DRONACHARYA College of Engineering

IS (CSE-303-F)

Section A

AI PROGRAMMING LANGUAGES: INTRODUCTION TO LISP & PROLOG

Prelude

Computer languages can be categorized in many ways depending on what aspects to be observed. This is also true for Prolog and Lisp.

o **Historical generations of languages**

1st	**Machine**
2nd	**Assembly**
3rd	**High-level, e.g., C, C++, Java, Fortran, Cobol, . . .**
4th	**Very high-level, e.g., SQL for databases**
5th	**Symbolic, e.g., Prolog and Lisp** Use symbols (rather than variables and constants as main program elements) Symbolic languages are often termed as AI languages.

§ Prolog

PROgramming in LOGic

Born in the early 1970s in France

A <u>declarative</u> language based on predicate logic

Most high-level languages, such as C, C++ and Java, are <u>procedural</u>.

This makes Prolog unique and interesting.

**In a declarative language, one specifies "what“. In contrast,
in a procedural language, one writes "how" the program is executed.**

Lisp, which will be discussed in the next section, is both <u>declarative</u> and <u>procedural</u>.

- **Application domain includes:**

Knowledge-based (so-called expert) systems,

natural language processing (e.g., Jeopardy!)

compiler writing,

symbolic algebra,

VLSI circuit analysis,

relational databases, (e.g., Jeopardy!) and

image processing.

SWI-Prolog is a public-domain Prolog and is available at:

http://www.swi-prolog.org

•Basic Prolog elements

o Facts

A <u>fact</u>, e.g., “John owns the book” is represented as:

owns (john, book).

↗ ↑

predicate arguments (any number of arguments)

o Questions

?-owns (john, book). or owns (john, book)?

Prolog responds → “Yes” given the above fact.

o A <u>constant</u> (e.g., john) starts with a lower-case letter.

A <u>variable</u> (e.g., X, _3) starts with an upper-case letter or “_”.

o Rules

Examples of rules

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

X is a parent of Y if X is the mother of Y.

“:-” is read as “if”; “,” as logical “AND”; “|” as logical “OR” .

```
parent(X,Y) :-  father(X,Y).
grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
sibling(X,Y):- mother(M,X), mother(M,Y),
               father(F,X), father(F,Y).
```

e.g.,

```
mother (cathy, bob).
father (bob, ann).
grandparent(X, ann)? →X = cathy
```

Prolog facts and rules are called <u>clauses</u>.

A set of clauses (i.e., facts and rules) with the same predicate is called a <u>procedure</u>.

Example

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

The program elements we have seen such as owns, john, book, parent and mother are symbols. We can use these symbols without declaring that they are symbols in a symbolic language such as Prolog and Lisp.

This is not the case for conventional, non-symbolic languages such as Java and C.

o structures

e.g.,

class (cis, 524, 50)

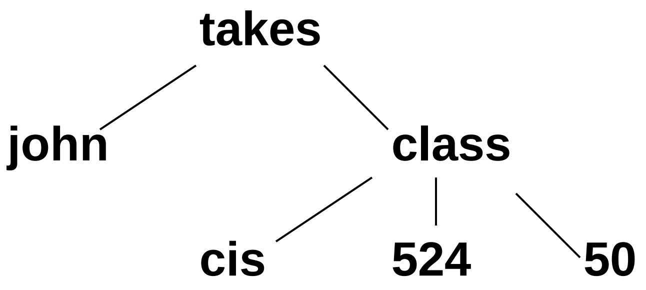
↑ ↑

functor arguments

takes (john, class (cis, 524, 50))

“takes” is the functor and there are two arguments: an atom “john” and a structure “class (cis, 524, 50)”.

A structure can be represented as a tree.



Exercise #1 (A simple database in Prolog)

a. Set up a database describing three types of relations for capitals, countries, and continents using the following information:

Capitals	Latitude Longitude	Countries	Continents
Washington_DC	(38 -77)	USA	North_America
Ottawa	(45 -76)	Canada	North_America
London	(51 0)	United_Kingdom	Europe
Paris	(48 2)	France	Europe
Rome	(41 12)	Italy	Europe
Lagos	(6 -3)	Nigeria	Africa

Three types of relations for the above data:

capital_of	e.g.,	capital_of (paris, france).
country_in	e.g.,	country_in (usa, north_america).
location	e.g.,	location (ottawa, 45, -76).

The latitude and longitude figures are given in degrees, where North and East are positive, and South and West are negative integers.

