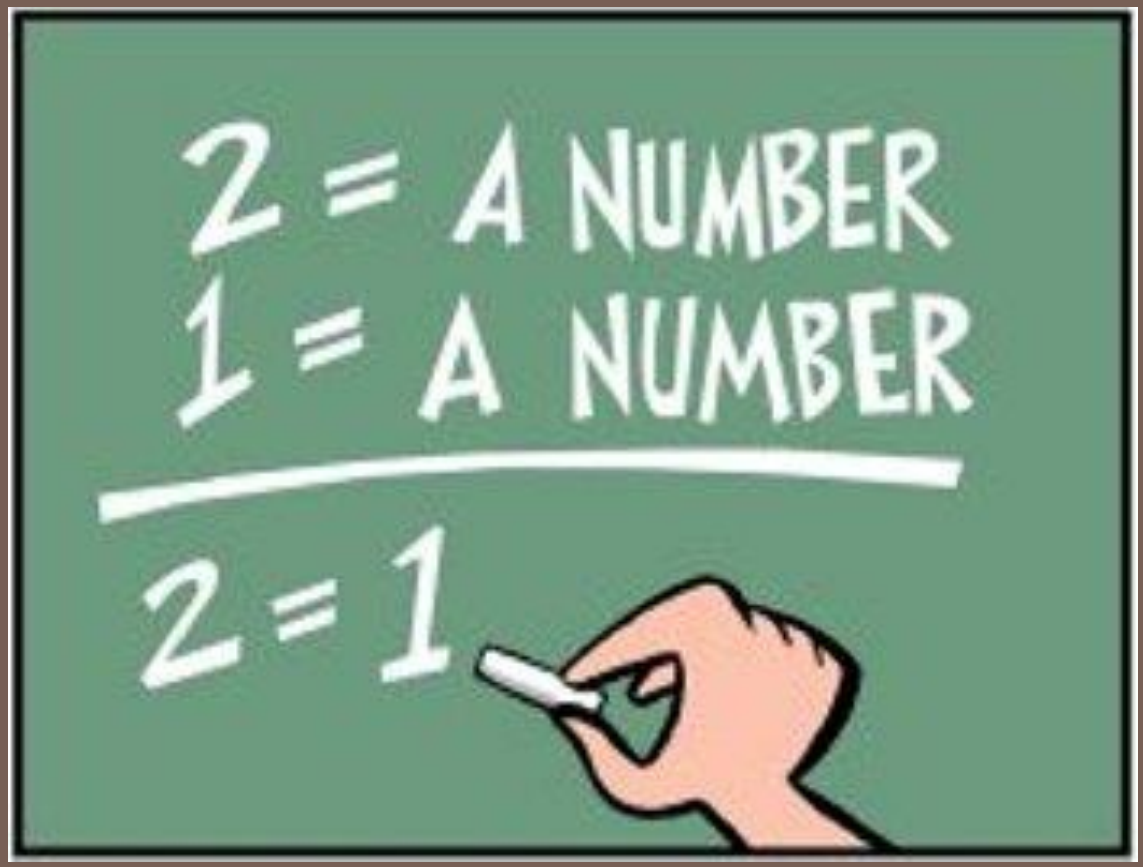


Lecture 2: Prolog as a language

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

Syntax

Terms:

- constant
- Variable
- structure

Constants

- Naming (specific objects, specific relationships)
  - `likes mary john book wine owns jewels can_steal`
  - `a`
  - `void`
  - `=`
  - `'george-smith'`
  - `-->`
  - `george_smith`
  - `ieh2304`
- Integers (size is implementation dependent)

Non-Constants

The following symbols are not constants:

- `2340ieh` – Begins with number.
- `george-smith` – Contains dash.
- `Void` – Begins with capital.
- `_alpha` – Begins with underscore.

Variables

Begin with capital or with underscore:

- Answer
- Input
- _3_blind_mice

Anonymous variable: A single underscore

- `likes(john,_).`
- Need not be assigned to the same variable `likes(_,_).`

Structures

- Collection of Objects, *Components*, grouped together in one object.
- Help Organize.
- Make code more readable.

Structures

Example: Index Card for Library

- Author's Name
- Title
- Date
- Publisher
- Name could be split also first, last, etc.

