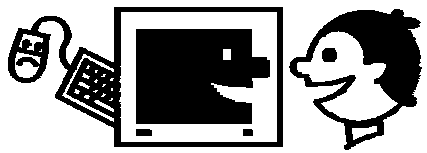
Language, Logic and Computation

Lecture 1

Matthew Stone



Department of Computer Science
Center for Cognitive Science
Rutgers University

Prolog Programs

- ***Facts*** specify information the program uses

  `loves(terry, sandy)`

  - ***Predicates*** correspond to domain relations

    `loves` corresponds to loving relation
  - ***Terms*** correspond to domain individuals

    `terry` corresponds to Terry

    `sandy` corresponds to Sandy

Prolog Programs

- ***Rules*** describe inferences–computations

```
loves(terry, X) :- loves(X, terry).
```

  – ***Variables*** range over all individuals

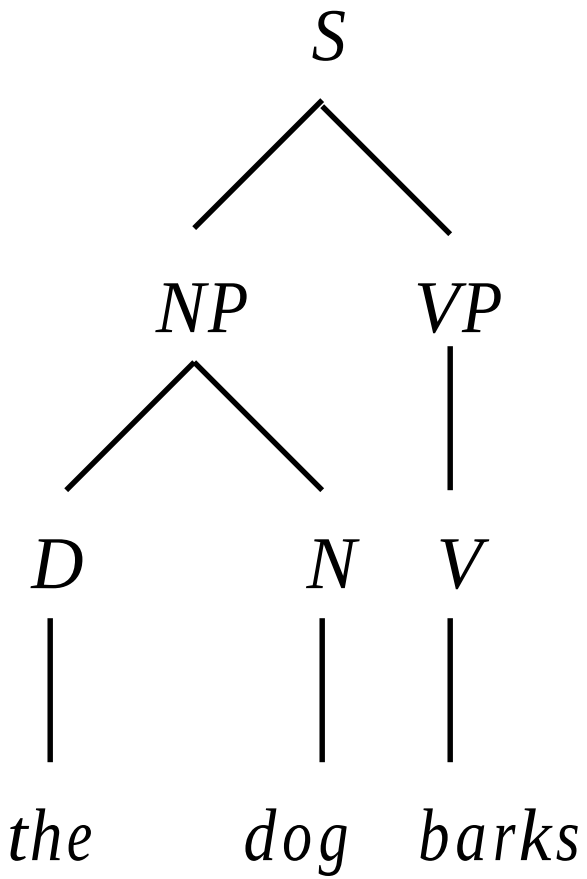
    If someone X loves Terry, Terry loves them back.

    “Terry returns love.”

Prolog Programming as Knowledge Representation

- Start by understanding the world.
  - Determine the ***objects*** that you need to consider; create Prolog terms for them.
  - Determine the ***relationships*** that you need to know about those objects; create Prolog predicates for them.
  - Develop a precise conceptualization that takes a stand on meaningful choices.
- State the facts.

