Programming for Logic

Dr Buddhitha Hettige

Department of Knowledge Engineering and Communication
Faculty of Computing
University of Sri Jayewardenepura

Overview

- Logic Programming using Prolog
- AI applications of Prolog
  - Monkey Banana problem
  - 8 Queans problem
  - Natural Language processing
  - Expert Systems
- Prolog interface for Application development
  - NLP/Gamming
  - Searching
  - Machine Translation

PROLOG

Logic Programming

- Logic Programming uses **Predicate Logic** as the Building block for programming
- It also executes a program exactly the same manner as how the resolution works in predicate logic
- **Prolog** is well known as a logic programming language
  - High-level interactive language

Programming languages & PROLOG

- Programming languages are of two kinds:
  - **Procedural** (BASIC, ForTran, C++, Pascal, Java);
  - **Declarative** (LISP, Prolog, ML).
- In procedural programming, we tell the computer **how** to solve a problem.
- In declarative programming, we tell the computer **what** problem we want solved.
- (However, in Prolog, we are often forced to give clues as to the solution method).

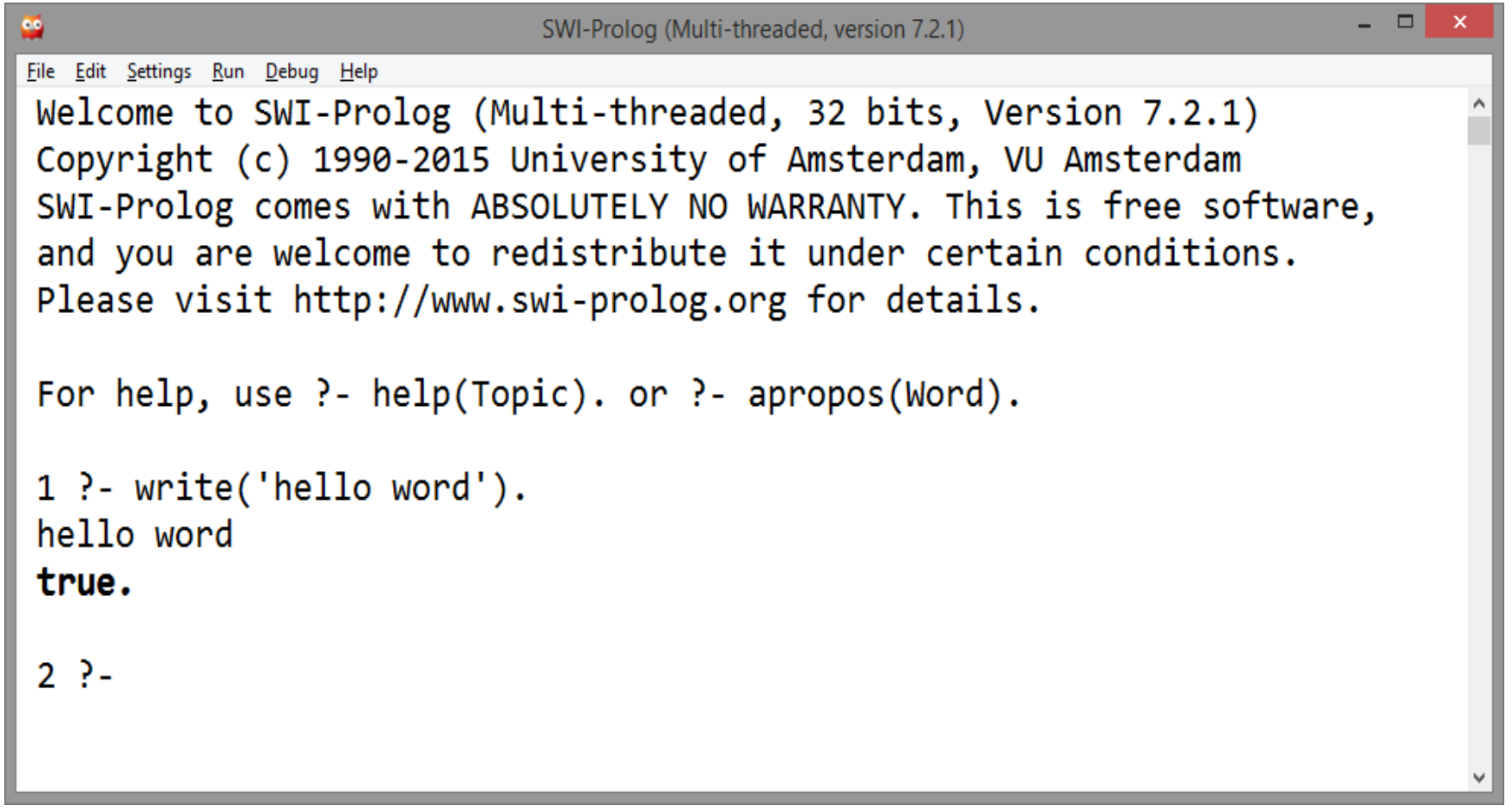
Prolog contd.

