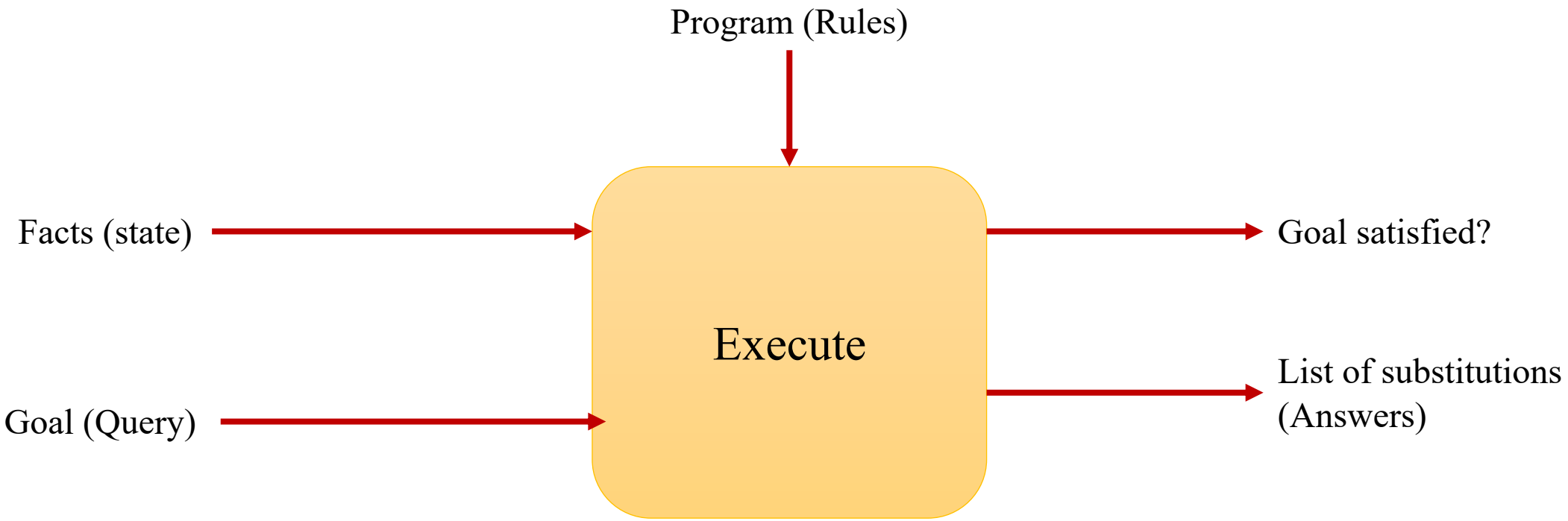
Logic Programming: Prolog

06/02/2024

Koustav Rudra

Prolog Computation Model



Prolog Rules

grand_advisor(X,Z) :- advisor(X,Y), advisor(Y,Z)

head
(consequent)

body
(antecedent)

