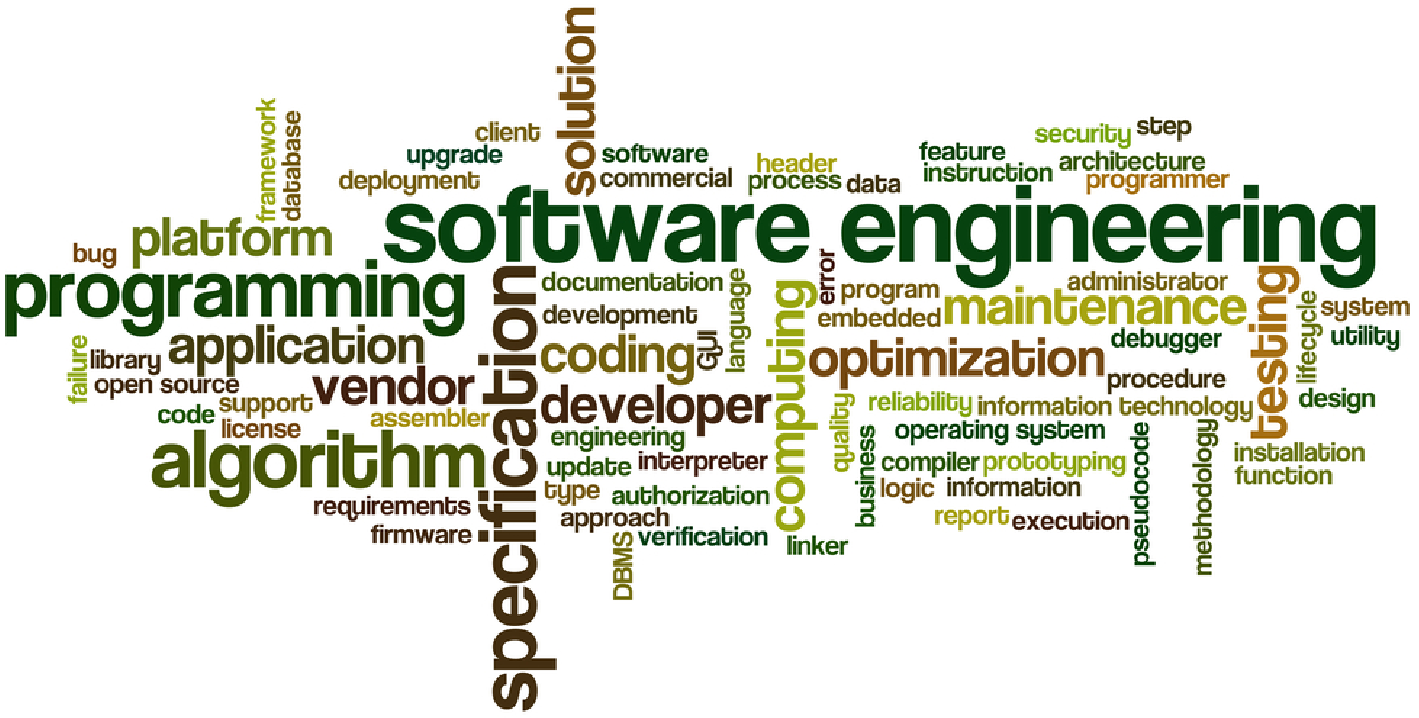


SWE 205: Introduction to Software Engineering

Lecture 14

Programming Paradigms
Logic Programming

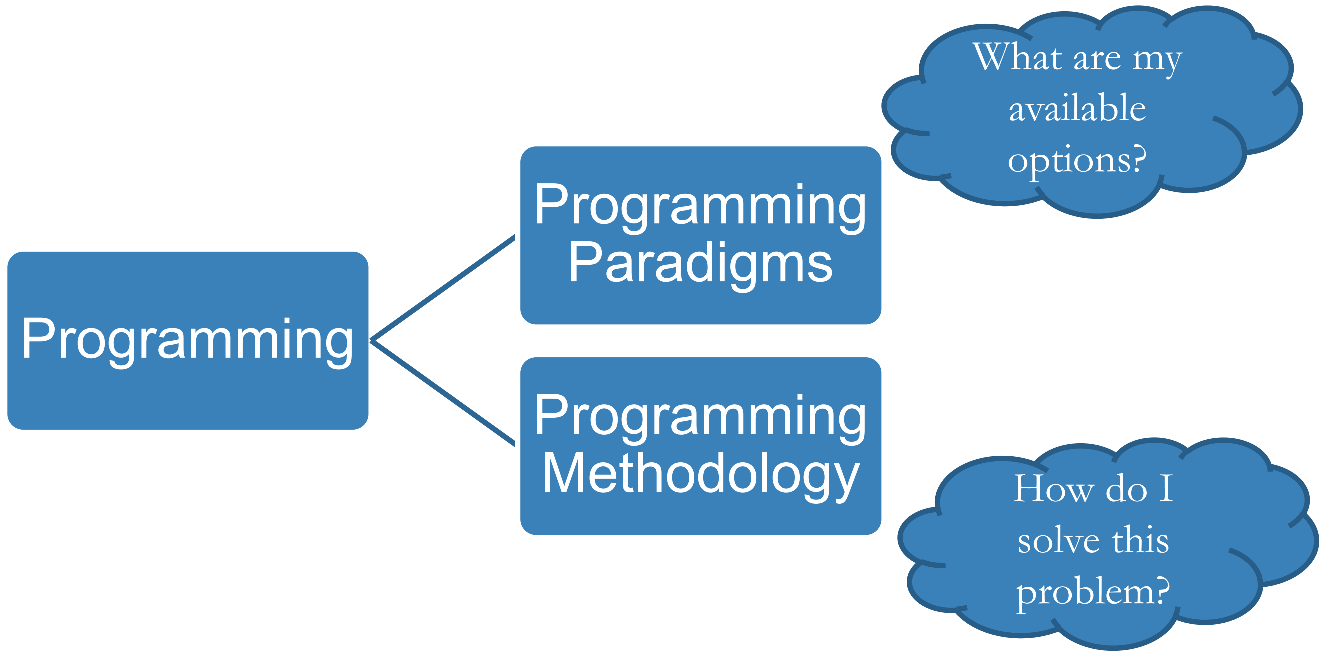
Course Topics

- ~~Introduction~~
- ~~Software Process Models~~
- ~~Requirements Engineering~~
- ~~Modeling~~
- Software Construction Techniques
- Testing
- Project Management
- Refactoring
- Ethical Issues