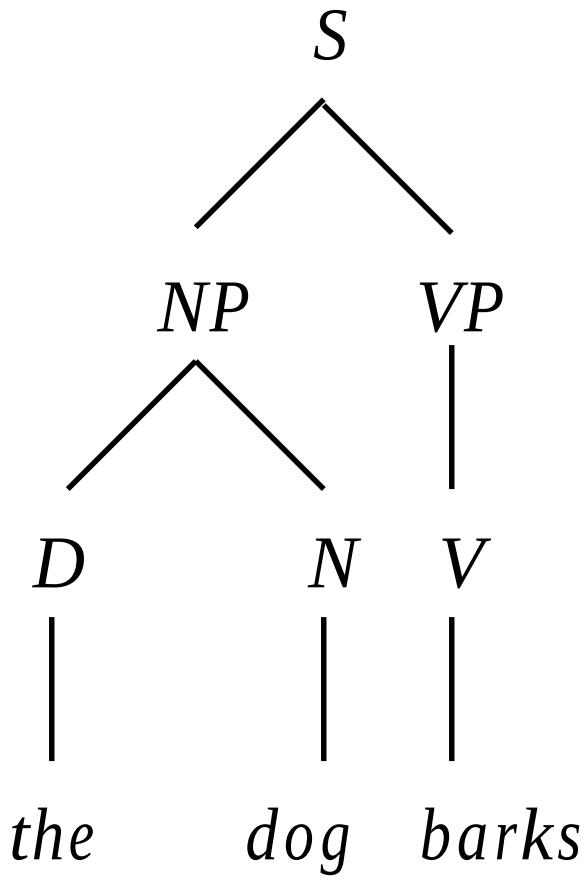
Case study: linguistic structure



Case study: linguistic structure

- Objects:
  - Nodes
  - Categories
  - Words
  - Edges?

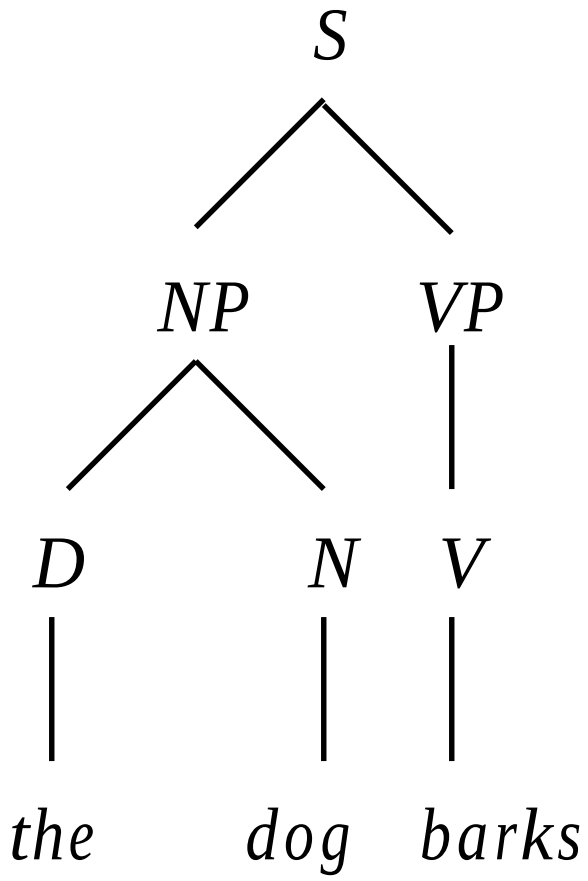
Representation means being ***selective***: what objects do you need explicit–in ***your task***?



Case study: linguistic structure

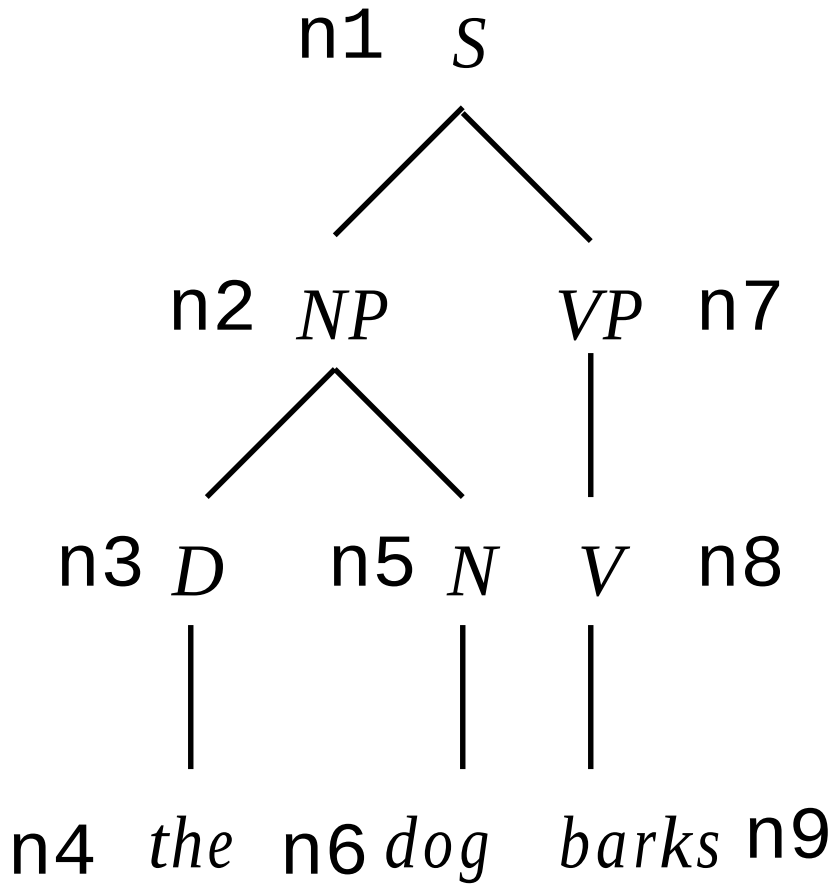
- Relations:
  - Label(node, category)
  - Children(node,nodes)
  - Text(node,word)
  - Root(node)