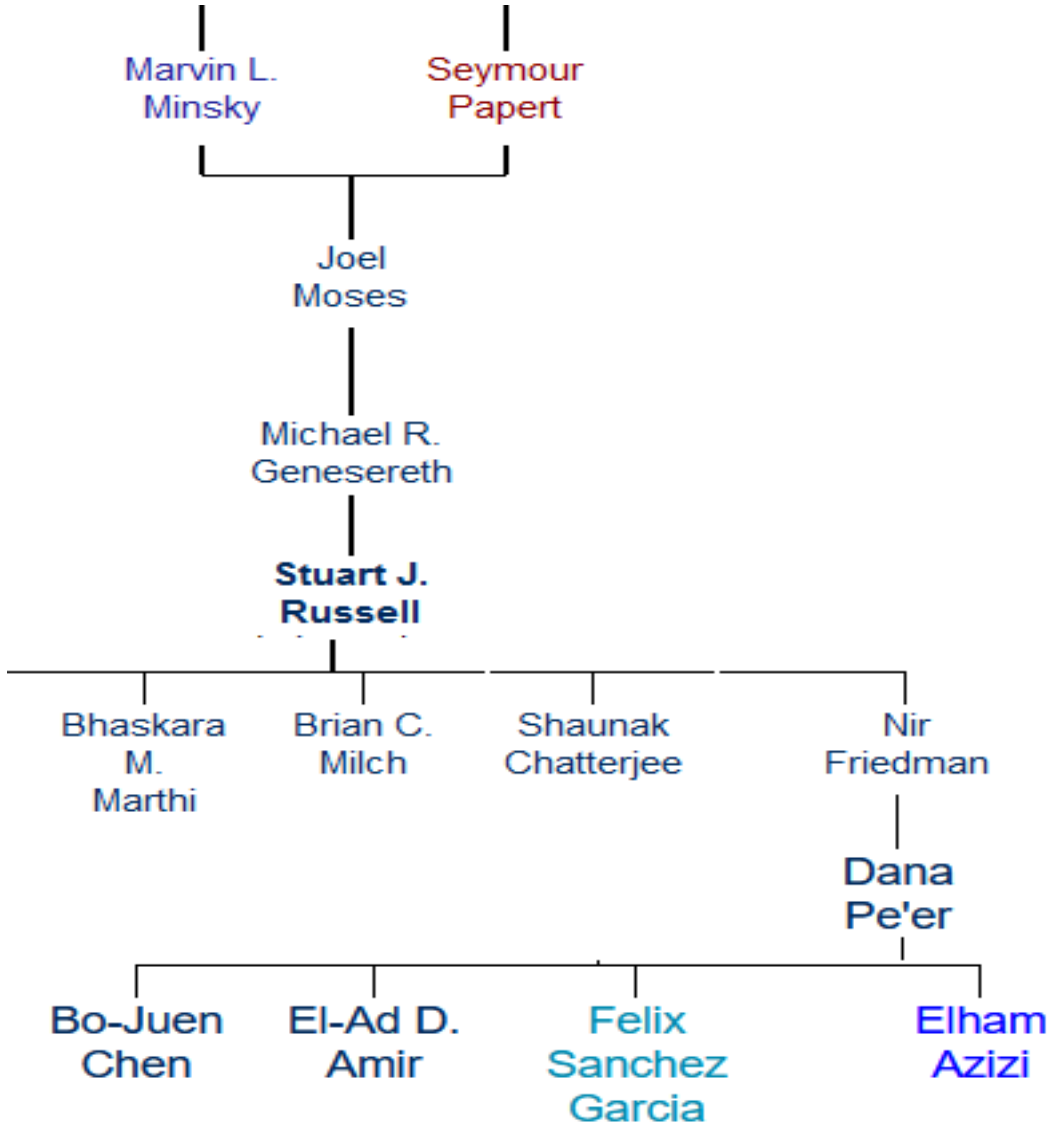
- $\forall_{xz}\exists_{y}$ advisor(X,Y) $\wedge$ advisor(Y,Z) $\rightarrow$ grand_advisor(X,Z)
- IF there is a Y such that X is advisor of Y AND Y is advisor of Z THEN X is a grand advisor of Z
- Prolog rules are Horn Clauses:
  - $(P_{11} \vee P_{12} \vee \ldots \vee P_{1m}) \wedge \ldots \wedge (P_{n1} \vee P_{n2} \vee \ldots \vee P_{nr}) \rightarrow Q$
  - Q:- $P_{11}; P_{12}; \ldots ; P_{1m}, \ldots \ldots, P_{n1}; P_{n2}; \ldots; P_{nr}$

Prolog Rules: Recursion

- ancestor(X, Z) :- advisor(X, Z)
- ancestor(X, Z) :- advisor(X, Y), advisor(Y, Z)
- ancestor(X, Z) :- advisor(X, Y1), advisor(Y1, Y2), advisor(Y2, Z)

- ancestor(X, Z) :- advisor(X, Z)
- ancestor(X, Z) :- advisor(X, Y), ancestor(Y, Z)
- X is an ancestor of Z if X is an advisor of Y AND Y is an ancestor of Z

Prolog Facts



- advisor(minsky, moses).
- advisor(papert, moses).
- advisor(moses, genesereth).
- advisor(genesereth, russell).
- advisor(russell, bhaskara).
- advisor(russell, milch).
- advisor(russell, shaunak).
- advisor(russell, friedman).
- advisor(friedman, dana).
- advisor(dana, felix).
- advisor(dana, chen).
- advisor(dana, amir).
- advisor(dana, azizi).

