SRI DESIGN DOCUMENT

OUTLINE OF RULE:

Rules are comprised up of components, of which can be either Facts or other Rules. A Rule is valid if its required components are valid.

Name	Description
Components	A vector of string vectors representative of either other rules or facts. The evaluate() function inside of Query will utilize this to check validity of those facts or rules.
Name	String name identifier
ops	An integer (either 1 or 0) that represents AND or OR operator that will be used to evaluate the validity of the rule.
toString()	Returns string representation of the rule.
Rule(string s, int ops, vector<string> c)	Constructor (ops is a vector of ints that act as a switch for AND and OR operators to use)

OUTLINE OF RULE BASE:

Holds references to each Rule, along with methods to manage them. Is a part of “Working Memory.”

Name	Description
rules	Map of rule pointers. Contains all rules in working memory.
Add(Rule * r)	Adds rule to the rule base. No two rules may have the same name.
Remove(Rule * r)	Remove rule from the rule base. Returns removed rule if successful.
RB()	Initializes an empty rule base
toString()	Returns string representation of all rules into a string so it can be printed

OUTLINE OF KNOWLEDGE BASE:

Holds maps of maps of maps… of strings representing Facts, along with methods to manage them. Is a part of "Working Memory."

Name	**Description**
facts	A map of maps of maps of vectors of string vectors that represent combinations of actors (strings). Each combination is a "Fact."
Add(vector<string>)	Adds fact to the knowledge base. Adds a string to the map if it doesn't exist already exist, steps into the index and add string there if it does. Does not add if fact already exists (using evaluate function)
Remove(string)	Remove fact from the knowledge base. Uses map's clear() and remove() as necessary.
KB()	Initializes an empty knowledge base
toString()	returns string representation of all facts into a string so it can be printed

OUTLINE OF PARSE:

Holds maps of maps of maps… of strings representing Facts, along with methods to manage them. Is a part of "Working Memory."

Name	**Description**
RB* RuleBase	Pointer to the RuleBase
KB* KnowledgeBase	Pointer to KnowledgeBase
Query* QQ	Pointer to Query
vector<string> RuleVector	Buffer for rules
Parse(KB* knowledgeBase, RB* ruleBase, Query* QQ)	Constructor for Parse
map<string, int>	Converts string input to boolean operators (int)
map<string, vector<vector<string>>> Query Output	Stores output of query
int searchLength(int start, int end)	Function that subtracts the two inputs
void ParseRule(string input)	Controls the Parsing of a line when inserting a rule
Int numPreds(string input)	Calculates numRuns
Void ParseLine(string input)	Gets a line of input
Void ParseFile(string fileName)	Controls the Parsing of input from a file

Void ParseTerminalInput ()	Runs an infinite loop waiting for input that will break if a ‘q’ is entered as the only command
Void AddFact(vector<string>)	Tries to add a vector of strings to the KB
Void AddRule(bool logic)	Adds a rule to RB

OUTLINE OF QUERY

The “Thinking” function that drives inferences. “Binds” actors together into combinations, which are then passed into each rule in the RB. The rules recursively evaluate their validity and populate the KB as necessary. Functions are put in place to not recur into duplicate combinations.

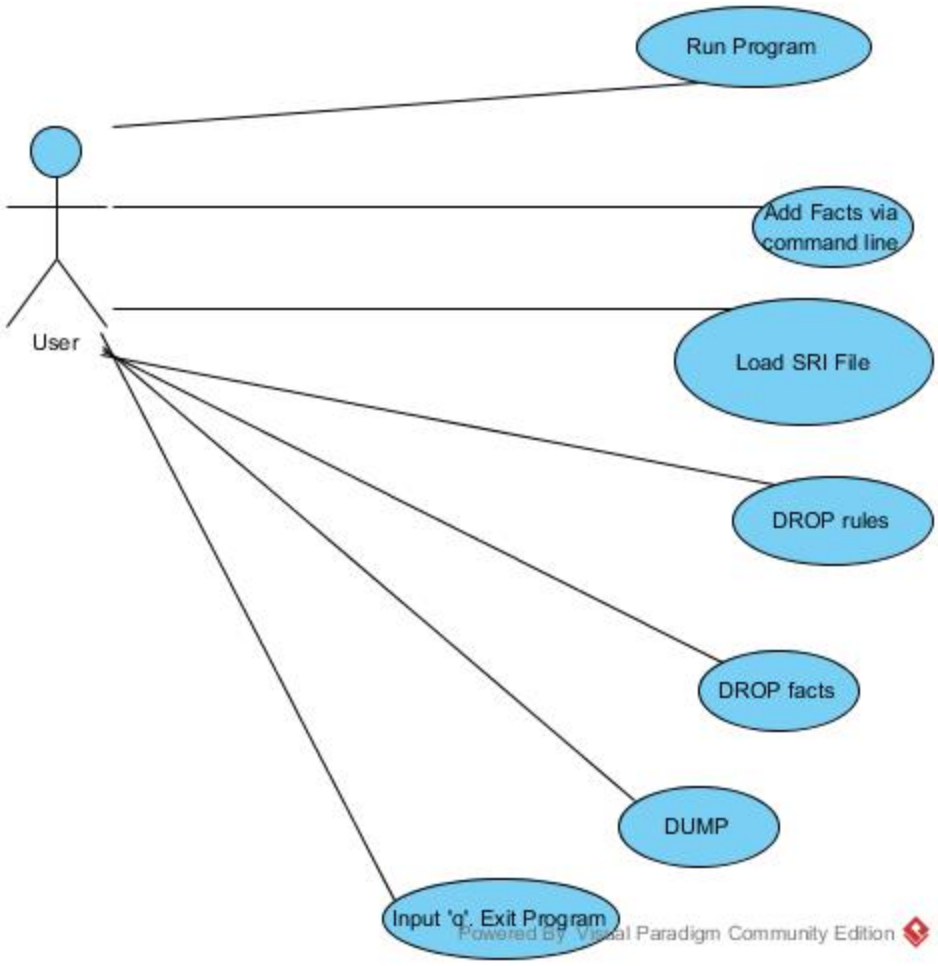
Name	**Description**
kb*	Pointer to a KB object
rb*	Pointer to a RB object
evaluateRule(Rule * r, vector<string> actors)	Calls either operatorOR or operatorAND
inference(vector<string> newFact)	“Binds” actors together into combinations, which are then passed into each rule in the RB. The rules recursively evaluate their validity and populate the KB as necessary.
operateOR(string name, Rule * r, vector<string actors>)	Evaluates OR type rule
operateAND(string name, Rule * r, vector<string actors>)	Evaluates AND type rule
Traverse	Creates a path through the KB (fact)
factEvaluate	Checks if fact exists in the KB
ruleEvalHelper	Determines if the rule’s component is a fact or a rule and evaluates accordingly

