Logic Programming

In this lecture you get a taste of logic programming, via the Prolog language.

As examples we use:

- Search (in non-deterministic FSMs)
- Grammar programming
- Constraint programming

Prolog as a Form of Natural Deduction

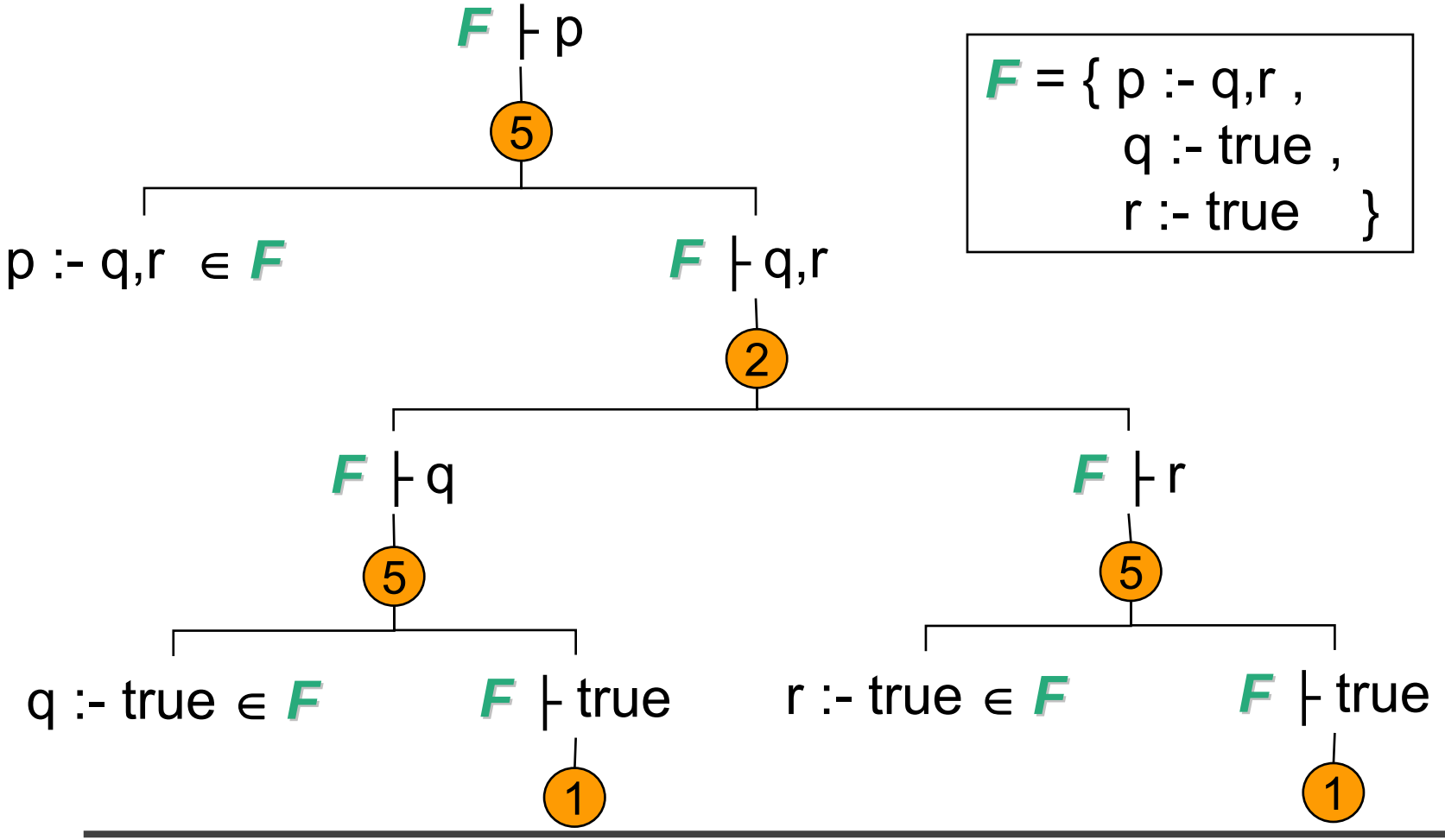
Proof	Sub-proofs		
$\Gamma \vdash \text{true}$			1
$\Gamma \vdash A , B$	$\Gamma \vdash A$	$\Gamma \vdash B$	2
$\Gamma \vdash A ; B$	$\Gamma \vdash A$		3
$\Gamma \vdash A ; B$	$\Gamma \vdash B$		4
$\Gamma \vdash A$	$A :- B \in \Gamma$	$\Gamma \vdash B$	5

Only point of choice

plus negation as failure:

$\Gamma \vdash \text{not}(A)$	$\Gamma \nvdash A$	6

A Prolog Natural Deduction Proof



F = { p :- q,r ,
q :- true ,
r :- true }

Defining FSM Accept in Logic

(from earlier lecture)

FSM: model(Q,Σ,S_0,F,δ)

String: σ

accept(σ) $\leftarrow$
    model(Q, Σ, S_0, F, δ) and
    trace(S_0, σ, F, δ)

trace(S, [], F, δ) $\leftarrow$ S $\in$ F

trace(S, [X|R], F, δ) $\leftarrow$
    (S,X,S1) $\in$ δ and
    trace(S1,R, F, δ)

[] is the empty sequence
[X|R] separates first element, X,
from rest of sequence, R.

Defining FSM Accept in Prolog

E ∈ Set substituted by member(E, Set).

```
accept(Seq) :-
    model(Q, I, S0, F, D),
    trace(S0, Seq, F, D).
```

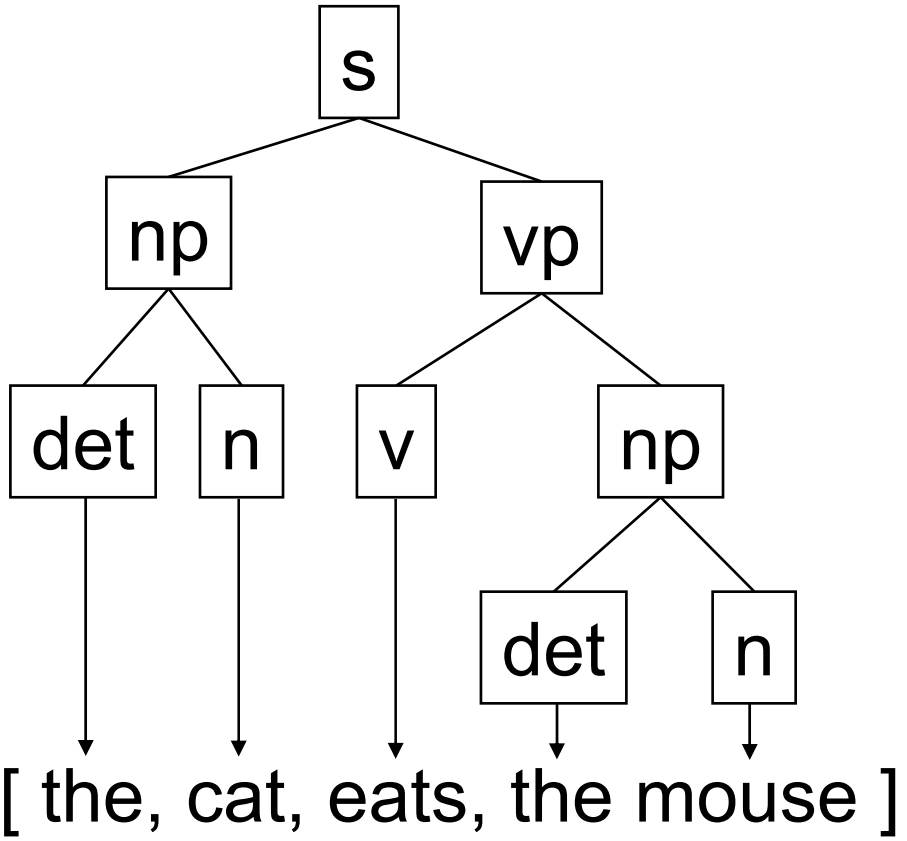
```
trace(S, [], F, D) :- member(S, F).

trace(S, [X|R], F, D) :-
    member((S,X,S1), D),
    trace(S1,R, F, D)
```

Grammar Example

```
s --> np, vp.
np --> n.
np --> det, n.
vp --> v, np.

n --> [cat].
n --> [mouse].
v --> [eats].
det --> [the].
```



DCG Example as Translated to Clauses

```
s --> np, vp.
np --> n.
np --> det, n.
vp --> v, np.

n --> [cat].
n --> [mouse].
v --> [eats].
det --> [the].
```

```
s(Si,Sr) :- np(Si,Sn), vp(Sn,Sr).
np(Si,Sr) :- n(Si,Sr).
np(Si,Sr) :- det(Si,Sn), n(Sn,Sr).
vp(Si,Sr) :- v(Si,Sn), np(Sn,Sr).

n(Si,Sr) :- 'C'(Si, cat, Sr).
n(Si,Sr) :- 'C'(Si, mouse, Sr).
v(Si,Sr) :- 'C'(Si, eats, Sr).
det(Si,Sr) :- 'C'(Si, the, Sr).
```

```
'C'([H | T], H, T).
```

