CS131: Programming Languages

Fall 2015
Week #5

Today

- Prolog Introduction
- HW #4

Prolog: Introduction

- Declarative/logic programming language
- 2 main usage modes:
  - Declare facts
  - Run a query on the fact database

Prolog: Basic Syntax

- Declare facts:

```
sunny.
friday.
at_ucla.
```

- Query the database:

```
| ?- sunny.          | ?- at_usc.
yes                  no
```

the query

the answer

Prolog: Predicates/Relations

- Predicates characterize or relate items

```
male(bart).
male(homer).
female(marge).

parent(marge,bart).
parent(homer,bart).
```

Prolog: Variables & Unification

- Non-variables begin with lowercase letters
- Variables begin with uppercase letters
- Can use variables in queries:
  - Prolog tries to provide all values for the variables that satisfy the query

```
| ?- parent(X,bart).

X = marge ? ;

X = homer

yes
```

typing a semicolon tells Prolog to give you the next match

Prolog: Unification

- Prolog tries to unify (match) variables to what it knows
- Will the following unify (based on facts from before):

```
| ?- parent(bob,X).

| ?- parent(X,X).

| ?- parent(X,Y).

| ?- parent(Y,X)
```

Prolog: Rules/Clauses

```
p(X) :- a(X), b(X).
```

- For a given X, if a(X) and b(X), then p(X)

● How would we express:

```
mother(X,Y) :-
```

Prolog: Rules/Clauses

```
p(X) :- a(X), b(X).
```

- For a given X, if a(X) and b(X), then p(X)

* How would we express:

```
mother(X,Y) :- parent(X,Y), female(X).
```

Prolog: Rules/Clauses

- `grandparent(X,Z) :-`

Prolog: Rules/Clauses

- ```
  grandparent(X,Z) :-
      parent(X,Y) , parent(Y,Z).
  ```

- Rules may be recursive.

  ```
  ancestor(X,Y) :-
  ```
  ```

Prolog: Rules/Clauses

- `grandparent(X,Z) :-`

  `parent(X,Y) , parent(Y,Z).`

- Rules may be recursive.

```
ancestor(X,Y) :- parent(X,Y).

ancestor(X,Y) :-

    parent(Z,Y) , ancestor(X, Z).
```

Prolog: List Syntax

- Syntax for list:
  - `[H|T]` - H is the head, T is the tail (another list)
  - we can put this wherever we put a value or variable
- How do we write:
  - `head(X,Y) % true if X is the head of the list Y`

Prolog: List Syntax

- Syntax for list:
  - `[H|T]` - H is the head, T is the tail (another list)
  - we can put this wherever we put a value or variable
- How do we write:
  - `head(X,Y) % true if X is the head of the list Y`
    - **`head(H,[H|_]).`**

Prolog Practice

- Define `append(X,Y,Z)` which is true if Z is the result of appending X to Y

Prolog Practice

- Define `append(X,Y,Z)` which is true if Z is the result of appending X to Y

```
append([],L2,L2).

append([H|T],L2,[H|L3]) :- append(T,L2,L3).
```

Prolog Practice

- Define `reverse(X,Y)` which is true if Y is the reverse list of X

Prolog Practice

- Define `reverse(X,Y)` which is true if Y is the reverse list of X

```
reverse([],[]).

reverse([H|T],R) :- reverse(T,Z), append(Z,[H],R).
```

Prolog: Unification (cont'd)

- Can use '=' to bind variables as well as to compare data structures in both directions

```
X=f(Y).
f(g(Y))=f(X).
X=1+2.
3=1+2.
```

it returns "X = g(Y)" (for `f(g(Y))=f(X).`)

it does not evaluate expressions (for `X=1+2.`)

no, fails (3 is syntactically different) (for `3=1+2.`)

Prolog: is keyword

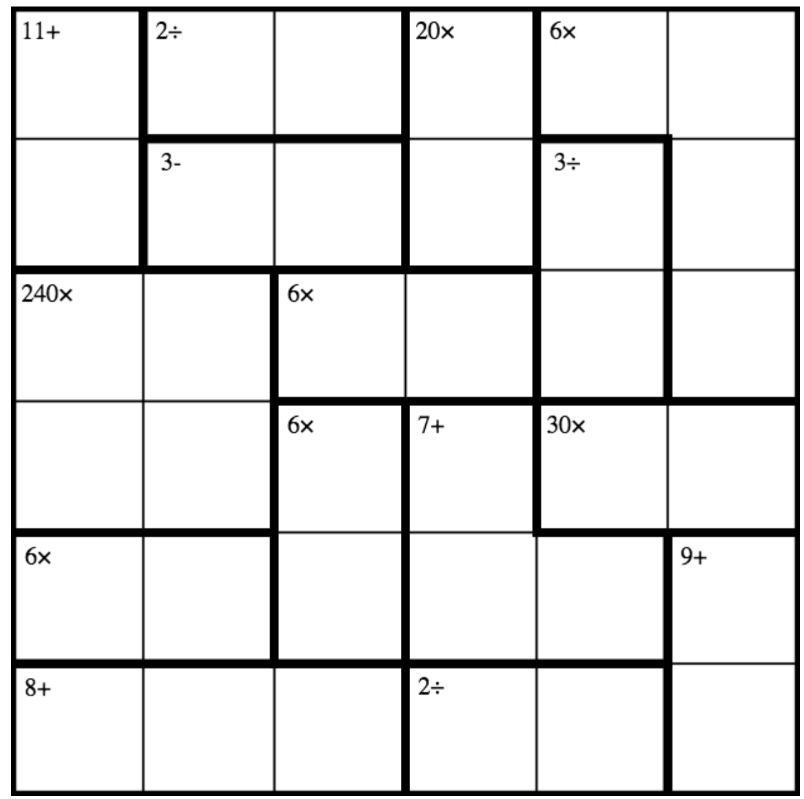
- "is" is a built-in arithmetic evaluator

```
X = 5, Y is 2 * X
```

- X is E
  - First computes the arithmetic expression E and then unifies the result with X
  - E can contain variables (but only numbers)

HW 4: KenKen Solver

- KenKen is a number game where you fill in blanks so that some conditions are satisfied
- N x N grid
- Each row, column contains all 1,2,3,...N exactly once
- In each bolded area, the mathematical condition holds



HW 4: Representation

```
kenken(N, C, T) :-
```

- N is the length of the rows/columns
- C is the set of constraints:

```
[
   +(11, [1-1, 2-1]),
   /(2, 1-2, 1-3),
   *(20, [1-4, 2-4]),
   -(3, 2-2, 2-3),
   ...
]
```

- T is the solution: a matrix of numbers

HW 4: KenKen Solver

- Use GNU Prolog's finite domain solver
  - Allows you to set the search space to a finite domain of numbers
- Key predicates:
  - `fd_domain`
  - `fd_all_different`
  - `fd_labeling`

http://www.gprolog.org/manual/html_node/gprolog054.html