ASSUMPTIONS:

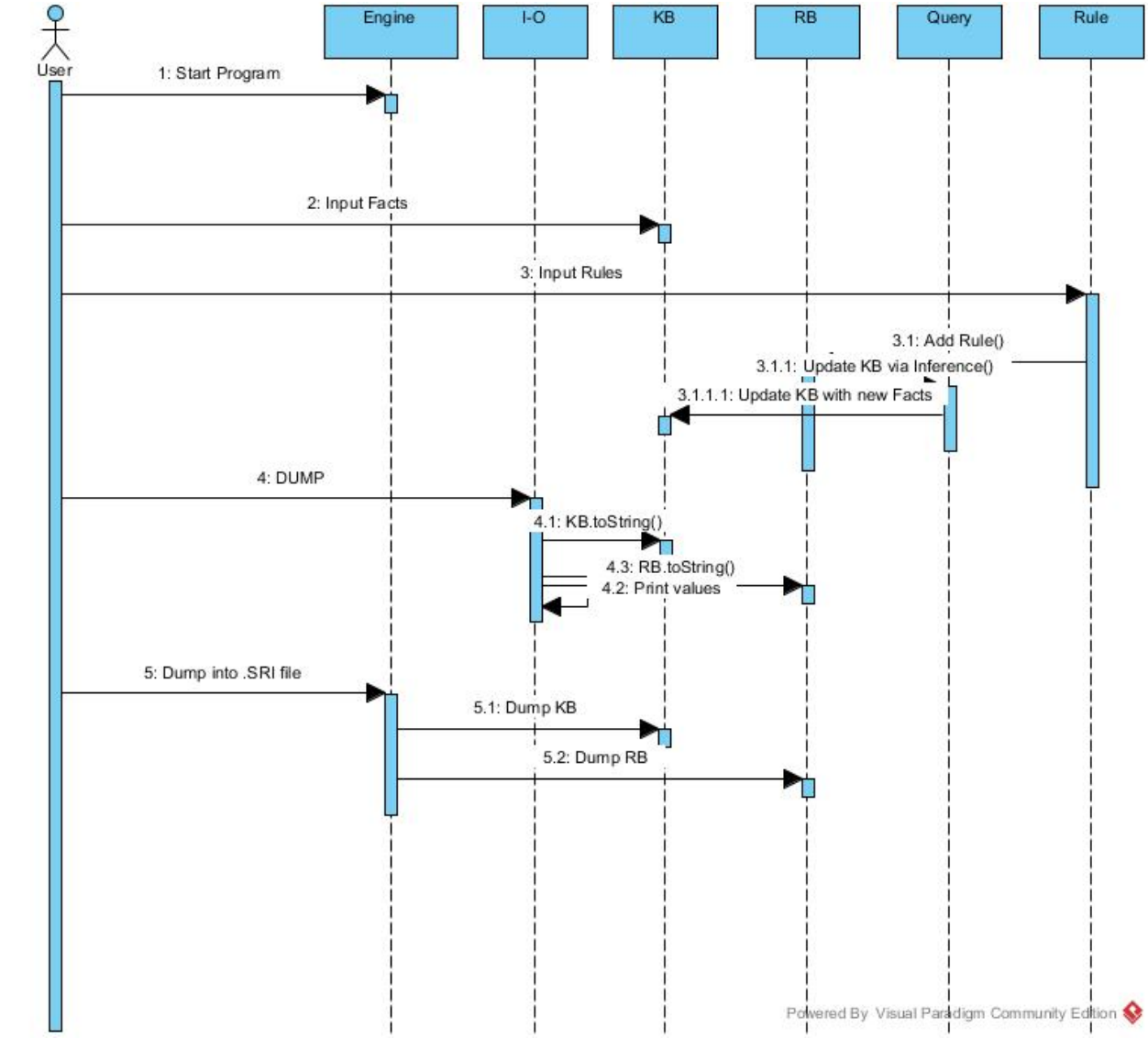
We are assuming the user is feeding the program proper input.
Rules are comprised of existing Facts and/or Rules already in working memory
A Fact is true if it can be found in the Knowledge Base
There can be no duplicate facts or rules

To compile:
Use: “make” without quotes in the root directory.

USE CASE DIAGRAM:



SEQUENCE DIAGRAM



CLASS DIAGRAM: