Mary Catherine Scott
P3 Write Up

General Discussion

I would like to start be giving a general description and discussion on how I felt this project went, where it is now, and what I feel I could still improve on. This was a difficult project for me to get started on. I spent a lot of time reading the code and getting familiar with it, but had a difficult time really getting going on it. When we started going over the code in class I was really excited to be able to see what was needing to happen to really make it work. When we did not finish the code before Thanksgiving Break I began to worry. I put it off for most of the break as the class was told more code would be sent out to us, this was a mistake on my part. I did not work on it as I should over the break. When we returned and told no code would be coming I panicked a bit, but got to work. It was a slow process at first, but I was able to get re-write what was given in Java and get it to work as expected. Or at least mostly.

At this point the project does the task it was asked to do. It solves a given problem first using a random resolver and then using a heuristic resolver, which will be discussed later in the write-up. I do not believe that it is perfect by any means, but it does appear to work at least most of the time. I was able to run all the given test files through and receive the expected contradictions.

As for what could be improved, testing. I do not feel I put in the time or the effort to truly test this code to make sure it is working exactly the way it should. I believe it works because it gives me answers to problems. But there has not been near enough testing to really be sure about that. Also, there is no kill button. If a problem is unsolvable or just taking a really long time, there is nothing that will stop it, it will run until it hits some kind of exception.

Heuristics

After spending far too much time trying to get the thing to work at all, there was not much time for playing with heuristics, but I did spend at some time. The heuristic in my project determine the priority that a pair of sentences has on the queue. I started with an easily implemented heuristic, taking the absolute value of the difference in the number or predicates in the two sentences. After some thought, this was not a very good heuristic, but it did help me to see that it was working correctly. It took many more steps than the random solver, pretty much every time.

The next heuristic I tried still looked at the number of predicates, but this time if they were the same size it returned the size, if they were different sizes it returned the larger number of predicates. My hopes here is really just a pass through to get to the smaller the number of predicates the higher priority I want them to have. This was definitely and improvement upon the random search. It didn't always have few steps, but it did about 6 out of every 10 runs. As successful as this was in improving things, after trying a few other heuristics on just looking at size this turned out to not be the best approach. There is just so much more involved here than just size of the sentences. So, I started into looking at what the sentences actually contained.

I added some looking up of the actual predicates and found that, I don't think I fully understand how my heuristics should be working. At one time for fun, I switched how the compare function worked and it made things better at least for a moment. As I try this with several different test files, I find that for simpler test files the heuristic resolver is better, but for more difficult files, it's not.

I have run out of time to really try to make this better. If I get the chance I will work on it some more to try to improve my grade, but at this point I am turning it in.

Problems I know about

I do know that sometimes my code will not solve all the test cases, but it has solved them all at least once, now if only I could get it to consistently do it.

There also seems to be an issue in the printing on the solution for the heuristic solver… though I am not sure why.

How To Run

In the directory just about the source files where you can see the folder theorem prover run:

```
$ javac theoremprover/*.java
```

this will compile the code. Put all test files in this location not in with the source code. then run:

```
$ java theoremprover/TheoremProver test.txt
```

this will use the file test.txt and solve it.