- Good at
  - Grammars and Language processing,
  - Knowledge representation and reasoning,
  - Unification,
  - Pattern matching,
  - Planning and Search.
    - i.e. Prolog is good at Symbolic AI.
- Poor at:
  - Repetitive number crunching,
  - Representing complex data structures,
  - Input/Output (interfaces).

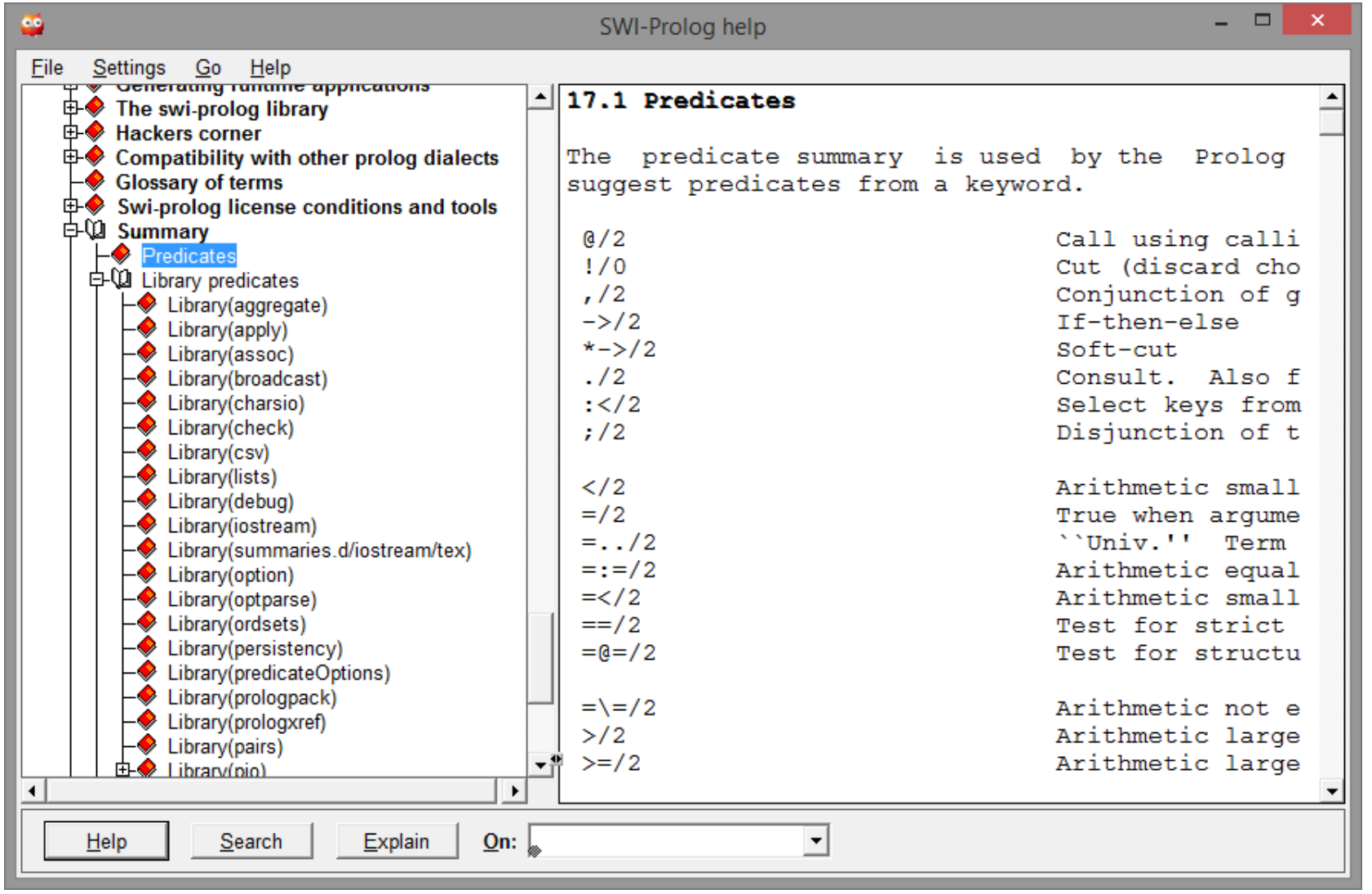
Prolog conventions

- Prolog uses special type of predicates called Horn clauses
  - One clause in the conclusion
  - No negations in the conclusion
- Prolog uses upper case letters for variables, and lower case letters for constants
- Rules are written in the following format
  - Conclusion :- Conditions
  - Read as Conclusion if Conditions

swi-prolog



Use Prolog Help



Syntax

- four kinds of terms in Prolog
  - atoms, numbers, variables, and complex terms (or structures)

Atom

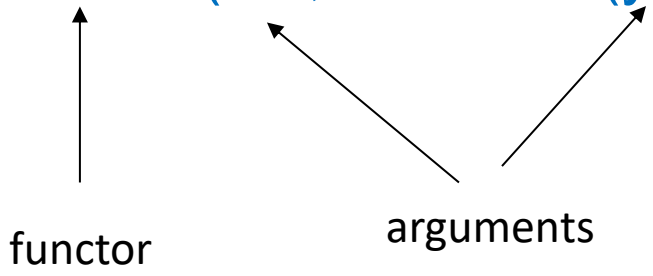
- A string of characters made up of upper-case letters, lower-case letters, digits, and the underscore character, that begins with a lower-case letter
  - saman, seelawathi, ruwan_silva
- An arbitrary sequence of character enclosed in single quotes.
  - ‘Saman Kumara’
- A string of special characters.
  - @= and ====> and ;

Numbers & Variables

- Numbers
  - Prolog implementations do support floating point numbers or floats
  - integers (that is: ... -2, -1, 0, 1, 2, 3, ...) are useful for such tasks as counting the elements of a list
- Variables
  - A variable is a string of upper-case letters, lower-case letters, digits and underscore characters that starts *either* with an upper-case letter *or* with underscore
  - Example
    - X, Y, Variable, _tag, X_526, and List, List24, _head, Tail, _input and Output

Structures

- Objects with several components
  - E.g. animal(cat, domestic(yes)).



- Prolog matches two Terms

Facts, Rules, and Queries

- Only three basic constructs in Prolog:
  - Facts
  - Rules
  - Queries
- A collection of facts and rules is called a

  knowledge base
- **Queries** asking questions about the information stored in the knowledge base