- male(felix).
- female(dana).

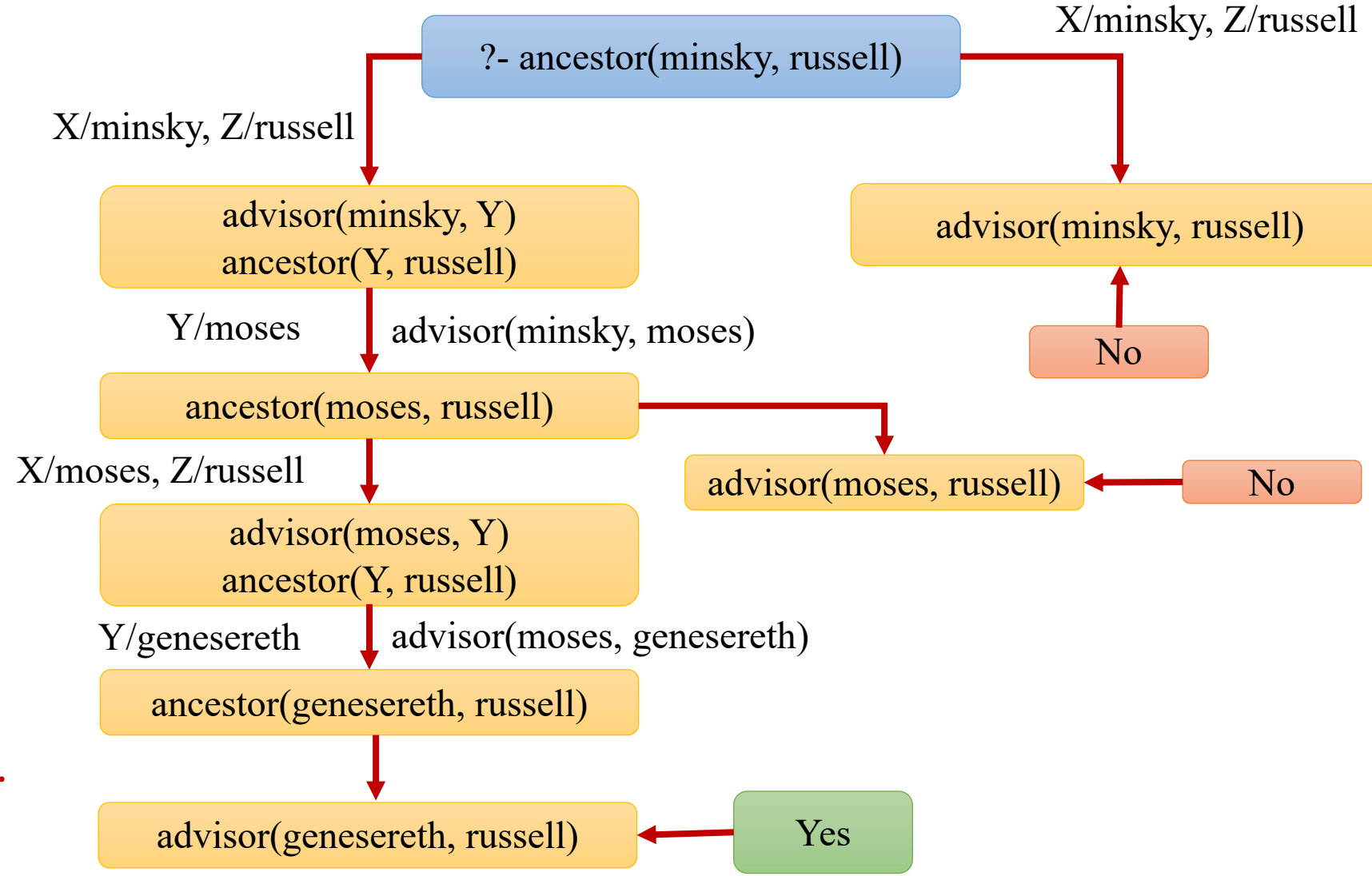
How Prolog answers?

ancestor(X, Z) :- advisor(X, Z)

ancestor(X, Z) :- advisor(X, Y),
ancestor(Y, Z)

?- ancestor(minsky, russell)

advisor(minsky, moses).
advisor(papert, moses).
advisor(moses, genesereth).
advisor(genesereth, russell).