Lecture Objectives

- ✓ To know the basics of programming languages
  - Logic Programming
  - Prolog

Paradigm Vs Methodology

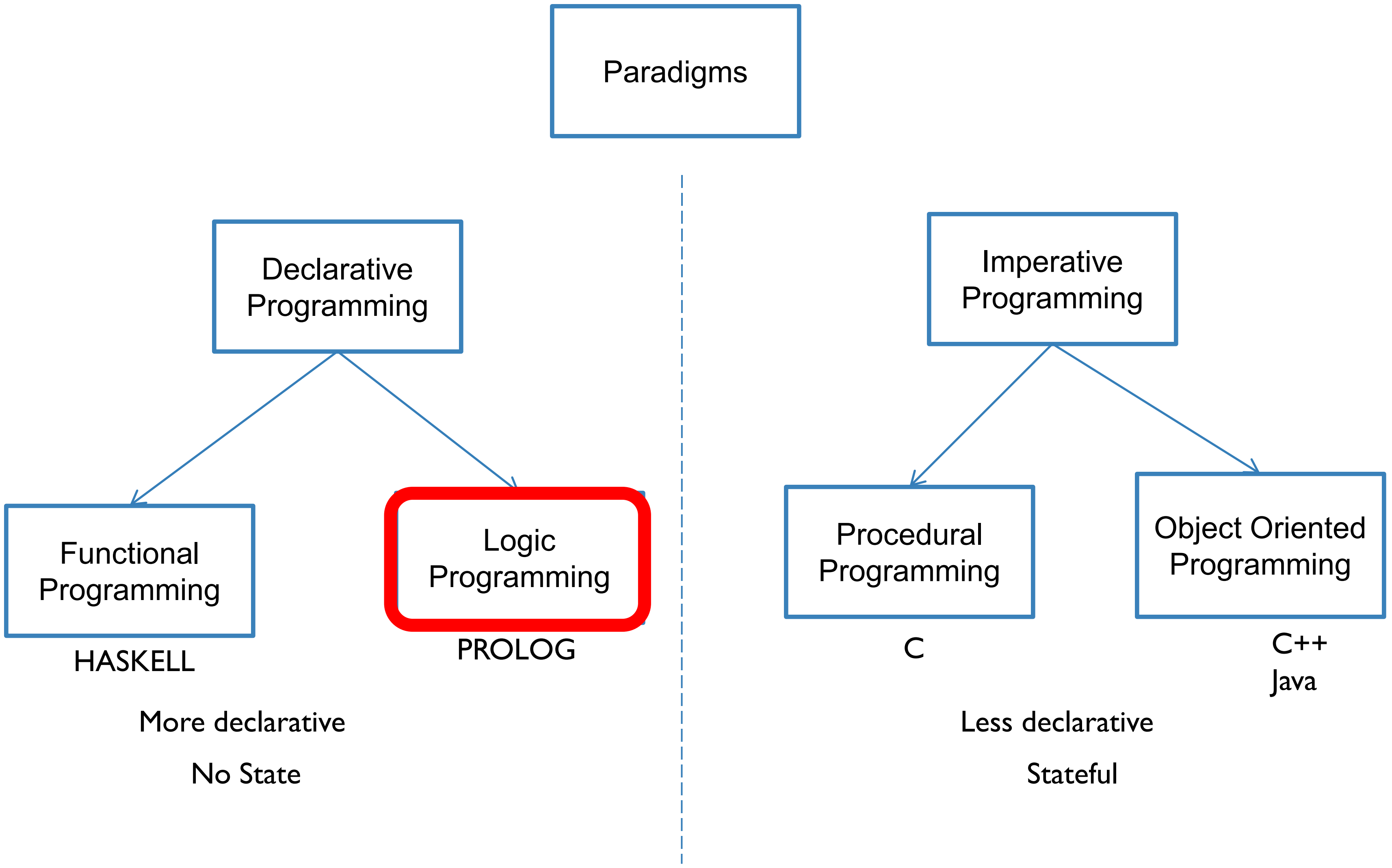


Declarative Vs Imperative

- **Declarative programming** is a programming paradigm in which programs describe the desired results of the program, without explicitly listing command or steps that need to be carried out to achieve the results.

- **Imperative programming** is a programming paradigm that describes computation in terms of statements that change a program state. Imperative programs define sequences of commands for the computer to perform.