Learning Prolog

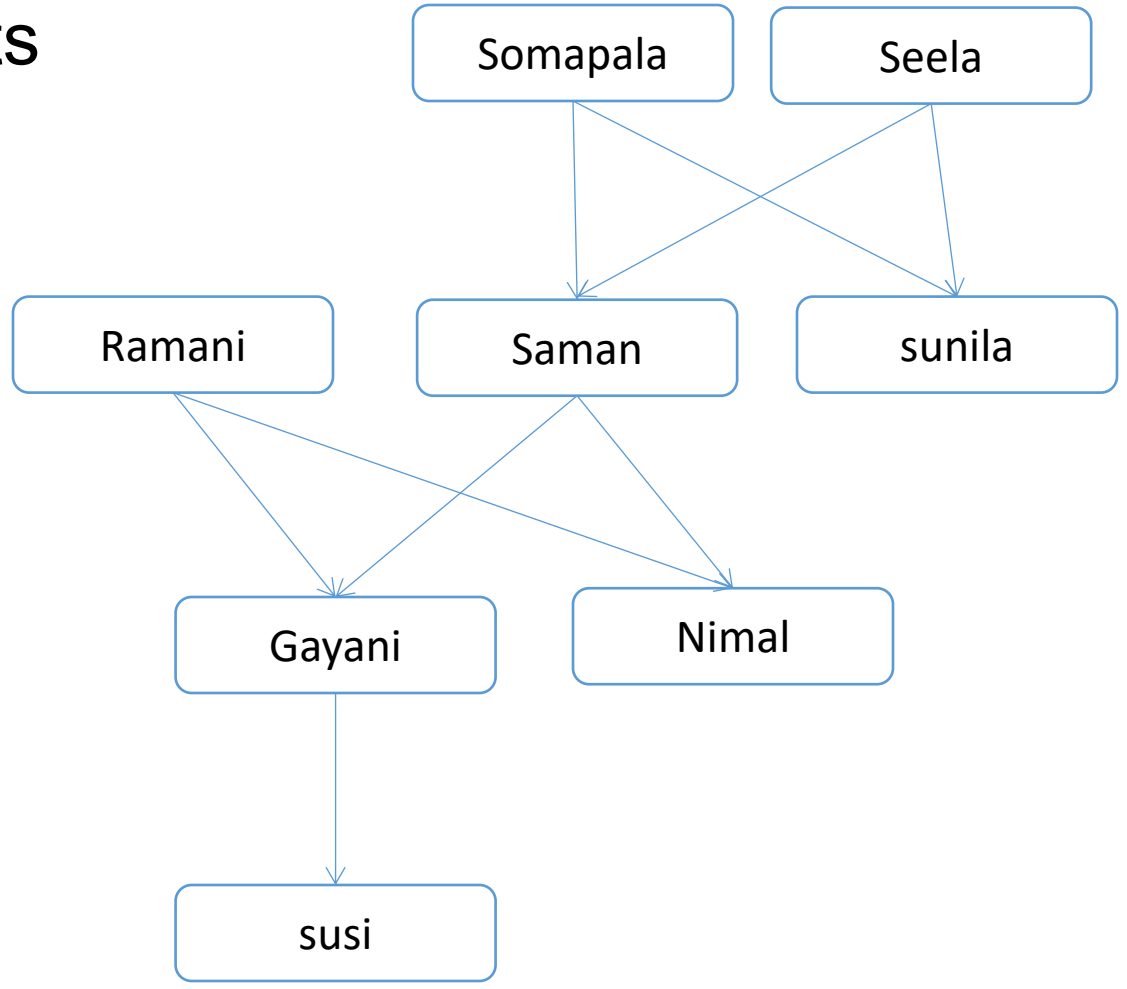
- Writing facts
  - animal(cat).
- Querying
  - animal(X).
- Rules
  - nothuman(X) :- animal(X).
  - mother(X,Y) :- parent(X,Y),female(X).

Facts

- Format
  - PredicateName(Data1, Data2, Data3).
- Example
  - Saman, Somapala are male persons
    - male(kamal).
    - male(somapala).
  - Somapala is father of saman
    - father(somapala, kamal).

Example

- Complete the following facts
- male/1
- female/1
- parent/2



Example

- male(saman).
- male(somapala).
- …
- female(seela).
- female(susi).
- …
- parent(somapala, saman).
- parent(susi, saman) .
- student(saman, saman@gmail.com, 23, ‘Panadura’)…

Rules

- Rules state information that is *conditionally* true of the domain of interest.
- Format:
  precateName(Arguments) :- facts
- Example

:-print (‘Hi’).

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

Example contd.

- Using above facts (male/1, female/1, parent/2) complete the following rule set
  - mother/2
  - father/2
  - grandmother/2
  - grandfather/2
  - brother/2
  - sister/2
  - child/2

Queries

- Queries are used to ask questions
- Example
  - male(X).
- Prolog working its way throu to bottom, trying to match (or unify) the expression male(X)
  - male(seela)

```
2 ?- male(X).
X = somapala .
```

```
3 ?- male(seela).
false.
```

Queries

- Rule: mother(X,Y):- female(X),parent(X,Y).
- Query : mother(X,Y).

```
4 ?- trace.
true.

[trace] 4 ?- mother(X,Y).
   Call: (7) mother(_G1631, _G1632) ? creep
   Call: (8) female(_G1631) ? creep
   Exit: (8) female(seela) ? creep
   Call: (8) parent(seela, _G1632) ? creep
   Exit: (8) parent(seela, kamal) ? creep
   Exit: (7) mother(seela, kamal) ? creep
X = seela,
Y = kamal .
```

Matching

- Two terms match, if they are equal or if they contain variables that can be instantiated in such a way that the resulting terms are equal.
- Example
  - 1 = 1
  - Ruwan = ruwan
  - saman = ‘saman’
  - male(X) = X.

```
[trace] 12 ?- 1 = 1.
true.

[trace] 13 ?- Saman = saman.
Saman = saman.

[trace] 14 ?- saman = 'saman'.
true.

[trace] 15 ?- male(X) = X.
X = male(X).
```

