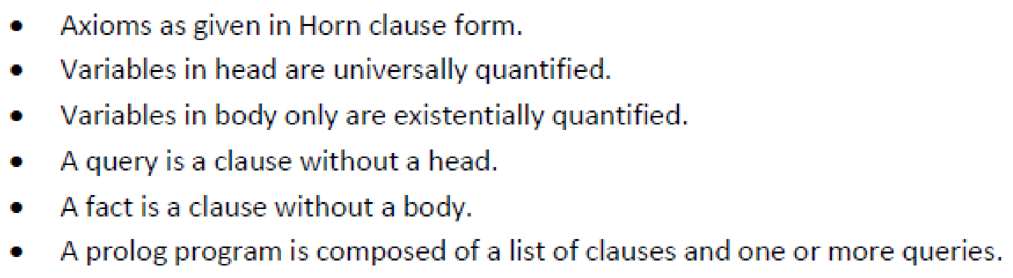
CSE 341 – HW #5

ELİF KELEŞ – 161044033

For this homework, I was asked to write a common lisp program that answers to queries in a list of horn clauses, using resolution and unification methods. As stated in the homework PDF, I assumed the input (from input.txt file) is a parsed list of horn clauses (of a Prolog-like language).

First I got help from the statements below (from the homework PDF) to structure my lists.

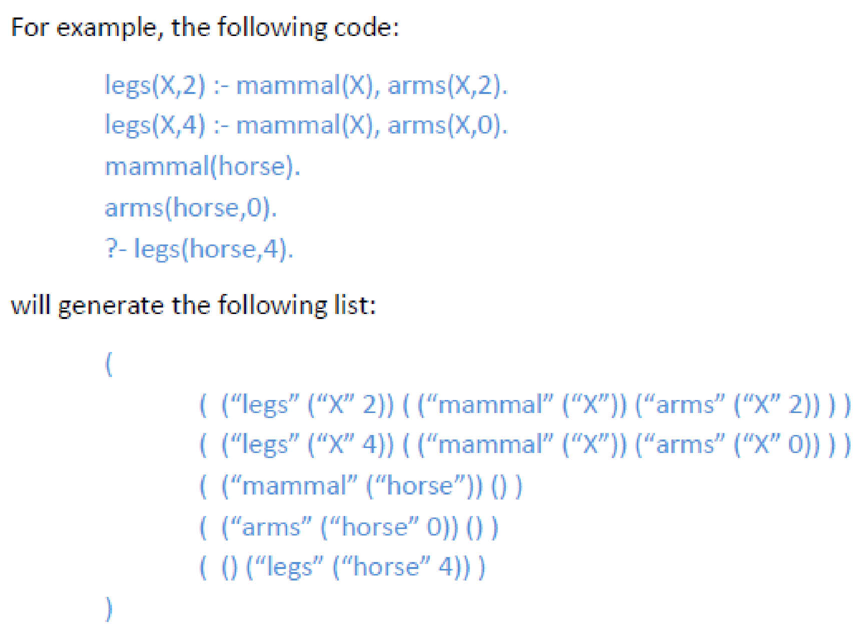


- Axioms as given in Horn clause form.
- Variables in head are universally quantified.
- Variables in body only are existentially quantified.
- A query is a clause without a head.
- A fact is a clause without a body.
- A prolog program is composed of a list of clauses and one or more queries.

Everything in the main list consist of two different lists. And I can decide which is which, by looking up the NIL list. Such as

An axiom -> ((full)(full)), A fact -> ((full)(nil)), A query -> ((nil)(full))

And I used the example in the PDF to create my input file.



For example, the following code:

```
legs(X,2) :- mammal(X), arms(X,2).
legs(X,4) :- mammal(X), arms(X,0).
mammal(horse).
arms(horse,0).
?- legs(horse,4).
```

will generate the following list:

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

After reading the input file, I created my axiom list, fact list, and query list due to this statement. I also created a variable list to keep all the “different ” data I find when comparing with the upper-case lettered value. (i.e in the PDF example we compare “horse” with “X” so the variable here is “horse”)

Then I call answerQuery function which checks for the match between axioms and the query first. If I find a proper axiom, I add the different variable to variable list, then I start to look for a match, between the second element of the axiom list and the fact list. I created a function called addFacts for this.

If the second list of the matching axiom mathces with some of the facts, I add the different value(s) to the variables list.