Reordering of Clauses

Original

```
ancestor1(X,Z) :-
advisor(X, Z).

ancestor1(X,Z) :-
advisor(X, Y),
ancestor1(Y, Z).
```

Clause swap

```
ancestor2(X,Z) :-
advisor(X, Y),
ancestor2(Y, Z).

ancestor2(X,Z) :-
advisor(X, Z).
```

Goal swap

```
ancestor3(X,Z) :-
advisor(X, Z).

ancestor3(X,Z) :-
ancestor3(Y, Z),
advisor(X, Y).
```

Clause and Goal swap

```
ancestor4(X,Z) :-
ancestor4(Y, Z),
advisor(X, Y).

ancestor4(X,Z) :-
advisor(X, Z).
```

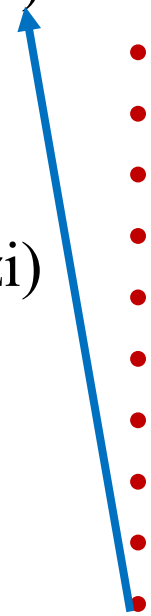
Reordering of Clauses: Original

```
ancestor1(X,Z) :-
advisor(X, Z).

ancestor1(X,Z) :-
advisor(X, Y),
ancestor1(Y, Z).
```

Original

- Call ancestor1(dana, azizi)
- Call advisor(dana, azizi)
- Exit advisor(dana, azizi)
- Exit ancestor1(dana, azizi)



- advisor(minsky, moses).
- advisor(papert, moses).
- advisor(moses, genesereth).
- advisor(genesereth, russell).
- advisor(russell, bhaskara).
- advisor(russell, milch).
- advisor(russell, shaunak).
- advisor(russell, friedman).
- advisor(friedman, dana).
- advisor(dana, felix).
- advisor(dana, chen).
- advisor(dana, amir).
- advisor(dana, azizi).

- male(felix).
- female(dana).