Representation means being ***selective***: what info do you need explicit–in ***your task***?



Case study: linguistic structure

- Objects:
  - Nodes



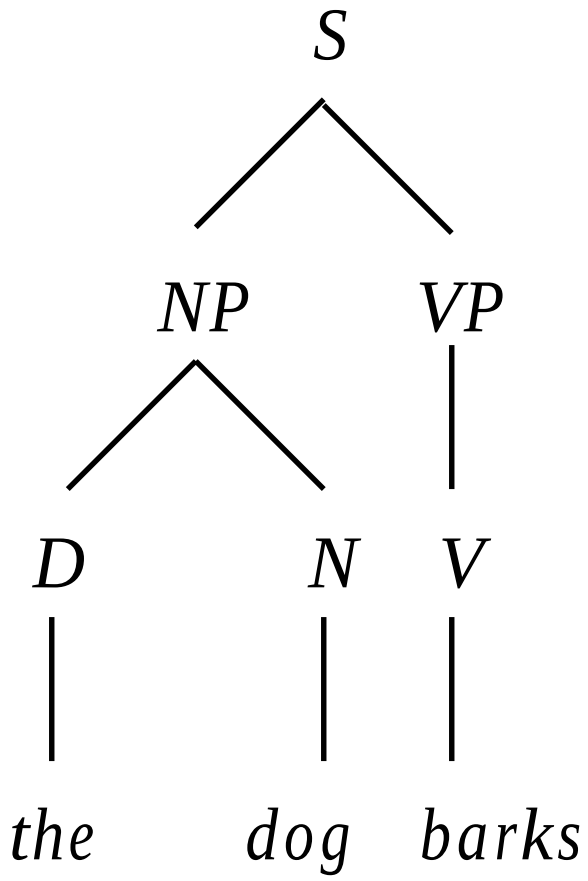
Case study: linguistic structure

- Objects:
  - Categories

```
s
np
d
n
vp
v
```

  - Words

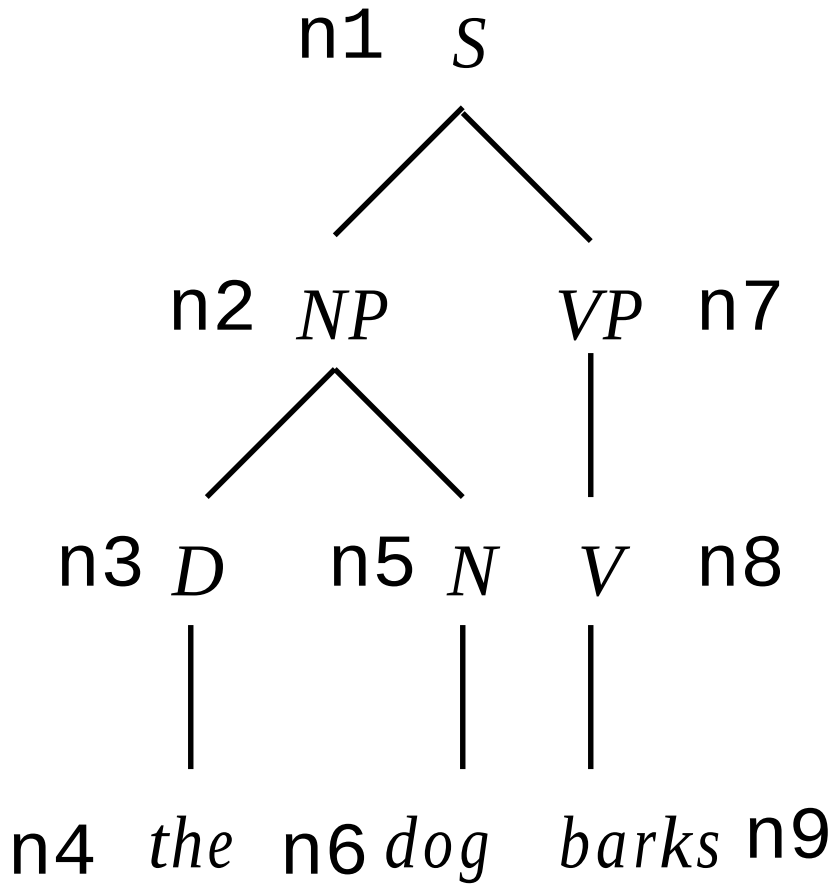
```
the
dog
barks
```



Case study: linguistic structure

