Lab Report A4A
CS4300
Derek Heldt-Werle 1,3,5
Matthew Lemon 2,4,6
9/29/16

1. Introduction

For this lab we will be implementing a standalone function that uses the resolution inference rule of propositional logic for resolution theorem proving. We intend to answer the following questions:

- Does there exist a relation between the number of clauses in the KB to the number of sentences?
- Does there exist a relation between the number of clauses in the KB to the number of iterations?
- Does there exist a relation between the number of clauses in the KB to time taken?

2. Method

The Resolution Theorem Prover
First the sentences are broken down into CNF pairs, e.g. if a sentence is [1, 2, 3] it is broken down into [1, 2], [1, 3], [2, 3] and added to our CNF struct which we are doing our computations on. We then add the inverse thm clauses into our CNF struct as well. After the CNF struct is built we then loop until looping no longer produces any new resolvents or we produce the empty set. We use a function called CS4300_Resolution to determine if a) two clauses produces a resolvent and b) what the resolvent is. If there is a resolvent produced and it is the empty set we set Sip equal to the pairs we have produced and return. Otherwise, if there is a resolvent produced and it is not the empty set, we add it to our clauses and continue searching.

3. Verification of Program

Trivial Case:
DP(1).clauses = [1]
DP(2).clauses = [-1]
thm = [2]
vars = [1,2]
sip:

Step	Formula	Derivation
1	1	Given
2	-1	Given

3	-2	Negated Solution
4	*empty set	1 resolve 2

Results of running sip = CS4300_RTP(DP, thm, [1,2])

```
1
-1
-2
```

```
 DP(1).clauses = [1,2]
 DP(2).clauses = [-1,3]
 DP(3).clauses = [-2, 3]
 thm = [3]
 vars = [1,2,3]
sip:
```

Step	Formula	Derivation
1	[1,2]	Given
2	[-1,3]	Given
3	[-2,3]	Given
4	-3	Negated Solution
5	[2,3]	1 resolve 2
6	[1,3]	1 resolve 3
7	-1	2 resolve 4
8	3	2 resolve 6
9	-2	3 resolve 4
10	2	4 resolve 5
11	1	4 resolve 6
12	*empty set	4 resolve 8

Results of running sip = CS4300_RTP(DP, thm, [1,2,3])

```
[1,2]
[-1,3]
```

```
[-2,3]
-3
[2,3]
[1,3]
-1
3
-2
2
1
```

Case 3: No solution (Will show state of temp prior to return as no more resolutions can be made to the KB)

```
 DP(1).clauses = [1,2]
 DP(2).clauses = [-1,3]
 DP(3).clauses = [-2, 3]
 thm = [-3]
 vars = [1,2,3]
```

temp:

Step	Formula	Derivation
1	[1,2]	Given
2	[-1,3]	Given
3	[-2,3]	Given
4	3	Negated Solution
5	[2,3]	1 resolve 2
6	[1,3]	1 resolve 3

Sip :

```
[]
```

Results of running sip = CS4300_RTP(DP, thm, [1,2,3])

```
[ ]
```

4. Data and Analysis

N/A

5. Interpretation

Through both testing, and gaining sufficient understanding of how resolution theorem proving works, it became clear that the three questions we posed are all related. If there is a solution, the number of sentences produced, iterations, and time taken can vary significantly as the order and values of the clauses can lead to the production of an empty set in the first or final iteration. However, if there is not a solution, and since we are using the brute force pairwise method of resolution, it will go through and compare every sentence to one another every single time a new sentence is added. This leads to a complexity of $\Theta(n+k)^2$ where k is the number of resolvents since every time a new resolvent is added to our pairs, it must then be compared with every member of the sentences.

6. Critique

Resolution Theorem Proving has been a great refresher on boolean algebra and truth tables. In our implementation of the RTP we ended up breaking all of the clauses down into pairs (stated in Method) but I believe we could have left the clauses as is and still produced a solution. During lecture we were shown how RTP can be used on clauses of 3 or more elements and if given the time or necessity we could easily convert our solution to one that does just this.

7. Log

Author Matthew Lemon & Derek Heldt-Werle
Coding Portion (Worked together): 6
Report (Derek): 2
Report (Matt): 2