Exercise #1 (cont.)

b. Write the following questions as Prolog queries.

1. **Is Rome the capital of France?**

   Ans. ?- capital_of (rome, france).

2. **Is Washington_DC the capital of a country in Europe?**

   Ans. ?- capital_of (washington_dc, X), country_in (X, europe).

3. **Which city is the capital of Italy?**

   Ans. ?- capital_of (X, italy).

4. **Which cities are West of Rome?**

   Ans. ?- location (X, _, V), location(rome, _, V1), V < V1.

5. **List the European countries whose capitals are North of Rome and South of London.**

   Ans. ?- country_in (Y, europe), capital_of (X, Y), location (X, U, _), location (london, U1, _), location (rome, U2, _), U < U1, U > U2.

o Lists

e.g., [a, b, c] is a <u>list</u> having three elements a, b and c.

> **"a" is the <u>head</u> of the list.**
>
> **"[b, c]" is the <u>tail</u> of the list.**
> Note. The tail is not "b, c".

[Head | Tail] format.

> **When [Head | Tail] = [a, b, c], Head = a and Tail = [b, c].**

Format variations

[X, Y | L], where X and Y are the first two elements and L is the list of the remaining elements.

[Head, . . Tail], [a, b, . . L].

Exercises

- **Define procedure h(X, L) that succeeds if X is the head of list L.**

Ans. h(X, [X | L]).

Then,

```
?-h(a, [a, b, c]). → yes
?-h(X, [a, b, c]). →X = a
?-h(X, [ ]). → no
```

Exercises (cont.)

- **Define procedure last(X, L) that succeeds if X is the last element of list L.**

Strategy: Consider key representative scenarios.

(1) If L is empty, last should fail. Specify no clause.
(2) L contains one element.

```
last(X, [X]).
```

(3) General case
The last element of [_ | Y] is the last element of Y.

```
last(X, [_ | Y]) :- last(X, Y).
```

Then,

```
?-last(c, [a, b, c]). → yes
?-last(X, [a, b, c]). →X = c
?-last(X, [ ]). → no
```

Programming hints for Prolog lists

1. First, consider simplest cases.

2. Represent a list in {H | T] form, and manipulate H and T.

3. The use of recursion is very common for T.

Exercises (cont.)

- Define procedure

  Append (L1, L2, L3), where L1 and L2 are appended giving L3.

  Any one of L1, L2 and L3 can be uninstantiated (no value assigned).

  e.g., ?- append ([a, b], [c], L3). → L3 = [a, b, c]

Ans.

```
append ([ ], L, L).    % Appending an empty list to L gives L.

append ([X | L1], L2, [X | L3]) :- append (L1, L2, L3).
```

Exercises (cont.)

• insert(X, L1, L2) inserts element X into an appropriate place of list L1, giving list L2. e.g., ?-insert(c, [a, b, d], L2) → L2 = [a, b, c, d]

```
insert(X, [ ], [X]).  % insert to an empty list
insert(X, [X | T], [X | T]). % if X = head of L1, do not insert to avoid a duplicate.
insert(X, [H | T], [X, H | T]) :- X < H. % if X < H then insert X at the biginning.
insert(X, [H | T1], [H | T2]) :- X > H, insert(X, T1, T2).
```

• sort(L1, L2) sorts elements of L1 giving L2.

```
sort([ ] , [ ]).
sort([H | T], S) :- sort(T, L), insert(H, L, S).
```

In a declarative language, one specifies "what“.
In contrast,
in a procedural language, one writes "how" the program is computed.

Think about what Java or C code looks like for sorting.

Prolog References

<u>Books</u>

Clocksin, W. F. and Mellish, C. S. Programming in Prolog, 4th Ed. Springer-Verlag, 1994.
A widely-used textbook. Compact descriptions.

Ivan Bratoko, Prolog Programming for Artificial Intelligence, 4th edition, Addison-Wesley, 2011.
Detailed descriptions.

Journal articles published by Munakata

T. Munakata. "Procedurally Oriented Programming Techniques in Prolog", *IEEE Expert*, 1, 2, Summer, 1986, pp. 41-47.
Discusses how conventional programming structures such as do-while and if-then-else can be implemented in Prolog.