- Relations:

```
label(n1,s).
label(n2,np).
label(n3,d).
label(n5,n).
label(n7,vp).
label(n8,v).
root(n1).
```

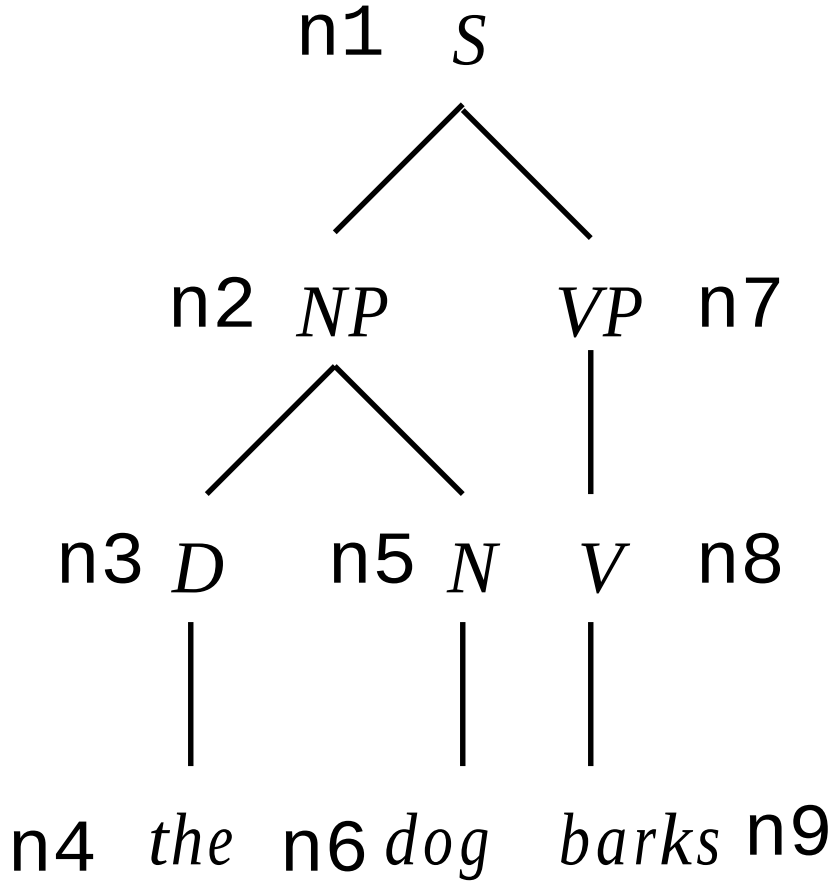


Case study: linguistic structure

- Relations:

```
children(n1,[n2,n7]).
children(n2,[n3,n5]).
children(n3,[n4]).
children(n5,[n6]).
children(n7,[n8]).
children(n8,[n9]).

text(n4,the).
text(n6,dog).
text(n9,barks).
```



