Language, Logic and Computation
Lecture 1

Matthew Stone

THE VILLAGE

Department of Computer Science
Center for Cognitive Science
Rutgers University
Prolog Programs

• **Facts** specify information the program uses
  
  \[ \text{loves}(\text{terry}, \text{sandy}) \]

  – **Predicates** correspond to domain relations
    
    \[ \text{loves} \] corresponds to loving relation

  – **Terms** correspond to domain individuals
    
    \[ \text{terry} \] corresponds to Terry
    
    \[ \text{sandy} \] corresponds to Sandy
Prolog Programs

- **Rules** describe inferences–computations
  
  ```prolog
  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

```
S
   /\          \                        /
  NP   VP       D   N   V
   /\  /\      /\  /\     /\  /\      /
  the dog barks
```
Case study: linguistic structure

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

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

\[
S \quad \begin{array}{c}
\text{NP} \\
D \\
\text{the} \\
\text{dog}
\end{array} \quad \begin{array}{c}
\text{VP} \\
N \\
\text{barks}
\end{array}
\]
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?

\[ \text{S} \]
\[ \text{NP} \quad \text{VP} \]
\[ \text{D} \quad \text{N} \quad \text{V} \]
\[ \text{the} \quad \text{dog} \quad \text{barks} \]
Case study: linguistic structure

- Objects:
  - Nodes

\[
\begin{array}{c}
  n1 \\
  \downarrow \\
  n2 \\
  \downarrow \\
  n3 \quad D \quad n5 \quad N \quad V \quad n8 \\
  \downarrow \\
  n4 \quad the \quad n6 \quad dog \quad barks \quad n9 \\
\end{array}
\]
Case study: linguistic structure

- Objects:
  - Categories
    - s
    - np
    - d
    - n
    - vp
    - v
  - Words
    - the
dog
barks
Case study: linguistic structure

- Relations:
  \[ \text{label}(n_1, \text{s}). \]
  \[ \text{label}(n_2, \text{np}). \]
  \[ \text{label}(n_3, \text{d}). \]
  \[ \text{label}(n_5, \text{n}). \]
  \[ \text{label}(n_7, \text{vp}). \]
  \[ \text{label}(n_8, \text{v}). \]
  \[ \text{root}(n_1). \]
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).

```
  n1  S
     /|
    / |/
   n2 NP VP n7
   /   |
  n3 D  n5 N V n8
   / |     |
  n4 the n6 dog barks n9
```
Aside: Rules, representation and inference

You can determine a node’s parent.

\[
\text{parent}(N,P) \text{ is true if } N \text{ is an element of the list of } P \text{'s children.}
\]

\[
\text{parent}(N,P) :-
\]
\[
\text{children}(P,L),
\]
\[
\text{element}(N,L).
\]
Aside: Rules, representation and inference

You can define element too:

\[
\text{element}(N,L) \text{ is true if } L \text{ is a list with head } H \text{ and tail } T, \text{ and } N = H, \text{ or } N \text{ is an element of } T.
\]

\[
\text{element}(N,L) :-
\]
\[
L = [H|T],
\]
\[
(N = H; \text{ element}(N,T)).
\]
Equivalently

\[
\text{element}(N,L) \ :-
\]
\[
L = [H|T], \ N = H.
\]

\[
\text{element}(N,L) \ :-
\]
\[
L = [H|T], \ \text{element}(N,T).
\]
Equivalently

\[\text{element}(N, [N|_T]).\]
\[\text{element}(N, [_H|T]) :- \]
\[\text{element}(N,T).\]
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).

parent(n7,X)

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

? children(X,L'),
    element(n7,L').

children(n1,[n2,n7]).

element(N,[N|_T]).

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

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

children(n1, [n2,n7]).

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

parent(N,P) :-
element(N,T'),

children(P,L),
element(N,L).

? element(n7, [n7]).

element(N'',[N''|_T'']).

element(N,'',[N'|_T'']).

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

```
S
  /  \
/    \nNP    VP
  /  \
D    N
     /  \
the dog barks
```
Linguistic data structures

- Lexical nodes:
  leaf(the)
  leaf(dog)
  leaf(barks)
Linguistic data structures

• Internal nodes:
  
  \[
  \text{node}(d, [\text{leaf}(\text{the})]) \\
  \text{node}(n, [\text{leaf}(\text{dog})]) \\
  \text{node}(v, [\text{leaf}(\text{barks})])
  \]

\[
\text{node}(\text{np}, \text{[node}(d, [\text{leaf}(\text{the})]), \text{node}(n, [\text{leaf}(\text{dog})])))
\]
Linguistic data structures

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