CSE 341 - PROGRAMMING LANGUAGES – HW #5(MIDTERM) REPORT

HARUN ALBAYRAK

171044014

My code file : 171044014_Albayrak_Harun.lisp

My Input file : input.txt

My Second Input file : input2.txt

My Third Input file : input3.txt

My Output file : output.txt

Run command : clisp 171044014_Albayrak_Harun.lisp

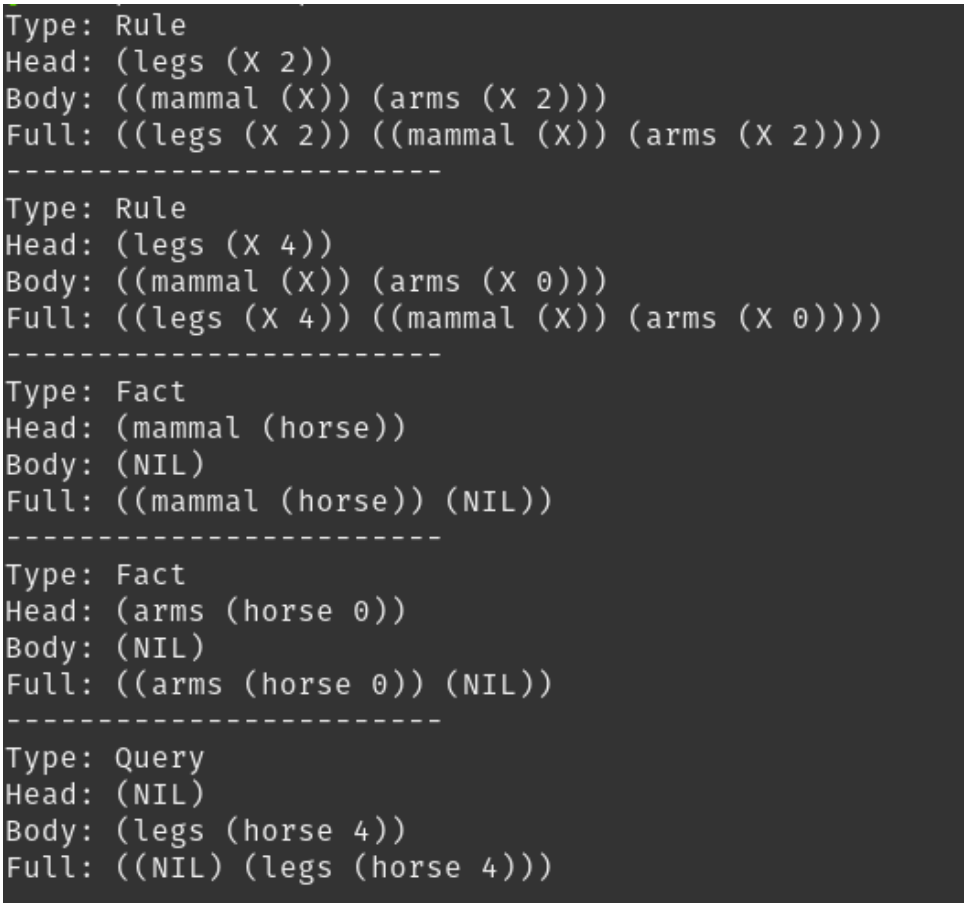
!! NOTE : If the inputs are not given in the same way, the program may not work.

Firstly, I converted given parsed list to clause class object that consist of head, body and full part.

Example:

input.txt

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



If you want to check, you need to active these comment lines : 310,311,312,313,314. Lines in the code file.

After this stage, I append this objects to the list.

The queries are compared to the rules and facts.

And this comparings are done using unification and resolution methods.

Comparing the query and a fact.

Unification :

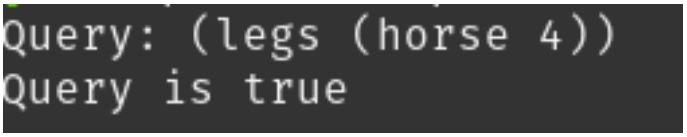
If the query is equal to a fact, then the query is true. (are written to output.txt)

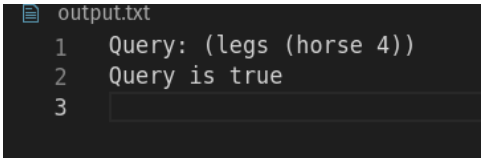
Example:

legs(horse,4).

?- legs(horse,4).

This query will be true.





Unification and Resolution :

If the query has a variable and remaining part of the query is equal to a fact, then values of the variables are returned. (are written to output.txt)

Example:

legs(horse,4).

legs(cow,4).

?- legs(X,4).

input.txt

```
(
    ( (“legs” (“horse” 4)) () )
    ( (“legs” (“cow” 4)) () )
    ( () (“legs” (“X” 4)) )
)
```

```
Query: (legs (X 4))
X: horse
X: cow
```

output.txt

```
Query: (legs (X 4))
X: horse
X: cow
```

Comparing the query and a rule.

If the query's name is equal to rule's name, then their arguments are compared.

If the query has no variable, then resolution and unification are done.

Firstly, the query's arguments are written to facts' arguments in the rule's body. (Unification)

Secondly, the facts' that changed their arguments in the rule's body are compared to facts in the program. If all facts are true, then the query is true.

(are written to output.txt)

Example:

legs(X,2) :- mammal(X), arms(X,2).

legs(X,4) :- mammal(X), arms(X,0).

mammal(horse).

arms(horse,0).

?- legs(horse,4).

The query will be true.

input.txt

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

```
Query: (legs (horse 4))
Query is true
```

output.txt

```
Query: (legs (horse 4))
Query is true
```

This process:

?- legs(horse,4).

legs(horse,4) :- mammal(horse), arms(horse,0).

mammal(horse). --> There is a fact. (True)

arms(horse,0). --> There is a fact. (True)

True & True = True, so this query is true.

If the query has variable, then resolution and unification are also done.

Firstly, The query's arguments are written to facts' arguments in the rule's body with variables.

And the same process works. If the variables in the fact's argument is the same of all fact's argument, the values of variables are returned.

(are written to output.txt)

Example:

legs(X,2) :- mammal(X), arms(X,2).

legs(X,4) :- mammal(X), arms(X,0).

mammal(horse).

arms(horse,0).

?- legs(X,4).

input.txt

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

```
Query: (legs (X 4))
X : horse
X : horse
```

output.txt

```
Query: (legs (X 4))
X: horse
X: horse

```

Example:

input.txt

```
(
    ( (“legs” (“X” 2)) ( (“mammal” (“X” “Y”)) (“arms” (“X” 2)) ) )
    ( (“legs” (“X” 4)) ( (“mammal” (“X”)) (“arms” (“X” 0)) ) )
    ( (“mammal” (“horse” “cow”)) () )
    ( (“mammal” (“horse”)) () )
    ( (“mammal” (“cow”)) () )
    ( (“mammal” (“jaguar”)) () )
    ( (“legs2” (“deneme1” “teker1” 4)) () )
    ( (“legs2” (“deneme2” “teker2” 4)) () )
    ( (“arms” (“horse” 0)) () )
    ( () (“legs” (“A” 4)) )
    ( () (“legs” (“horse” 4)) )
    ( () (“legs2” (“Y” Z” 4)) )
)
```

output.txt

```
Query: (legs (A 4))
A: horse
A: horse
Query: (legs (horse 4))
Query is true
Query: (legs2 (Y Z 4))
Y: deneme1
Z: teker1
Y: deneme2
Z: teker2

```

!! NOTE : If the inputs are not given in the same way, the program may not work.