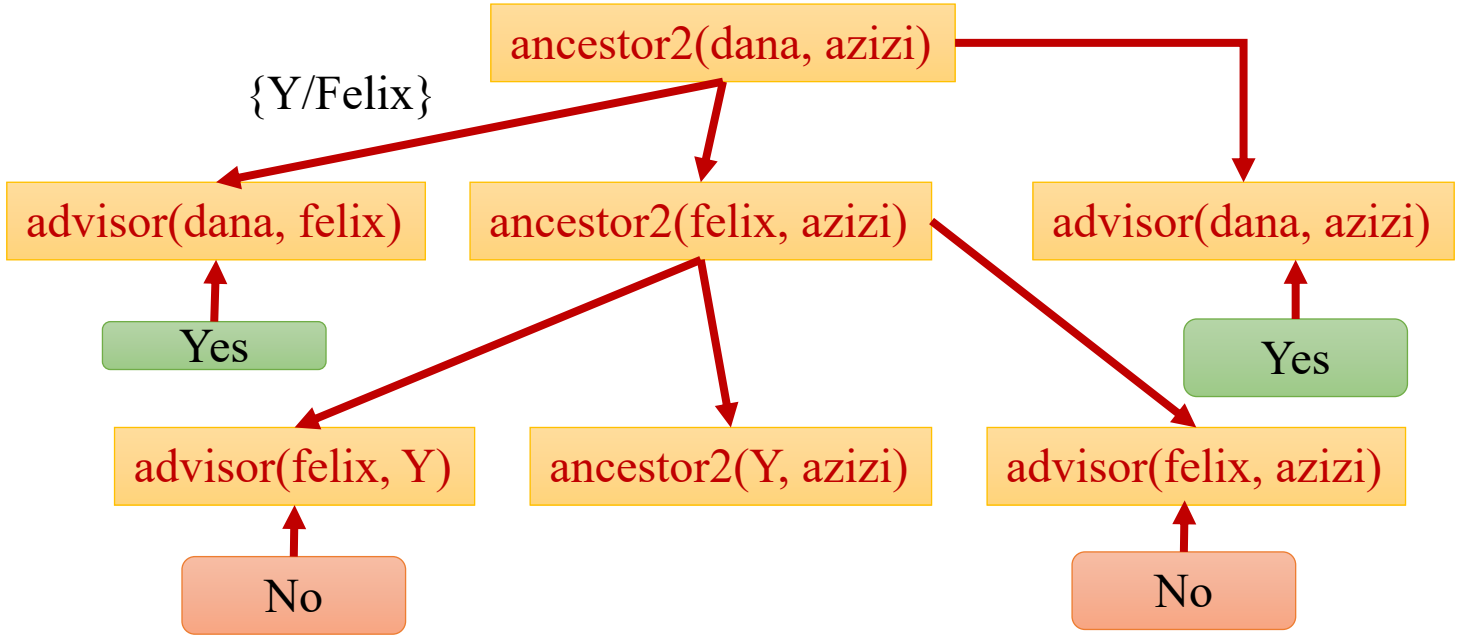
Reordering of Clauses: Clause Swap

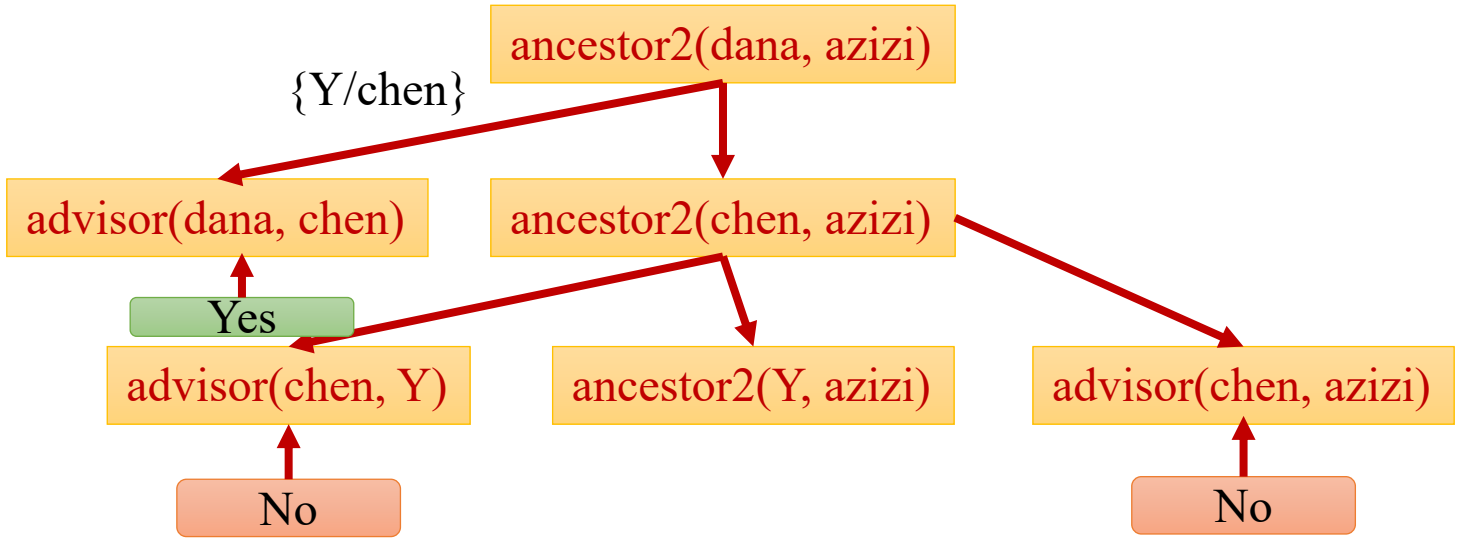
Clause swap

```
ancestor2(X,Z) :-
advisor(X, Y),
ancestor2(Y, Z).

ancestor2(X,Z) :-
advisor(X, Z).
```

- advisor(dana, felix).
- advisor(dana, chen).
- advisor(dana, amir).
- advisor(dana, azizi).