Proof search

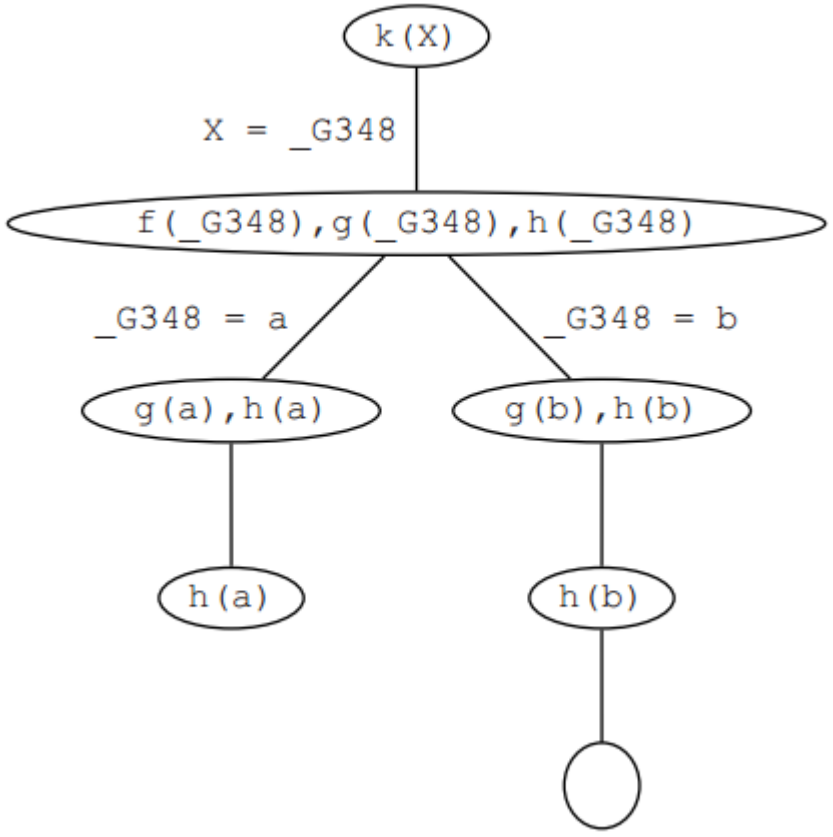
- How Prolog actually searches a knowledge base to see if a query is satisfied

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X) :- f(X),g(X),h(X).
```

- Suppose we then pose the query

```
k(X).
```

Proof Search contd.




```
[trace] 5 ?- k(X).
   Call: (7) k(_G1589) ? creep
   Call: (8) f(_G1589) ? creep
   Exit: (8) f(a) ? creep
   Call: (8) g(a) ? creep
   Exit: (8) g(a) ? creep
   Call: (8) h(a) ? creep
   Fail: (8) h(a) ? creep
   Redo: (8) f(_G1589) ? creep
   Exit: (8) f(b) ? creep
   Call: (8) g(b) ? creep
   Exit: (8) g(b) ? creep
   Call: (8) h(b) ? creep
   Exit: (8) h(b) ? creep
   Exit: (7) k(b) ? creep
X = b.
```

Recursion

- a predicate is recursively defined if one or more rules in its definition refers to itself.
- Example