Aside: Rules, representation and inference

You can determine a node's ***parent.***

`parent(N,P)` is true if N is an element of the list of P's children.

```
parent(N,P) :-
   children(P,L),
   element(N,L).
```

Aside: Rules, representation and inference

You can define element too:

`element(N,L)` is true if L is a list with head H and tail T, and N = H, or N is an element of T.

```
element(N,L) :-
   L = [H|T],
   (N = H; element(N,T)).
```

Equivalently

```
element(N,L) :-
   L = [H|T], N = H.

element(N,L) :-
   L = [H|T], element(N,T).
```

Equivalently

```
element(N,[N|_T]).
element(N,[_H|T]) :-
   element(N,T).
```

Inference

```
? parent(n7,X)

parent(N’,P’) :-
     children(P’,L’),
     element(N’,L’).

? children(X,L’),
  element(n7,L’).

children(n1,[n2,n7]).

? element(n7,[n2,n7]).
```

```
children(n1,[n2,n7]).

parent(N,P) :-
     children(P,L),
     element(N,L).

element(N,[N|_T]).
element(N,[_H|T]) :-
     element(N,T).
```

Inference

```
? element(n7,[n2,n7]).

element(N’,[_H’|T’]) :-
  element(N’,T’)

? element(n7, [n7]).

element(N’’,[N’’|_T’’]).
```

```
children(n1,[n2,n7]).

parent(N,P) :-
      children(P,L),
      element(N,L).

element(N,[N|_T]).
element(N,[_H|T]) :-
      element(N,T).
```

Data Structures

Package together related information.

- To understand `n1`, you need to know about a lot of separate objects that `n1` is related to.

Allow information in program to change.

- Otherwise you must always have `n1` with the same parent, children, label, etc.