Reordering of Clauses: Goal Swap

```
ancestor3(X,Z) :-
advisor(X, Z).

ancestor3(X,Z) :-
ancestor3(Y, Z),
advisor(X, Y).
```

```
ancestor4(X,Z) :-
ancestor4(Y, Z),
advisor(X, Y).

ancestor4(X,Z) :-
advisor(X, Z).
```

- `?- ancestor3(bhaskara, felix)`
- Infinite Loop

Takeaways from Ordering

- Try simplest idea first (practical heuristics in problem solving)
  - ancestor1 being the simplest, ancestor4 being the most complex

- Check your clause ordering to avoid infinite recursion

- Procedural aspect is also important along with declarative

Logic Programming: Prolog

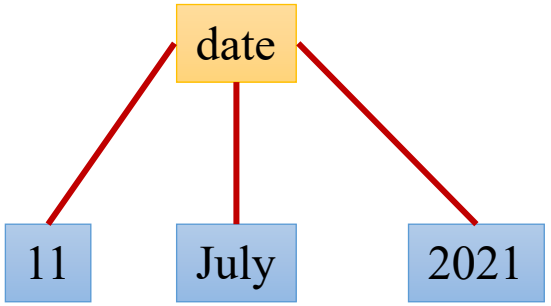
12/02/2024

Koustav Rudra

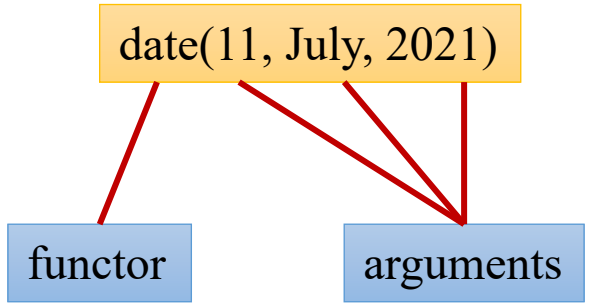
Prolog: Data Objects - Structures

- Structured Data Objects –
  - Structured data objects (or structures) are objects that have multiple components.
  - The components themselves can in turn be structures

  - e.g., date can be viewed as a structure with three components --- day, month, year
  - 11th July 2021 : date(11,July,2021)



Tree Representation



Prolog Representation

Structured Data Objects: Example

- P1: point(1,1)
- P2: point(2,3)
- S: seg(P1,P2): seg(point(1,1),point(2,3))
- T: triangle(point(4,Z),point(6,4),point(7,1))

- Structures can be naturally pictured as trees
- Prolog can be viewed as a language for processing trees

Prolog: Data Structure

- Lists
  - Lists of anything, symbolic lists
- Lists can be written as:
  - [item1, item2, …]
  - [Head|Tail]
    - Head is the first element in the list, remaining is the tail (list)
  - [item1, item2, …|Others]
    - Head consists of item1, followed by the tail which is other items [list]
- [a, b, c] = [a|[b,c]] = [a,b|[c]] = [a,b,c|[]]

- Items can be list as well
  - [[a,b], c, [d, [e,f]]]
  - The head of the above list is list [a,b]

Prolog: Data Structure

- ?- [X|Y] = [a,b|c]
  - X = a, Y = [b|c]
- ?- [X|Y] = [a|[b,c]]
  - X = a, Y = [b,c]
- ?- [X|Y] = [a,b|[c]]
  - X = a, Y = [b,c]
- ?- [X|Y] = [a,b,c|[]]
  - X = a, Y = [b,c]

List Examples: Membership

- member(X,Y) ➔ X is a member of list Y

- member(X,[X|Tail]).
- member(X,[Head|Tail]) :- member(X,Tail).

- a , [[b], [a,b], b]
  - Looking only at first level
  - How to find membership within sub-lists?

List Examples: Concatenation

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

- ?-conc([a], Z, [a,b]).
  - Z = [b]
- ?-conc([a], [b], Z).
  - Z = [a, b]

List Examples: Concatenation

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

- ?- conc([a,b],[c,d],[a,b,c,d])
- ?- conc([b],[c,d],[b,c,d])
- ?- conc([],[c,d],[c,d])

X = a, L1 = [b], L2 = [c,d], L3 = [b,c,d]

X = b, L1 = [], L2 = [c,d], L3 = [c,d]

- ?- conc([a],[b],[a,d])
- ?- conc([],[b],[d])

List Examples: Concatenation

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

- ?- conc([a],y,[a,b])
- ?- conc([],y,[b])
- y = [b]

X = a, L1 = [], L2 = y, L3 = [b]

Y = [b]

Thank You