Prolog

a logic programming language

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Prolog

Variously:

- **logic programming**
- **declarative programming**
- **unification with backtracking**

Roots in “Horn Clauses” of first-order logic

Prolog programs in two parts:

- **Database of “rules”**
  Each has the form “infer conclusion from premises”
  (premises may be empty : facts or axioms)
- **Ask “query” about the database**

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Reasoning with Prolog: Family relationships

Facts:

```
parent(daniel,don).
parent(mary,don).
parent(don,dave).
parent(don,mike).
parent(mike,samantha).
parent(mike,brooke).
```

Queries:

```
?-parent(mike,samantha).
yes
?-parent(brooke,mike).
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Family relationships example (continued)

More queries:

```
?-parent(X,don).
X = daniel
yes
```

You can press semicolon after each solution to force backtracking:

```
?-parent(X,don).
X = daniel;
X = mary;
no
?-parent(don,X).
X = dave;
X = mike;
no
?-parent(X,X).
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Family relationships example (continued)

Rule:

```
grandparent(A,B) :- parent(A,C), parent(C,B).
```

Queries:

```
?- grandparent(don,brooke).
yes
?- grandparent(brooke,don).
no
?- grandparent(X,samantha).
X = don
yes
?- grandparent(mary,X).
X = dave;
X = mike;
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Family relationships example (continued)

More rules:

```
ancestor(A,B) :- parent(A,B).
ancestor(A,B) :- parent(A,C), ancestor(C,B).
```

Queries:

```
?- ancestor(mary,X).
X = don;
X = dave;
X = mike;
X = samantha;
X = brooke;
no
?- ancestor(X,X).
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

List predicate examples

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

?- member(c,[a,b,c,d,e]).
yes
?- member(f,[a,b,c,d,e]).
no
?- member(X,[a,b,c,d,e]).
X = a;
X = b;
X = c;
X = d;
X = e;
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

List predicate examples (continued)

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

?- append([a,b,c],[1,2,3],X).
X = [a, b, c, 1, 2, 3]
yes
?- append([a,b,c],X,[a,b,c,1,2,3]).
X = [1, 2, 3]
yes
?- append(X,[1,2,3],[a,b,c,1,2,3]).
X = [a, b, c]
yes
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

List predicate examples (continued)

```
?- append(X,Y,[a,b,c,d]).
X = []
Y = [a, b, c, d];

X = [a]
Y = [b, c, d];

X = [a, b]
Y = [c, d];

X = [a, b, c]
Y = [d];

X = [a, b, c, d]
Y = [];

no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

List predicate examples (continued)

```
-> listlength([ ],0).
-> listlength([_|T],N) :- listlength(T,M), N is 1+M.

?- listlength([ ],X).
X = 0
yes
?- listlength([a,b,c,d,e,f,g],X).
X = 7
yes
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

List predicate examples (continued)

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

?- reverse([a,b,c,d],X).
X = [d, c, b, a]
yes
?- reverse(X,[1,2,3,4]).
X = [4, 3, 2, 1]
yes
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Numerical examples

```
factorial(0,1).
factorial(N,F) :- N>0, M is N- 1, factorial(M,X), F is N*X.

?- factorial(6,X).
X = 720
yes
?- factorial(X,720).
run-time error: Used comparison > on non-integer term
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Numerical examples (continued)

```
fibonacci(1,1).
fibonacci(2,1).
fibonacci(N,F):- N>1, X is N- 2, Y is N- 1,
                 fibonacci(X,A), fibonacci(Y,B), F is A+B.

?- fibonacci(3,X).
X = 2
yes
?- fibonacci(4,X).
X = 3
yes
?- fibonacci(5,X).
X = 5
yes
?- fibonacci(6,X).
X = 8
yes
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

Numerical examples (continued)

```
fibonacci(N,F):- helper(N,0,1,F).

helper(1,A,B,B).
helper(N,A,B,F):- N>0, X is N- 1, Y is A+B, helper(X,B,Y,F).

?- fibonacci(10,X).
X = 55
yes
?- fibonacci(20,X).
X = 6765
yes
?- fibonacci(30,X).
X = 832040
yes
?- fibonacci(40,X).
X = 102334155
yes
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

More list examples

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

?- delete(X,[a,b,c,d],Y).
X = a
Y = [b, c, d];

X = b
Y = [a, c, d];

X = c
Y = [a, b, d];

X = d
Y = [a, b, c];

no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

More list examples (continued)

```
permute([ ],[ ]).
permute(L,[H|T]) :- delete(H,L,M), permute(M,T).

?- permute([a,b,c],Z).
Z = [a, b, c];
Z = [a, c, b];
Z = [b, a, c];
Z = [b, c, a];
Z = [c, a, b];
Z = [c, b, a];
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

More list examples (continued)

```
allpermutations(L) :- permute(L,Z), print(Z), fail.

?- allpermutations([a,b,c]).
[a, b, c]
[a, c, b]
[b, a, c]
[b, c, a]
[c, a, b]
[c, b, a]
no
```

Here “fail” is a deliberately undefined proposition, so it can never succeed

- **Therefore it always forces backtracking**

©Copyright 2006 Norman Ramsey. All Rights Reserved.

More list examples (continued)

```
ordered([ ]).
ordered([_]).
ordered([X,Y|Z]) :- X=<Y, ordered([Y|Z]).

?- ordered([10,20,30,40]).
yes
?- ordered([10,30,20,40]).
no
```

©Copyright 2006 Norman Ramsey. All Rights Reserved.

More list examples (continued)

```
sort(A,B) :- permute(A,B), ordered(B).

?- sort([20,40,10,30],L).
L = [10, 20, 30, 40]
yes
```

Very inefficient O(n!)-time sorting algorithm, where n = length of list

©Copyright 2006 Norman Ramsey. All Rights Reserved.