An Arithmetic Puzzle


```
  S E N D
+ M O R E
---------
M O N E Y
```

Replace variables with numbers (0,…,9) such that each variable is different and the sum is correct.

```
|?- solve_money([S,E,N,D], [M,O,R,E], [M,O,N,E,Y]).
```

S = 2, E = 8, N = 1, D = 7,
M = 0, O = 3, R = 6, Y = 5

…and other combinations

```
  2 8 1 7
+ 0 3 6 8
---------
0 3 1 8 5
```

CLP Solution

```
add(A, B, C) :-   add(A, B, C, 0).

add([], [], [], 0).
add([A|As], [B|Bs], [C|Cs], Carry) :-
    A in 0..9,
    B in 0..9,
    C #= (A + B + NextCarry) mod 10,
    Carry #= (A + B + NextCarry) / 10,
    add(As, Bs, Cs, NextCarry).

solve_money([S,E,N,D], [M,O,R,E], [M,O,N,E,Y]) :-
    add([0,S,E,N,D], [0,M,O,R,E], [M,O,N,E,Y]),
    all_different([S,E,N,D,M,O,R,Y]),
    labeling([], [S,E,N,D,M,O,R,Y]).
```

Sudoku : Example Problem

		1					4	
9	5						1	7
		7	8			5	6	
4				9	7		5	
				8				
	3		5	4				1
	6	2			4	1		
1	9						8	6
	7					9		

Complete the table such that the numbers in each box, row and column are different, using the integers 1 to 9.

Sudoku : Constraint Solver

```
go(Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9) :-
    Vars = [A1,A2,A3,A4,A5,A6,A7,A8,A9,
            B1,B2,B3,B4,B5,B6,B7,B8,B9,
            C1,C2,C3,C4,C5,C6,C7,C8,C9,
            D1,D2,D3,D4,D5,D6,D7,D8,D9,
            E1,E2,E3,E4,E5,E6,E7,E8,E9,
            F1,F2,F3,F4,F5,F6,F7,F8,F9,
            G1,G2,G3,G4,G5,G6,G7,G8,G9,
            H1,H2,H3,H4,H5,H6,H7,H8,H9,
            I1, 2,I3,I4,I5,I6,I7,I8,I9],
    domain(Vars, 1, 9),
    Row1 = [A1,A2,A3,A4,A5,A6,A7,A8,A9],…,
    Row9 = [I1,I2,I3,I4,I5,I6,I7,I8,I9],
    Col1 = [A1,B1,C1,D1,E1,F1,G1,H1,I1],…,
    Col9 = [A9,B9,C9,D9,E9,F9,G9,H9,I9],
    Box1 = [A1,A2,A3,B1,B2,B3,C1,C2,C3],…,
    Box9 = [G7,G8,G9,H7,H8,H9,I7,I8,I9],
    all_different(Row1),  …, all_different(Row9),
    all_different(Col1), …, all_different(Col9),
    all_different(Box1), …, all_different(Box9),
    constrain_puzzle(Vars),
    labeling([ff],Vars).
```

```
constrain_puzzle(Vars) :-
   Vars = [ _, _, 1, _, _, _, _, 4, _,
            9, 5, _, _, _, _, _, 1, 7,
            _, _, 7, 8, _, _, 5, 6, _,
            4, _, _, _, 9, 7, _, 5, _,
            _, _, _, _, 8, _, _, _, _,
            _, 3, _, 5, 4, _, _, _, 1,
            _, 6, 2, _, _, 4, 1, _, _,
            1, 9, _, _, _, _, _, 8, 6,
            _, 7, _, _, _, _, 9, _, _ ],
```

Sudoku : Answer to Example

6	8	1	7	5	9	3	4	2
9	5	3	4	2	6	8	1	7
2	4	7	8	1	3	5	6	9
4	1	6	3	9	7	2	5	8
5	2	9	6	8	1	4	7	3
7	3	8	5	4	2	6	9	1
8	6	2	9	7	4	1	3	5
1	9	4	2	3	5	7	8	6
3	7	5	1	6	8	9	2	4

```
| ?- go(A,B,C,D,E,F,G,H,I).
A = [6,8,1,7,5,9,3,4,2],
B = [9,5,3,4,2,6,8,1,7],
C = [2,4,7,8,1,3,5,6,9],
D = [4,1,6,3,9,7,2,5,8],
E = [5,2,9,6,8,1,4,7,3],
F = [7,3,8,5,4,2,6,9,1],
G = [8,6,2,9,7,4,1,3,5],
H = [1,9,4,2,3,5,7,8,6],
I = [3,7,5,1,6,8,9,2,4] ?
```