Programming Paradigms



Logic Programming Paradigm

- Problem Solving = Problem Description + Logical Deductions

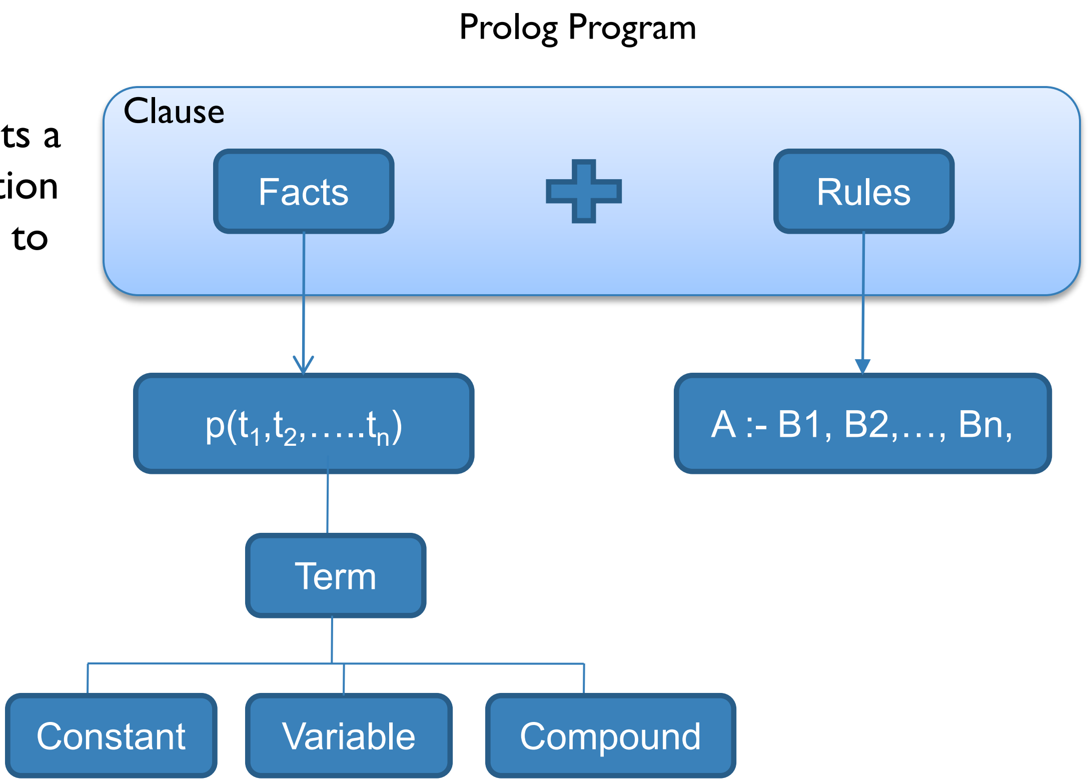
- In a logic program, we tell the computer **‘what’** we want it to do and not **‘how’** we want it to do it (as in imperative).

- Declare WHAT the problem is, and leave it to the computer to use built-in reasoning to solve the problem

Logic Programming Paradigm

- Prolog is an example of Logic programming language.

A fact represents a unit of information that is assumed to be true.



A rule represents a conditional assertion (‘this is true if this is true’).

