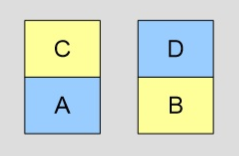
**Artificial Intelligence I**, *prof. Pasquale Caianiello 22.01.17*

Homework 5: Parsing and Interpreting Language in Blocks World.

*Presented by: Yuna Frolov*

In a given Block World,



The goal of the program is to read sentences that are given by the user and translate them into logical form, thus asserting new rules, queries and commands.

It should also understand natural language sentences; acknowledging facts, executing commands and answering to questions.

**\_\_init\_\_.py**

* To execute the program you need to run the “**Program.pl**” file.
* To exit the program type “quit”
* To know the current state of the world type “state”

**\_\_init\_\_.py**

I have worked with 6 Prolog files:

**Program.pl: (main)**

The role of the program is to connect the parts of the implementation, i.e. the Blocks World’s knowledge base, the tokenizer, the lexicon, the parser and the interpreter. The program also handles the input from the user.

**BlocksWorld.pl:**

The implemented Blocks World contains a table and four blocks – two blue and two yellow. Besides the blocks and the table the Blocks World contains a “hand” that is able to grasp any of the blocks and can thus be used to move the blocks around. Some facts are stated, and some are asserted in the last lines of the file (dynamic facts).

The initial state is as shown in the main picture, and the ‘hand’ does not hold anything.

**Tokenizer.pl:**

The tokenizer converts an input string to a list of tokens (implemented as Prolog atoms) that represent the symbols of the language.

for example: “Put the block onto the table” → [put,the,block,onto,the,table]

**Lexicon.pl:**

The lexicon is a knowledge base, and like the Blocks World, it is modeled using Prologs natively support for facts, deﬁning the fact lex with tree arguments, respectively the symbol, the category and the list of (conjugated) terms used during interpretation. This states that the fact is only true if there exists an entry in the lexicon with the arguments in question.

**Parser.pl:**

The parser follows the *shift-reduce algorithm*. The parser utilizes a stack that at the initial state is empty, and a list of symbols that at the initial state contains the symbols that should be parsed. As long as the algorithm has not deducted a sentence from the input symbols, and has not refused the input as not contained within the language, one of two actions is supported: shift or reduce. These actions are repeatedly chosen non-deterministically:

• Shift – the next symbol is looked up in the lexicon, and the category of a non-deterministic match is pushed onto the stack.

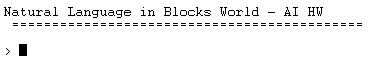
• Reduce – a combinator that applies to the top elements of the stack is non-deterministically chosen to reduce the top elements of the stack.

**Interpreter.pl:**

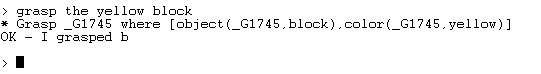
Commands are handled by extracting the action (e.g. grasp or put), while questions are handled by extracting the interrogation (e.g. what). Since all the terms in the remaining expression indeed are facts of the Blocks World (some static and some dynamic) it becomes obvious to solve this expression against the Blocks World’s knowledge base with respect to the variables of the action or interrogation.

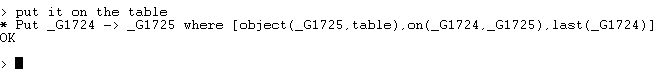
Here are some runs of the program on my machine:

**Welcome screen:**

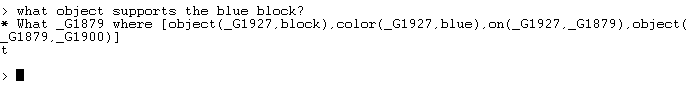


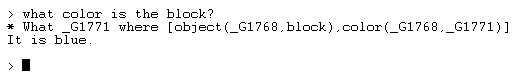
**Example of executing a command:**





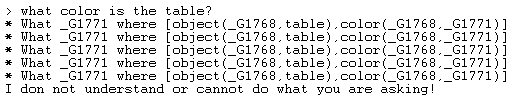
**Example of answering a question:**



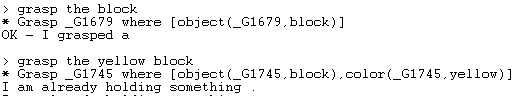


**Examples of failed commands and questions:**

**Invalid question:**



**Trying to grasp twice:**



**\_\_init\_\_.py**

In my research I found the article “Natural Language in Blocks Worlds” by Niklas Christoﬀer Petersen helpful.