T. Munakata. "Notes on Implementing Sets in Prolog," *Communications of the ACM*, 35, 3, March 1992, pp. 112-120.

T. Munakata. "Notes on Implementing Fuzzy Sets in Prolog," *Fuzzy Sets and Systems*, 98, 3, Sept., 1998, pp. 311-317.

T. Munakata and R. Barták, "Logic Programming for Combinatorial Problems," *Artificial Intelligence Review*, Published online: November 2009, DOI 10.1007/s10462-009-9150-5; published hardcopy: 33, 1, 2010, pp. 135-150.
Implementation of combinatorial problems such as permutations and combinations in Prolog is discussed.

§ Lisp

LISt Processing

Lisp was originated by John McCarthy in the late 1950s.

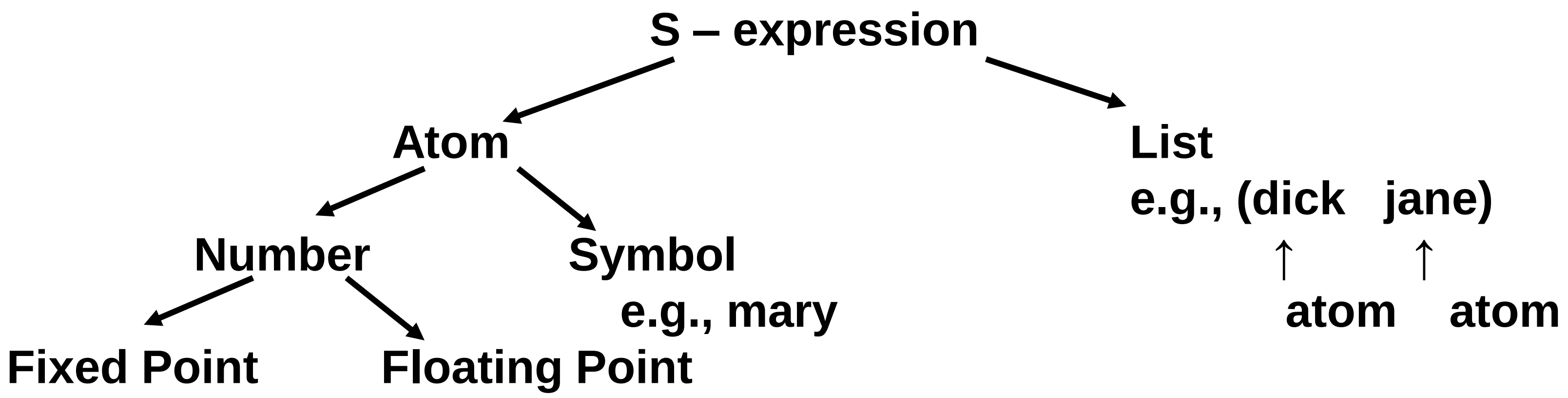
Lisp and Prolog are the two major AI languages today

GNU CLISP – an ANSI Common Lisp

www.clisp.org UNIX and MS windows versions are available.

● **Basic Features of Lisp**

o Lisp has essentially one data type, called S–expressions (short for "Symbolic expression").



§ Lisp

- o A Lisp program is an ordered set of lists, i.e., in Lisp, programs and data have the same form. This means that programs can be manipulated by programs as if they are data.

- o Many similarities between Lisp and Prolog.
  Symbolic.
  Not convenient for numeric computation.
  Programs and data have the same form.
  List is important data structure.
  Extensive use of recursion
  It turns out that many AI problems are naturally represented in Lisp and Prolog:
    Symbolic Algebra
    Natural language processing
    Knowledge representation

● **Basic Arithmetic Operations**

```
                                        Lisp's Response
( +   8   3)                 →                  11
```

In the above 8 and 3 are arguments.

+ is an operator or more generally a <u>function</u>.

Lisp is called a <u>functional language</u> since everything gets done by executing a function.

```
(+   5    3    4)             →                 12
```

Any number of arguments. Similarly,

```
(-    8   3)                  →                 5
 (- 8    3   4 )    =    8  -  3  -  4    →      1
(*  4   3    2)               →                 24
(/    48   8   3 )            →                 2
(1+    8)                     →                 9
```

No space between 1 and + or -.

```
(1-     8)                    →                 7
```

Only one argument for 1+ and 1-.

- **Symbols**

e.g x is a symbol.

```
(setq  x  5)             →    5        Assigns value 5 to x.
                                       x = 5 in C, Java.
      setq does not evaluate its first argument.

x                        →    5

(+  x  8)                →   13

(setq  y  (*  3  4))     →   12
```