CSE 341

Programming Languages

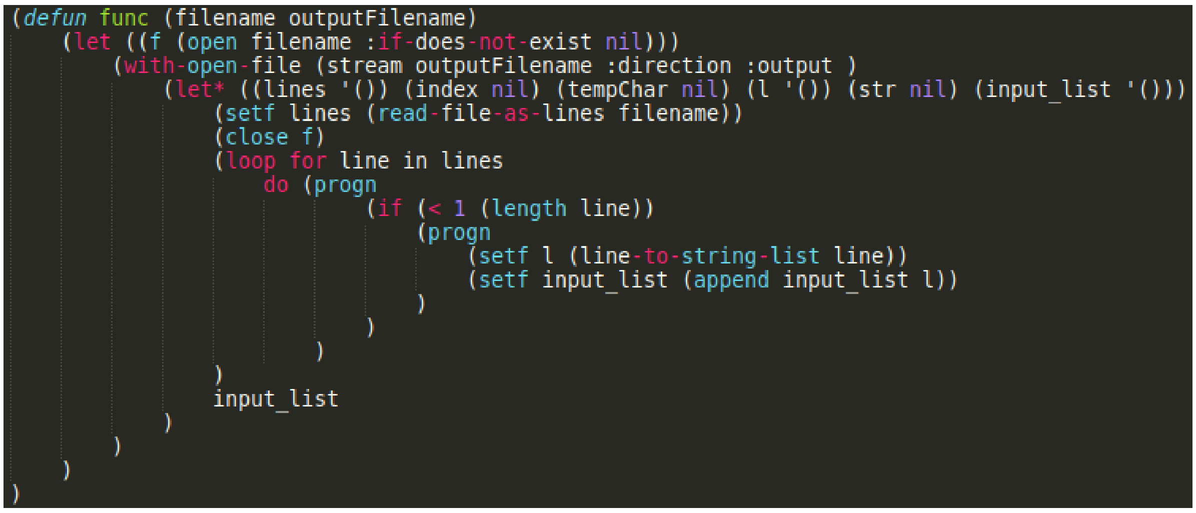
Fall 2020– 2021

HW5-Midterm Report

Caner KARAKAŞ

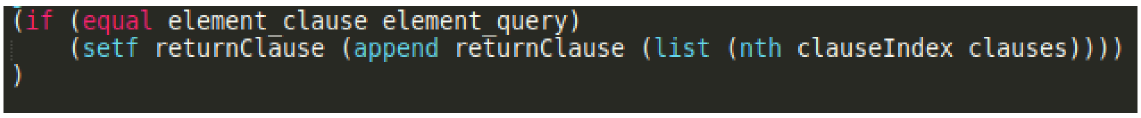
131044061

The input to be read from the file is converted to the format the function expects. Line by line reading operation has been done. The first and last lines will be parentheses and therefore not evaluated. After reading the next lines, they were turned into a list and transferred to the general list.

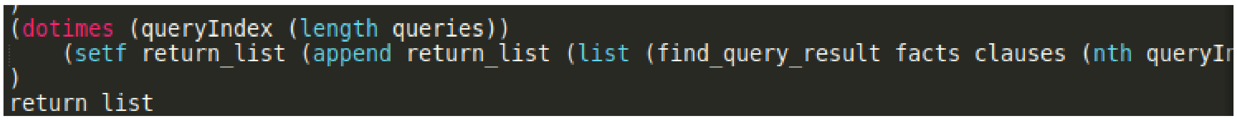


```
(defun func (filename outputFilename)
    (let ((f (open filename :if-does-not-exist nil)))
        (with-open-file (stream outputFilename :direction :output )
            (let* ((lines '()) (index nil) (tempChar nil) (l '()) (str nil) (input_list '()))
                (setf lines (read-file-as-lines filename))
                (close f)
                (loop for line in lines
                    do (progn
                            (if (< 1 (length line))
                                (progn
                                    (setf l (line-to-string-list line))
                                    (setf input_list (append input_list l))
                                )
                            )
                        )
                )
                input_list
            )
        )
    )
)
```

The desired function is splitting into inputs. Shredding processes are kept in lists as clauses, queries and facts. all queries are processed in the loop. The query that comes as an input is searched in the clause list and the addressed clause is found. if more than one, all are kept. It is sent to the process in order. If the correct answer comes, the transaction is terminated. If not, problem solution is continued through the next clause. Returns nil if no result is found or false.

