Raj Patel – [rpp25@pitt.edu](mailto:rpp25@pitt.edu)

1. **Forward Chaining**

This program includes Rule and Atom objects. The Rule objects contain parameters for LHS, a list of atoms, and RHS, an atom that is a fact. The Atom objects contain a name and argument list parameters. The Rule object also stores its’ unifications in a dict() called ‘subs’. The setup() function reads in the text file and through the use of regular expression parsing, the Rule and Atom objects (as facts) are created.

This algorithm will accept the list of rules, the list of facts, and the goal, as arguments, all of which are determined in the setup() function. The algorithm then iterates through all rules in a FIFO queue. The first rule is popped, and all facts are individually applied to the rule if the fact unifies with a certain atom in the rule. If an atom unifies, a new rule is created with only ONE atom unified and substituted with a fact, and it is added to the end of the queue. If all atoms on the rule’s LHS are unified and substituted, then a new fact is derived and added to the list of facts.

If nothing is unified and substituted, the rule gets places to the end of the queue. The algorithm continues until either the goal is reached, or if no rule in the queue can be unified anymore, returns False.

This algorithm produces the correct inferences and correct results, though in a fairly inefficient manner. Testcases 1, 3, 4, and 8 are all provable, while the rest return False, or unprovable.

1. **Incremental Forward Chaining**

My goal for this algorithm was for each fact, to recursively apply the fact to every atom in the rule. Currently the algorithm runs through and applies all facts, and deriving new ones, but only for one iteration. While this works for one test cases, it obviously doesn’t for the others. This is especially challenging for the relationships test cases because there are several duplicates and corner cases to account for.