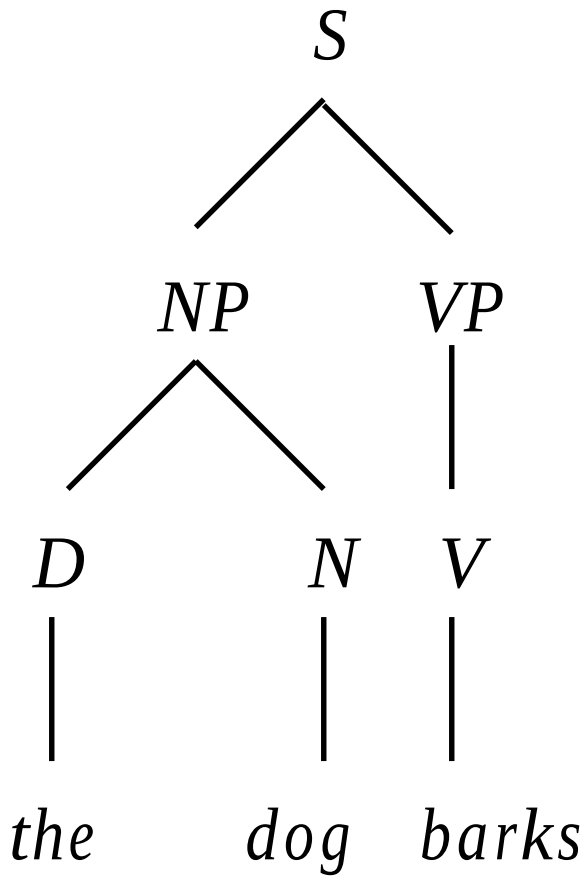
Case study: linguistic structure

- Objects:
  - Categories

```
s
np
d
n
vp
v
```

  - Words

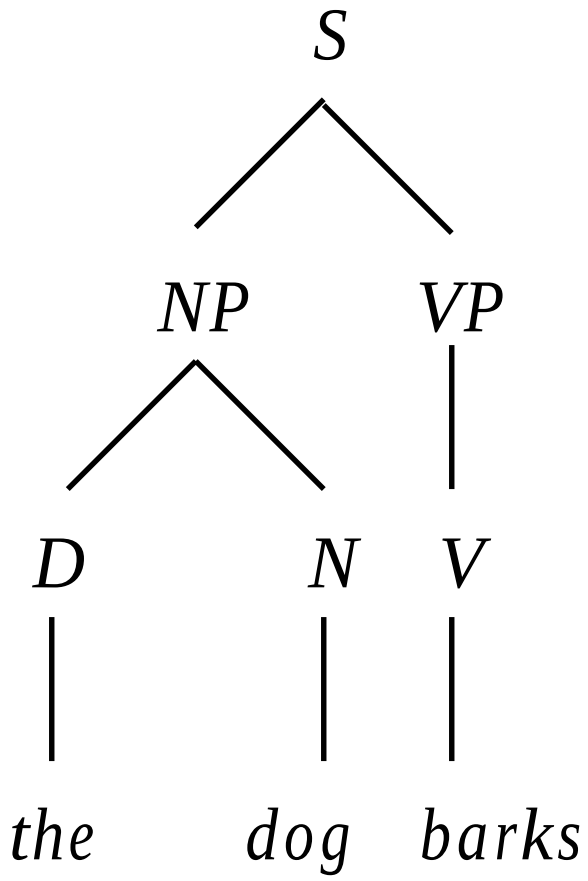
```
the
dog
barks
```



Linguistic data structures

- Lexical nodes:

```
leaf(the)
leaf(dog)
leaf(barks)
```

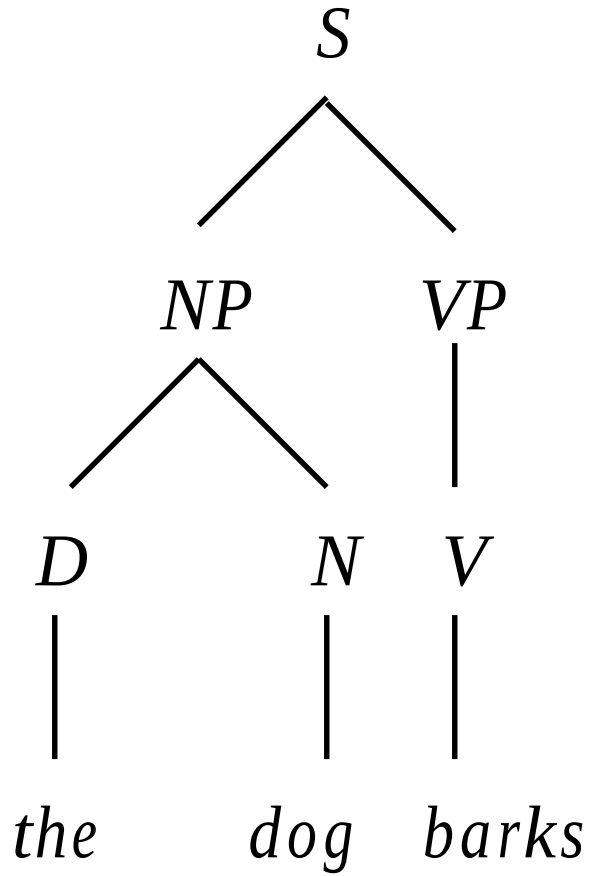


Linguistic data structures

- Internal nodes:

```
node(d,[leaf(the)])
node(n,[leaf(dog)])
node(v,[leaf(barks)])
```

```
node(np,
      [node(d,[leaf(the)]),
       node(n,[leaf(dog)])])
```



Linguistic data structures

```
node(s,
     [node(np,
           [node(d,[leaf(the)]),
            node(n,[leaf(dog)])]),
      node(vp,
           [node(v,[leaf(barks)])])])
```