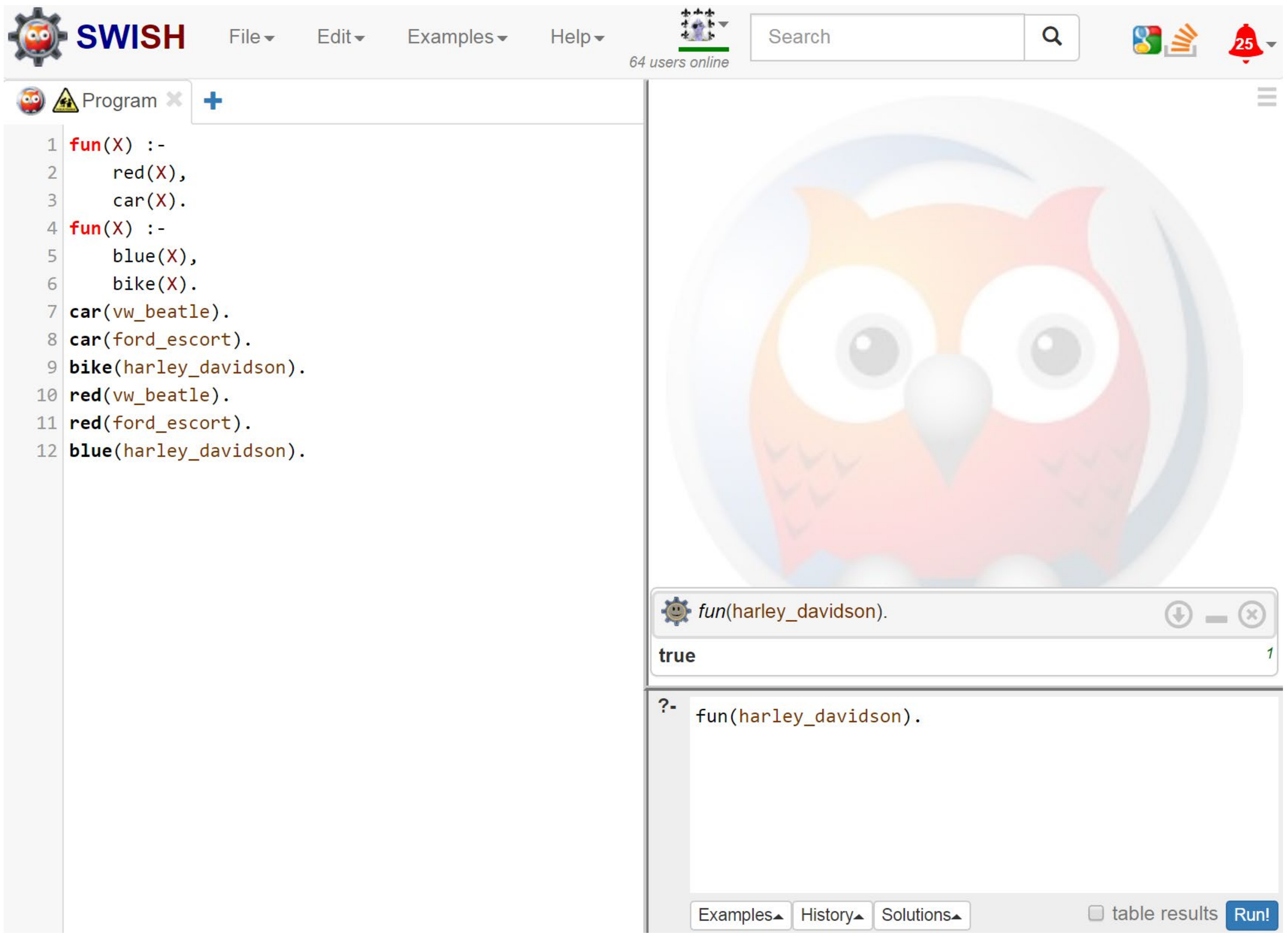
Prolog

- A logic programming language created in 1972
- PROgramming in LOGic
- Facts in Prolog
  - **father(peter).**
  - **woman(mia).**

- Asking Prolog is done in the Interpreter window:
  - **Prolog** listens to your queries and answers:
    - **?- father(peter).   % asking if peter is a father**
    - **true**

Online Prolog compiler

https://swish.swi-prolog.org/



```
fun(X) :-
    red(X),
    car(X).
fun(X) :-
    blue(X),
    bike(X).
car(vw_beatle).
car(ford_escort).
bike(harley_davidson).
red(vw_beatle).
red(ford_escort).
blue(harley_davidson).
```

```
fun(harley_davidson).
true
```

```
?- fun(harley_davidson).
```

Prolog: Example

If we have the Prolog program:

- male(charles).
- male(edward).
- male(philip).
- female(anne).
- parent(philip, anne).
- parent(philip, edward).
- parent(philip, charles).

- father(X,Y) :- parent(X,Y), male(X).

Here is how you would formulate the following queries:

1) Is Anne a female ?

Query:

```
?- female(anne)
true
```

2) Who is the daughter of Philip ?

Query:

```
?- father(philip,Y),female(Y)
```

Will return

```
Y = anne
```

Prolog: Example

If we have the Prolog program:

- male(charles).
- male(edward).
- male(philip).
- female(anne).
- parent(philip, anne).
- parent(philip, edward).
- parent(philip, charles).

Then the query

- ?- parent(X, charles), parent(X,Y), male(Y).

Will return

- X = philip,
- Y = edward;

Questions:

- Facts:
  - eats(fred, oranges). %Fred eats oranges
  - eats(fred,t_bone_steaks).
  - eats(tony, apples).
  - eats(john, apples).
  - eats(john, grapefruit).
  - ?- eats(fred, oranges). yes
  - ?- eats(mike, apples). no
  - ?- eats(fred, apples). no

Logic Programming Paradigm

- Can you think of a program that follows the same programming paradigm?

