Run (query) and tell it some facts

STk> (load "query.scm")
okay
STk> (query)
::: Query input:
(assert! (colleen likes cookies))
Assertion added to data base.
::: Query input:

Query system goal:

• Fill in the variables in the query (with all possible sets of values)

• NEVER fill in the variables with an inconsistent set of variables

Query: (?who is the parent of bart)

TWO facts match:

(?who is the parent of bart) ;query
(homer is the parent of bart) ;fact

?who = homer

These are frames!

Query: (?who is the parent of bart)

(?who is the parent of bart) ;query
(marge is the parent of bart) ;fact

?who = marge

(?who is the parent of bart)

(female marge)

(?who = marge)

(female lisa)

(female marge)

(female jackie)

(female lisa)

(female marge)

(female patty)

(female marge)

(female patty)

(female salma)

(female marge)

(female salma)

(female lisa)

(female salma)

(female marge)

(female lisa)

(female marge)

These are frames!
Query: (marge is the parent of ?x)

(abraham is the parent of herb) (her is the parent of bart)
(mona is the parent of herb) (her is the parent of maggie)
(mona is the parent of homer) (her is the parent of lisa)
(clancy is the parent of marge) (marge is the parent of bart)
(clancy is the parent of patty) (marge is the parent of lisa)
(clancy is the parent of selma) (selma is the parent of marge)
(jackeline is the parent of marge) (female mona)
(jackeline is the parent of patty) (female jackie)
(jackeline is the parent of selma) (female marge)
(female selma) (female lisa)
(female lisa)
(female maggie)
(female ling)

(rule (?a is the car of (?a . ?b)))

Query: (female ?who)

(abraham is the parent of herb) (her is the parent of bart)
(mona is the parent of herb) (her is the parent of maggie)
(mona is the parent of homer) (her is the parent of lisa)
(clancy is the parent of marge) (marge is the parent of bart)
(clancy is the parent of patty) (marge is the parent of lisa)
(clancy is the parent of selma) (selma is the parent of marge)
(jackeline is the parent of marge) (female mona)
(jackeline is the parent of patty) (female jackie)
(jackeline is the parent of selma) (female marge)
(female selma) (female lisa)
(female lisa)
(female maggie)
(female ling)

(rule (?a is the car of (?a . ?b)))

We can write a query that returns NOTHING
• NO knowledge of ANY variables, return nothing.

Write a query that returns 0 results

(abraham is the parent of herb) (her is the parent of bart)
(mona is the parent of herb) (her is the parent of maggie)
(mona is the parent of homer) (her is the parent of lisa)
(clancy is the parent of marge) (marge is the parent of bart)
(clancy is the parent of patty) (marge is the parent of lisa)
(clancy is the parent of selma) (selma is the parent of marge)
(jackeline is the parent of marge) (female mona)
(jackeline is the parent of patty) (female jackie)
(jackeline is the parent of selma) (female marge)
(female selma) (female lisa)
(female lisa)
(female maggie)
(female ling)

Query: (marge is the parent of ?x)

All facts

Query

QEval

{ }

How many things did you figure out?
A. 0 B. 1 C. 2 D. 3 E. ??

Write a query that returns 0 results

(?who = mona)
(?who = jackie)
(?who = marge)
(?who = patty)
(?who = selma)
(?who = lisa)
(?who = maggie)
Query Goals:

- Fill in the variables in the query (with all possible sets of values)
- NEVER fill in the variables with an inconsistent set of variables
For EACH frame we need to compare it to EVERY fact!

Query 2:
(female ?who)

?who = marge
?who = marge

Query Goals:
• Fill in the variables in the query (with all possible sets of values)
• NEVER fill in the variables with an inconsistent set of variables

(and
(?who is the parent of bart)
(female ?who))

?who = marge
?who = marge

How many results?
A. 0 B. 1 C. 2 D. 3 E.?
How many results do you think there SHOULD be? A. 2 B. 4 C. ?

;;; Query input:
(or (x is the parent of bart)
  (x is the parent of lisa))

;;; Query results:
26

rules

;;; Query input:
(assert! (rule <the-rule> <the-body>))
When a rule (a type of fact) matches, do this:

;;; Query input:
(assert! (rule (parent-of-bart ?person)
  (?person is the parent of bart)))

rules

(rule (parent-of-bart ?person)
  (?person is the parent of bart)))

;;; Query input:
(parenent-of-bart ?who)

;;; Query input:
(mona is the parent of ?x);q1
(?x is the parent of ?y);q2

Solutions
### Query input:
(and (mona is the parent of ?x)
  (?x is the parent of ?y))

### Query results:
(and (mona is the parent of homer)
  (homer is the parent of bart))
(and (mona is the parent of homer)
  (homer is the parent of lisa))
(and (mona is the parent of homer)
  (homer is the parent of maggie))

### Solutions

<table>
<thead>
<tr>
<th>?x = homer</th>
<th>?y = bart</th>
<th>?y = lisa</th>
<th>?y = maggie</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = homer</td>
<td>x = homer</td>
<td>x = homer</td>
<td>x = homer</td>
</tr>
</tbody>
</table>

### Query input:
(or (?x is the parent of bart)
  (?x is the parent of lisa))

### Query results:
(or (marge is the parent of bart)
  (marge is the parent of lisa))
(or (homer is the parent of bart)
  (homer is the parent of lisa))
(or (selma is the parent of bart)
  (selma is the parent of lisa))

### Solutions

<table>
<thead>
<tr>
<th>?x = marge</th>
<th>?x = homer</th>
<th>?x = marge</th>
<th>?x = selma</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = marge</td>
<td>x = homer</td>
<td>x = marge</td>
<td>x = selma</td>
</tr>
</tbody>
</table>