```
is_digesting(X,Y) :- just_ate(X,Y).
is_digesting(X,Y) :- just_ate(X,Z), is_digesting(Z,Y)
```

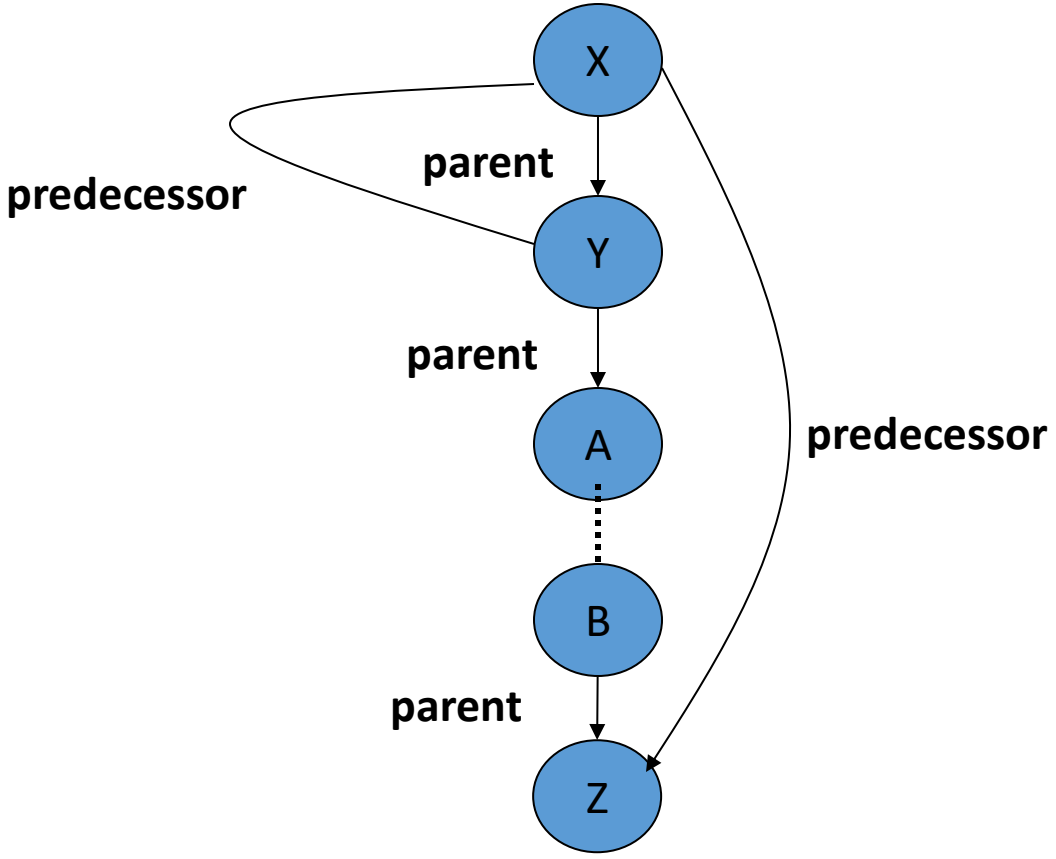
- recursive rule bundles up all this information into just three lines of code.
- Recursive rules are really important

Example

```
parent(asoka, praveen).
parent(malika,asoka).
parent(simon, malika).

