Natural Language Interface for Fridge

Introduction

In this project I have develop a natural language understanding system that can interface with a 'smart' refrigerator. Using this system, users can input text queries or instructions to their fridge. The queries can be either declarative or a question.

The modules in the program

The following modules have been used in the system:

- Lemma
- Lexicons
- Rules
- Parser
- Model
- Model checker
- Response

Implementation

Lemmas: -

We have used Lemmas in the system to identify the Parts-of-speech (POS) tagging for all the possible words that can occur in the query by the user to the system. We have identified the list of possible words and their respective POS tags. We have given multiple tags to some words to allow different composition of sentences. The lemmas are represented using facts. Only the uninflected words (or irregularly inflected) are stored as lemmas.

Lexicons: -

We have written lex rules to generate semantic representation for each of the tags. We created a lexicon for different types of verbs, prepositions, nouns, their phrases, relative clauses, auxiliaries, subject interrogative, complement interrogatives and different types of determiners. In addition to this, we have developed suffixes to handle the inflected words.

Rules: -

We defined predicates for all the grammar rules what would be used. This is used by the parser to parser sentence. We were able to build and handle the rules for interrogative and declarative sentences. These rules provide the system a way to combine terms to ultimately form a sentence.

Parser: -

We have used shift reduce parser for parsing the user input. The shift phase moves to the next word in the input stream.

A Reduce step applies a grammar rule to some of the recent parse trees, joining them together as one tree with a new root symbol.

Our parser takes the input in English language and creates a first order logic using

Model: -

We created a model which acts as a data set for our smart fridge. It contains domain D containing all entities of the fridge, including the fridge itself and its contents. The model also contains an interpretation function F which contains the relations between these entities

Model Checker: -

We created modelchecker to handle the different inputs possible – declarative queries which are of the form s(first_order_logic) and interrogative queries which are of the type ynq(first_order_logic) and q(first_order_logic). After validating and querying the model, it gets the appropriate answers and passes it to the response module which provides the output to the users.

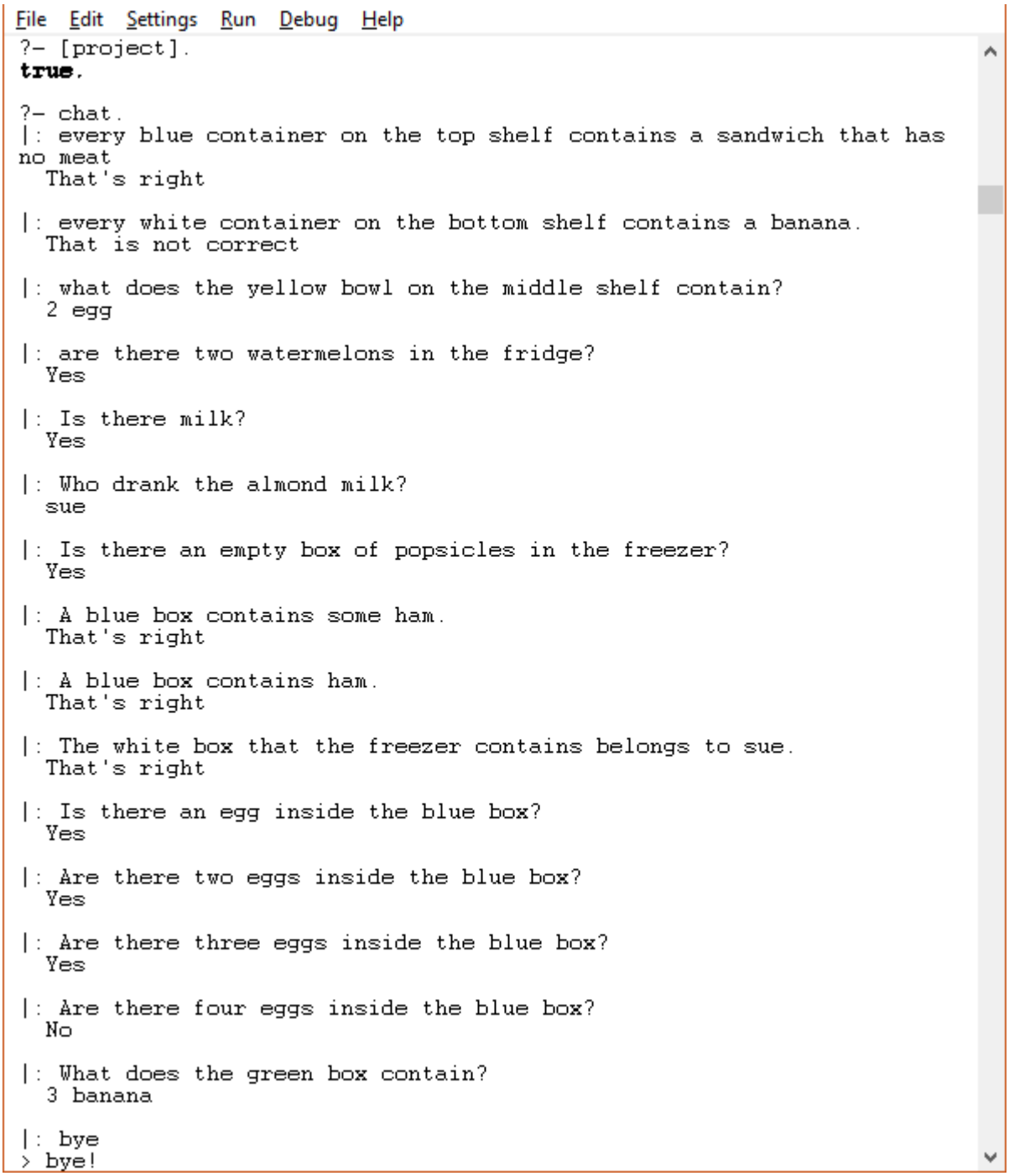
Response: -

For declarative sentences, the response module returns ‘That's right’ or 'That is not correct' based on the conditions in the model. For ynq i.e Yes No questions, the model returns corresponding answers. Finally, for wh questions, we return the answer to the question using our model.

Our modelchecker can run on any given model as long as it is represented correctly.

Execution examples

We could successfully run following examples of project document: -



```
File  Edit  Settings  Run  Debug  Help
?- [project].
true.

?- chat.
|: every blue container on the top shelf contains a sandwich that has
no meat
  That's right

|: every white container on the bottom shelf contains a banana.
  That is not correct

|: what does the yellow bowl on the middle shelf contain?
  2 egg

|: are there two watermelons in the fridge?
  Yes

|: Is there milk?
  Yes

|: Who drank the almond milk?
  sue

|: Is there an empty box of popsicles in the freezer?
  Yes

|: A blue box contains some ham.
  That's right

|: A blue box contains ham.
  That's right

|: The white box that the freezer contains belongs to sue.
  That's right

|: Is there an egg inside the blue box?
  Yes

|: Are there two eggs inside the blue box?
  Yes

|: Are there three eggs inside the blue box?
  Yes

|: Are there four eggs inside the blue box?
  No

|: What does the green box contain?
  3 banana

|: bye
> bye!
```

Below are five other examples that we choose and executed: -