diplomat(c)
   2:                    SpeaksTo(John,Bob)
--
   4:                    !SpeaksTo(b,c) Diplomat(c)
   3:                    !Diplomat(Bob)
--
   3:                    !Diplomat(Bob)
   0:                    !Ambassador(b) !SpeaksTo(b,c) Diplomat(c)
--
```

```
   2:                SpeaksTo(John,Bob)
   4:                !SpeaksTo(b,c) Diplomat(c)
--
   7:                !Ambassador(b) !SpeaksTo(b,c)
   2:                SpeaksTo(John,Bob)
--
   1:                Ambassador(John)
   9:                !Ambassador(b)
--
Sentences 1 and 9 Complete the Proof!
RandomResolve: #sent-generated = 7, #steps = 7, time = 3.5e-05


Running Heuristic Resolve...
   3:                !Diplomat(Bob)
   0:                !Ambassador(b) !SpeaksTo(b,c) Diplomat(c)
--
   4:                !Ambassador(b) !SpeaksTo(b,c)
   1:                Ambassador(John)
--
   5:                !SpeaksTo(b,c)
   2:                SpeaksTo(John,Bob)
--
Sentences 5 and 2 Complete the Proof!
HeuristicResolve: #sent-generated = 3, #steps = 3, time = 2.3e-05

Heuristic vs Random ratios:  hSteps/rSteps = 0.428571, hTime/rTime = 0.657143
```

Output from second example given on canvas assignment:

b) **Every computer science student works harder than somebody, and everyone who works harder than someone else gets less sleep than that person. Maria is a computer science student. Therefore Maria gets less sleep than someone else.**

```
jadendawdy@Mac prover % ./prover example2.txt

Current Knowledge Base
0: !CS(b) WorksHarder(b,c)
1: !WorksHarder(d,e) LessSleep(d,e)
2: CS(Maria)
3: !LessSleep(Maria,f)   :from refuted part


Running Random Resolve (press Ctrl-c to cancel)...
   1:                !WorksHarder(d,e) LessSleep(d,e)
   3:                !LessSleep(Maria,f)
--
   2:                CS(Maria)
   0:                !CS(b) WorksHarder(b,c)
--
   0:                !CS(b) WorksHarder(b,c)
   4:                !WorksHarder(d,e)
--
```

```
   4:                   !WorksHarder(d,e)
   5:                   WorksHarder(b,c)
--
Sentences 4 and 5 Complete the Proof!
RandomResolve: #sent-generated = 4, #steps = 4, time = 6.1e-05


Running Heuristic Resolve...
   3:                   !LessSleep(Maria,f)
   1:                   !WorksHarder(d,e) LessSleep(d,e)
--
   4:                   !WorksHarder(d,e)
   0:                   !CS(b) WorksHarder(b,c)
--
   5:                   !CS(b)
   2:                   CS(Maria)
--
Sentences 5 and 2 Complete the Proof!
HeuristicResolve: #sent-generated = 3, #steps = 3, time = 5.5e-05

Heuristic vs Random ratios:  hSteps/rSteps = 0.75, hTime/rTime = 0.901639
```