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

Recursion using Prolog



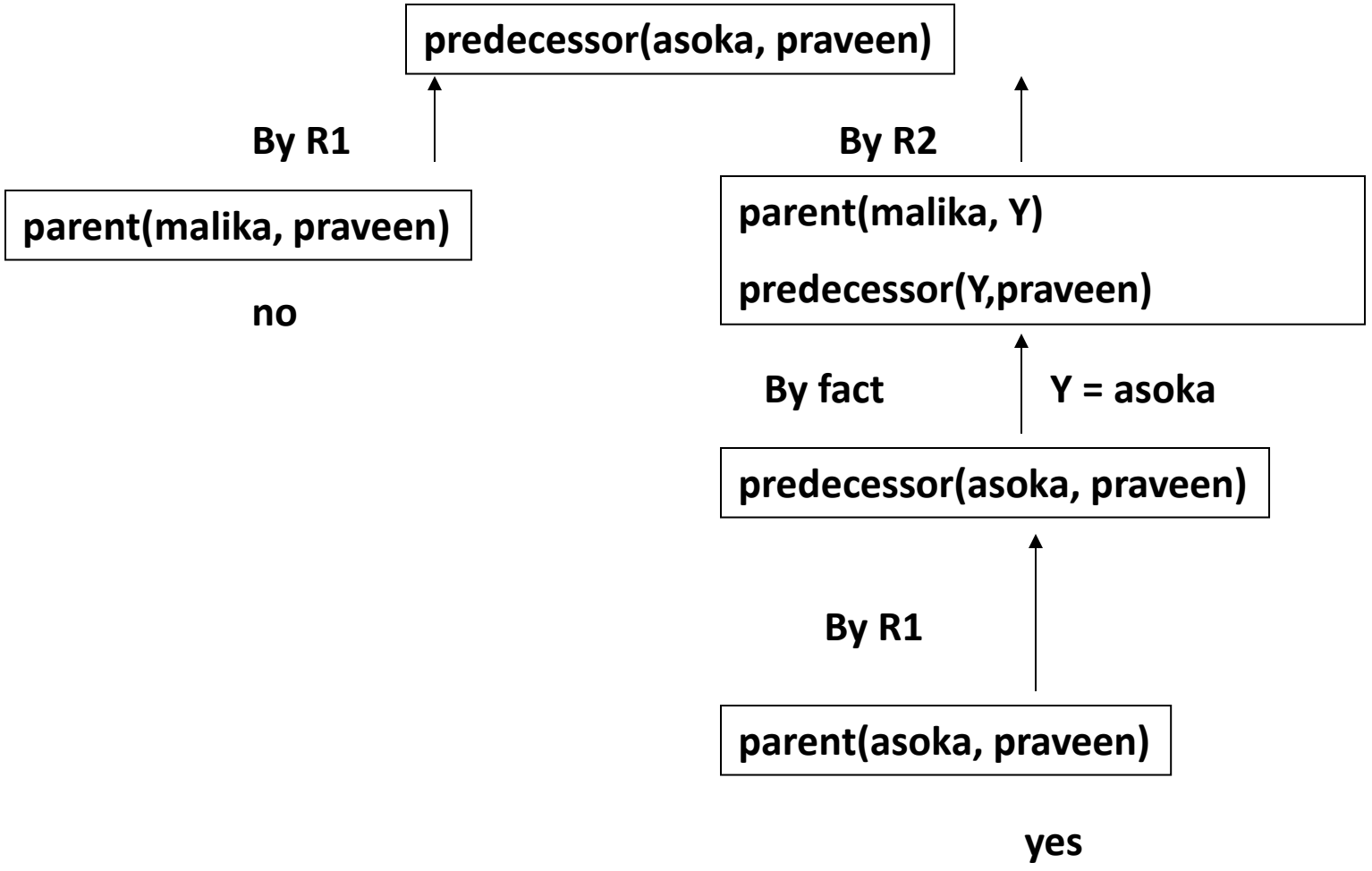
How Prolog answer queries

- Prolog considers a query as a **goal** to be satisfied by the program
- For this purpose, the goal will be matched with **facts** or **heads** of rules in the program
- When matching with a head of a rule Prolog substitute for variables and generate sub goals. Then Prolog tries to satisfy sub goals one by one.

Another query

- ?- predecessor(malika,praveen).
- This goal is matched with R1, instantiate X=malika and Y=praveen, then returns parent(malika,praveen)
- New goal matches with facts and fails
- Then prolog **backtracks** to find another clauses, with what the original goal can be matched
- Prolog find R2, and proceed as shown

Derivation


predecessor(asoka, praveen)
By R1
By R2
parent(malika, praveen)
no
parent(malika, Y)
predecessor(Y,praveen)
By fact
Y = asoka
predecessor(asoka, praveen)
By R1
parent(asoka, praveen)
yes

Exercise

- Draw derivation trees for the following quires
  - predecessor(simon,praveen).
  - predecessor(malika,X).
  - predecessor(asoka,simon).

Importance of order of clauses

- Procedural and declarative meaning
- Consider different versions of **predecessor/2, with the same declarative but different procedural meaning**

```
pre1(X,Y):-parent(X,Y).
pre1(X,Y):-parent(X,Z),pre1(Z,Y).