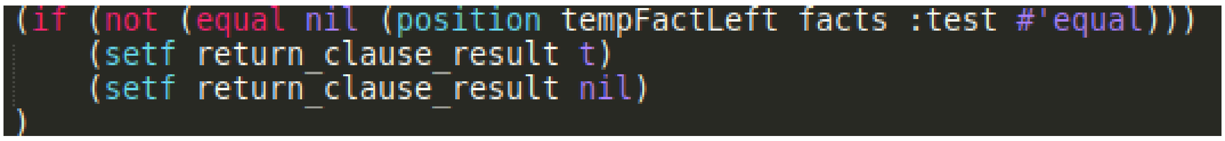


```
(if (equal element_clause element_query)
    (setf returnClause (append returnClause (list (nth clauseIndex clauses))))
)
```



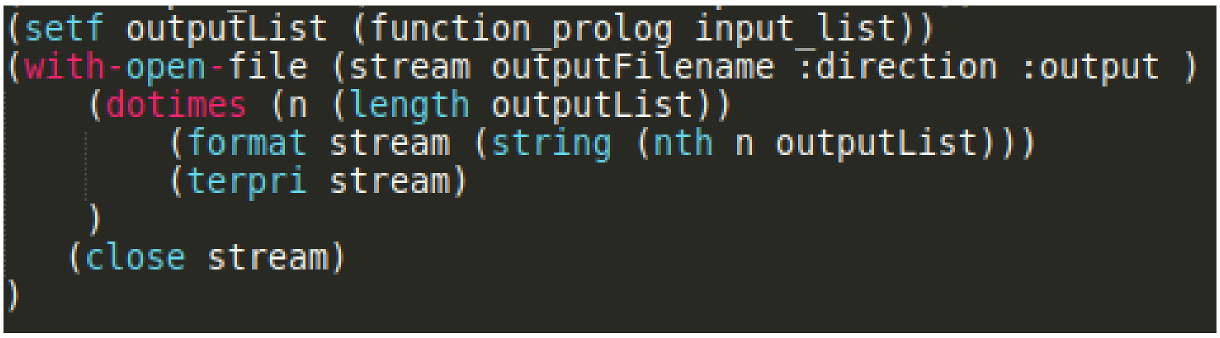
```
(dotimes (queryIndex (length queries))
    (setf return_list (append return_list (list (find_query_result facts clauses (nth queryI
)
return list
```

Another function is used to find the clause result to be the interlocutor. This function aims to split the incoming input according to the given BNF. The entry will be a list and has two elements. The first element is named clause, while the second element holds the predicates part of the input. Operations are performed on the Predicates section. A facts equivalent to Predicate is found and is returned if the result is true, otherwise it returns nil. This applies to all predicates. If it is not provided even for a situation, the answer is intended to be returned nil.

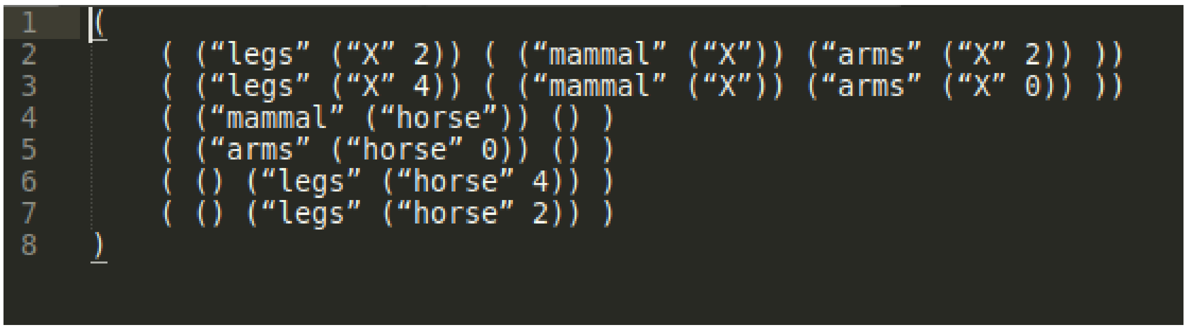


```
(if (not (equal nil (position tempFactLeft facts :test #'equal)))
    (setf return_clause_result t)
    (setf return_clause_result nil)
)
```

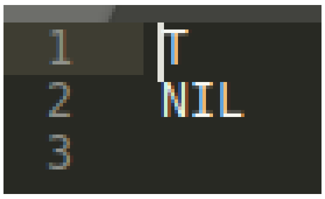
The query results returned are kept in a list. At the end of the process, it is written to the file.



```
(setf outputList (function_prolog input_list))
(with-open-file (stream outputFilename :direction :output )
    (dotimes (n (length outputList))
        (format stream (string (nth n outputList)))
        (terpri stream)
    )
   (close stream)
)
```



```
(
    ( (“legs” (“X” 2)) ( (“mammal” (“X”)) (“arms” (“X” 2)) ))
    ( (“legs” (“X” 4)) ( (“mammal” (“X”)) (“arms” (“X” 0)) ))
    ( (“mammal” (“horse”)) () )
    ( (“arms” (“horse” 0)) () )
    ( () (“legs” (“horse” 4)) )
    ( () (“legs” (“horse” 2)) )
)
```



```
T
NIL
```