Manipulator

+drop(name : string)
+dumpRules(&os : ostream &os)
+dumpFacts(&os : ostream)
+dump()
+load()
+Manipulator()
+~Manipulator()
+inference()

Rule

-paramVector : vector<vector<string>>

+Rule(log : string)
+Rquery(rmap : map<string, Rule*>, s : string)
+check(rmap : map<string, Rule*>, fmap : map<string, Fact*>)
+get_logop()
+getAttribute()
+setAttribute(attribute) : void
+~Rule()

Fact

-title : string
-vstring : vector<string>

+Fact(name : string)
+get_vstring()
+query()
+~Fact()

SRI client

1: LOAD

1.1: M->load()

alt

1.1.1: parses file if(rule) creates Rule object and stores in map

1.1.2: -or- creates Fact object and stores in map

2: DUMP

2.1: M->dump()

3:

2.1.1:

4: dumped in .sri file client can access

2.1.2:

5: DROP

5.1: M->drop()

5.1.1:

5.1.2:

6: INFERENCE

6.1: M->inference()

6.1.2:

6.1.1:

6.1.2.1:

6.1.1.1:

7: FACT

7.1: M->load()

alt

7.1.1: check if fact, create Fact obj, store in KB(map)

8: RULE

8.1: M->load()

alt

8.1.1: check if rule, create Rule obj, store in RB(map)

9: EXIT

9.1: exit SRI

main(i.e. SRI)   Manipulator   Rule   Fact

In the sequence above, assuming the program has been initialized, the SRI client types the LOAD command, calling the load() function within the Manipulator class. It parses the command line or input file, checks if it qualifies as a fact or rule, creates the respective object and stores the data in the RB(rule map) or KB(fact map).

The DUMP command also calls a function in the Manipulator class and calls dumpRules() and dumpFacts() and prints the state of the KB and RB into an .sri file that the client can access.

DROP traverses the KB and RB and deletes the given input.

INFERENCE traverses KB and RB and prints the resulting query.

FACT calls the load function in Manipulator and stores a Fact object in KB

RULE calls the load function in Manipulator and stores a Rule object in RB

EXIT when called, ends the SRI session.