pre2(X,Y):-parent(X,Z),pre2(Z,Y).
pre2(X,Y):-parent(X,Y).

pre3(X,Y):-parent(X,Y).
pre3(X,Y):- pre3(X,Z), parent(Z,Y).

pre4(X,Y):-pre4(X,Z), parent(Z,Y).
pre2(X,Y):-parent(X,Y).
```

Lists

- A useful data structure in Prolog
- List is the basis for powerful Prolog programs
- a finite sequence of elements
  - [mia, vincent, jules, yolanda]
    [mia, robber(honey_bunny), X, 2, mia]
    []
- List = [Head |Tail]
  - Head – the first element
  - Tail – list of other elements

List Operations

- **Print list**

```
printList([]).
printList([H|T]) :- write(H),nl,printList(T).
```

- **Print list in reverse order**

```
printList([]).
printList([H|T]) :- printList(T), write(H),nl.
```

- **Member in a list**

```
member(X,[X|T]).
member(X,[H|T]) :- member(X,T).
```

List Operations

- **Print length of a list**

```
Length([],0).
length([H|T],S) :- length (T,S1),S is 1 + S1.
```

- **Print summation of the number list**

```
sum([],0).
sum([H|T],S) :- sum(T,S1),S is H + S1.
```

- **Append two lists**

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

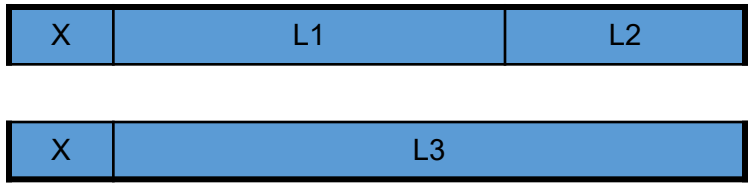
List Operations

- **Delete an item in a list**

```
delItem(X,[X|T],T).
delItem(X, [ Y|T], [Y|T]) :- delItem(X,T,T).
```

- **Concatenate two list**

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



If (Selection)

Format

(condition -> then ; else).