HINT (Databases)

Answer: SQL

Prolog

- Rules
  - Form: Head :- body
  - Meaning: body leads to head
  - Example: person(X) :- male(X).
    - If X is a male then X is a person
- Composite rules - AND
  - You can specify more conditions in the rule
  - Example: father(X,Y) :- parent(X,Y), male(X).

Prolog

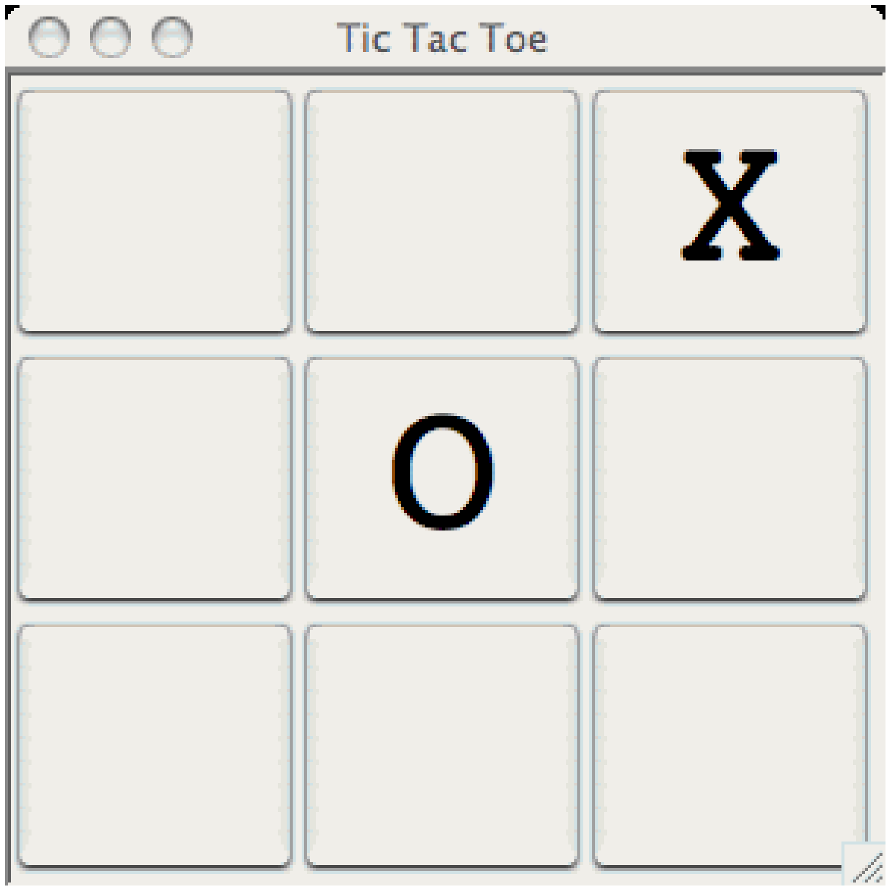
- Composite rules – OR
  - Just list the rules with different results
  - Example

```
person(X) :- male(X).
person(X) :- female(X).
```

  - Meaning a person is either a male or a female

Interesting Applications

- Combine Prolog with Java
- Tic-tac-toe



Questions:

```
- fun(X) :-
      red(X),
      car(X).
- fun(X) :-
      blue(X),
      bike(X).
- car(vw_beatle).
- car(ford_escort).
- bike(harley_davidson).
- red(vw_beatle).
- red(ford_escort).
- blue(harley_davidson).
```

- Let's now use the program and see if a harley_davidson is fun?
- ?- fun(harley_davidson).
- yes

Key Points

- Programming Paradigms
  - Declarative and Imperative

- Logic programming depends on facts to build information and on queries to retrieve those pieces of information

References

- Ian Sommerville, “Software Engineering”, 10th Edition, Addison-Wesley, 2015.
- Timothy C. Lethbridge and Robert Laganière, “Object-Oriented Software Engineering: Practical Software Development using UML and Java”, 2nd Edition, McGraw Hill, 2001.
- R. S. Pressman, Software Engineering: A Practitioner’s Approach, 10th Edition, McGraw-Hill, 2005.