If there is no axioms (or proper axioms that match with query) I look for a match between facts and the queries. This is basically the same control that I did in the addFacts function. If there is matching values I update the variables list.

In answerQuery and the other functions that I wrote, to be able to jump between elements of the lists, I used lisp functions like car, cdr, cadr, cdar and loops a lot.

After all the comparisons, I created a function (called checkIfEqual) which checks if all the elements in a list are the same or not. If elements are all the same returns true. Otherwise, returns false.

Then I assign the return value of checkIfEqual function, to a flag that I keep in answerQuery. If it is true the output will be a true list, if it is false the list will be an empty list. (as () or (T))

The result value is written to output.txt file. If the file exist, I append the result.

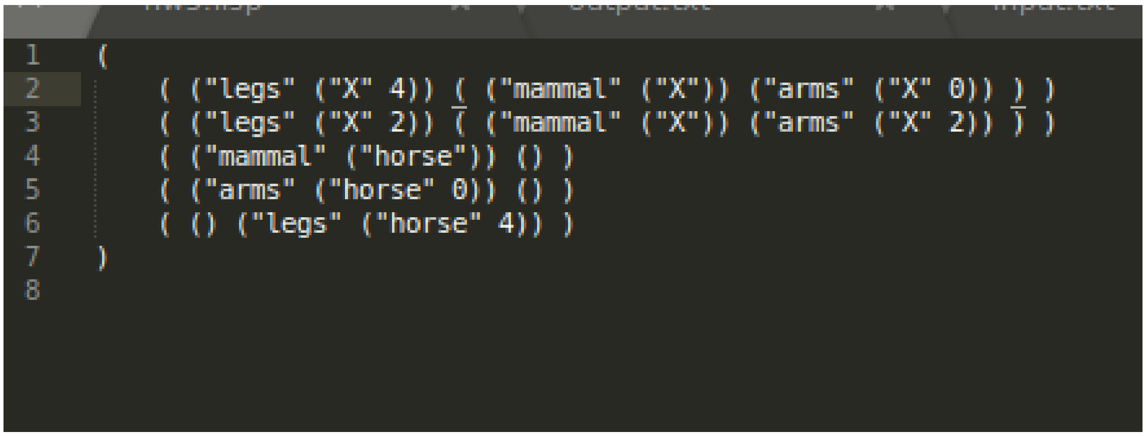
**I assumed the input file is given in the PDF (other than the empty list)

Refering to: the unknown character is upper-case and at the index zero of its own list,

there is no list which has more than two elements,

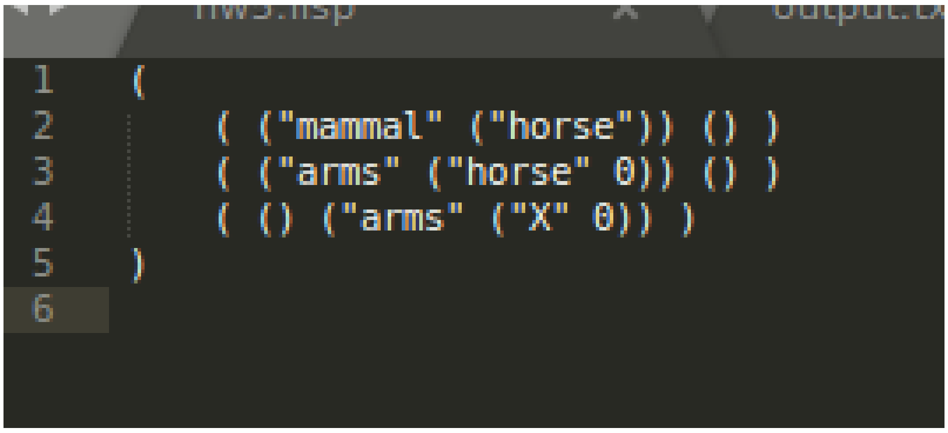
has the same parenthesis order,

such as



```
(
    ( ("legs" ("X" 4)) ( ("mammal" ("X")) ("arms" ("X" 0)) ) )
    ( ("legs" ("X" 2)) ( ("mammal" ("X")) ("arms" ("X" 2)) ) )
    ( ("mammal" ("horse")) () )
    ( ("arms" ("horse" 0)) () )
    ( () ("legs" ("horse" 4)) )
)
```

or



```
(
    ( ("mammal" ("horse")) () )
    ( ("arms" ("horse" 0)) () )
    ( () ("arms" ("X" 0)) )
)
```