Create a predicate printgrade(M) to print the grade of a given mark

M < 30 F

M < 45 & M >=30 S

M < 55 & M >- 45 C

M < 70 & M >= 55 B

M >=70 A

PROLOG Data bases

Example

- Student:- store name, age , sex, address, email address and student ID
  - student(‘Saman kumara, 26, male, ‘colombo 3’, ‘saman@yahoo.com’, ‘stu2201’).
- course :- course code name and unit
  - course(csu2280, ‘Deductive Reasoning and PROLOG for AI’, 30).
- examResult :- coursecode, Student ID, Marks
  - examResult(csu2280, stu2201, 56).

Add new Record

- **using assert/1**
- The following example shows a sample rule to add a new Student

```
AddstuResult :-
        write('Enter Course Code'),read(C),
        write('Enter Student ID'),read(ID),
        write('Enter Marks '),read(M),
        assert(examResult(C,ID,M)).
```

- Create rules to add a new Student and a new course

Prolog Database

- **Delete existing Records**
- The following example shows a sample rule to delete exam result;

```
deleteResult :-
        write('Enter Course Code'),read(C),
        write('Enter Student ID'),read(ID),
        retract(examResult(C,ID,_)).
```

Prolog Database

- Update records
- The following example shows sample rule for update existing exam result

```
updateResult :-
                write('Enter Course Code'),read(C),
                write('Enter Student ID'),read(ID),
                examResult(C,ID,OM),
                retract(examResult(C,ID,OM)),
                write('Enter Marks '),read(NM),
                assert(examResult(C,ID,NM)).
```

Prolog Database

- **Select Records**
- Select Student ID list how has followed csu2280 course

```
printList([]).
printList([H|T]) :- write(H),nl,printList(T).


stuList(Cou) :- Setof(ID, ^Mark examResult(Cou,ID,Mark), List),
                printLst(List).

stuList(Cou) :- bagof(ID, ^Mark examResult(Cou,ID,Mark), List),
                printLst(List).
```

Prolog Database

- **To Checks whether a student is already in the database**

```
checktudent  :-   write('Enter Name '), read(N),
                  (  student(N,_,_,_,_,_)
                           ->
                  write('Student Found')
                           ;
                  write(' NO Student')
                  ).
```

Arithmetic

Arithmetic examples	Prolog Notation
$6+2=8$	`8 is 6+2.`
$6*2=12$	`12 is 6*2.`
$6-2=4$	`4 is 6-2.`
$6-8=-2$	`-2 is 6-8.`
$6\div 2=3$	`3 is 6/2.`
$7\div 2=3$	`3 is 7/2.`
1 is the remainder when 7 is divided by 2	`1 is mod(7,2).`

Comparing integers

Arithmetic examples	Prolog Notation
$x < y$	`X < Y.`
$x \leq y$	`X =< Y.`
$x = y$	`X =:= Y.`
$x \neq y$	`X =\= Y.`
$x \geq y$	`X >= Y`
$x > y$	`X > Y`

Preventing backtracking

- Consider the function
  - If X<3 then Y=0
  - If X>=3 and X<6 then Y=4
  - If 6=<X then Y=4
- These can be written in Prolog
  - R1: f(X,0):-X<3.
  - R2: f(X,2):- 3=<X, X<6.
  - R3: f(X,4):- 6=<X.

!, not, fail Operators

- Example

```
not(p):-(P, ! , fail
            ;
          true
        ).
```

Negation as failure

```
not(p):-P,!,fail
        ;
       true.
```

- Note the importance of !

Example

- File Handling
- Java interfaces for SWI –PROLOG
- Artificial Intelligent Sample programs
  - Path finder
  - Farmer, Wolf, Goat and Cabbage Problem
  - 8-Quens problems
  - Tic-Tac-Toe in Prolog
- Download
  - https://budditha.files.wordpress.com/2016/09/prolog_guide.pdf