Examples

- `owns(john,book).`
- One Level:

```
owns(john,wuthering_heights).
owns(mary,moby_dick).
```

- Deeper:

```
owns(john,book(wuthering_heights,bronte)).
owns(john,book(wuthering_heights,
     author(emily,bronte))).
```

Questions

- Does John own a book by the Bronte sisters?

```
owns(john,book(X,author(Y,bronte))).
```

- For the yes/no question

```
owns(john,book(_,author(_,bronte))).
```

(note that each _ could be different)

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

Equality

An infix operator =

- `X = Y`
  A match is attempted between expression `X` and expression `Y`
- PROLOG does what it can to match `X` and `Y`

Example: Instantiated

- X is uninstantiated.
- Y is an object.
- X = Y: X and Y will be matched.
- Thus X will be instantiated by the object Y.

```
?- rides(man,bicycle) = X.

X = rides(man,bicycle) ;

No
```

Example: Symbols


```
?- policeman = policeman.

Yes
?- paper = pencil.

No
?- 1066 = 1066.

Yes
?- 1206 = 1583.

No
```

Arguments Instantiated

- If the structures are equal then their arguments are matched.

```
?- rides(man,bicycle) = rides(man,X).

X = bicycle ;

No
```

Arguments Instantiated


```
?- a(b,C,d(e,F,g(h,i,J))) =
   a(B,c,d(E,f,g(H,i,j))).

B = b
C = c
E = e
F = f
H = h
J = j ;

No
```

Equality


```
?- X=Y, X=1200.

X = 1200
Y = 1200 ;

No
?-
```

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

Arithmetic Comparisons


```
X = Y

X \= Y

X < Y

X > Y

X =< Y

X >= Y
```

Arithmetic


```
?- 123 > 14.

Yes
?- 14 > 123.

No
?- 123 > X.
ERROR: Arguments are not sufficiently
instantiated
?-
```

Example

- `Prince` was a prince during year, `Year` if `Prince` reigned between years `Begin` and `End`, and `Year` is between `Begin` and `End`.

```
prince(Prince,Year) :-
        reigns(Prince,Begin,End),
        Year >= Begin,
        Year =< End.

reigns(rhodri,844,878).
reigns(anarawd,878,916).
reigns(hywel_dda,916,950).
reigns(lago_ad_idwal,950,979).
reigns(hywel_ab_ieuaf,979,985).
reigns(cadwallon,985,986).
reigns(maredudd,986,999).
```

Runs

- Was Cadwallon a prince in 986?
- Is Rhodri a prince in 1995?

```
?- prince(cadwallon,986).

Yes
?- prince(rhodri,1995).

No
?-
```

Who was a Prince When

- Who was the prince in 900?
- Who was the prince in 979?

```
?- prince(Prince,900).
Prince = anarawd ;

No
?- prince(Prince,979).

Prince = lago_ad_idwal ;

Prince = hywel_ab_ieuaf ;

No
?-
```

Invalid Question

- When was Cadwallon a prince?

```
?- prince(cadwallon,Year).
ERROR: Arguments are not sufficiently
instantiated
```

Arithmetic Operations


```
X + Y
X - Y
X * Y
X / Y
X mod Y
```

Calculating

- Calculating the Population Density of a Country: Population over the Area

```
density(Country,Density) :-
        pop(Country,Pop),
        area(Country,Area),
        Density is Pop/Area.

pop(usa,305).
pop(india,1132).
pop(china,1321).
pop(brazil,187).

area(usa,3).
area(india,1).
area(china,4).
area(brazil,3).
```

Questions

- What is the population density of USA?

```
?- density(usa,X).

X = 101.667 ;

No
```

Questions

- What Country has which density?

```
?- density(X,Y).

X = usa
Y = 101.667 ;

X = india
Y = 1132 ;

X = china
Y = 330.25 ;

X = brazil
Y = 62.3333 ;

No
?-
```

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

How Prolog Answers Questions

Program:

```
female(mary).

parent(C,M,F):-mother(C,M),father(C,F).

mother(john,ann).
mother(mary,ann).

father(mary,fred).
father(john,fred).
```

Question:

```
?-female(mary),parent(mary,M,F),parent(john,M,F).
```

How does it work?

Matching

- An uninstantiated variable will match any object. That object will be what the variable stands for.
- An integer or atom will only match itself.
- A structure will match another structure with the same functor and the same number of arguments and all corresponding arguments must match

How Is this Matched


```
?- sum(X+Y) = sum(2+3).

X = 2,
Y = 3
```

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

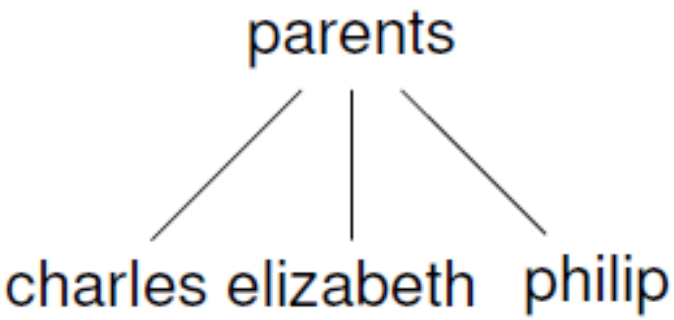
Representing Structures as Trees

Structures can be represented as trees:

- Each functor — a node.
- Each component — a branch.

Example

parents(charles,elizabeth,philip).

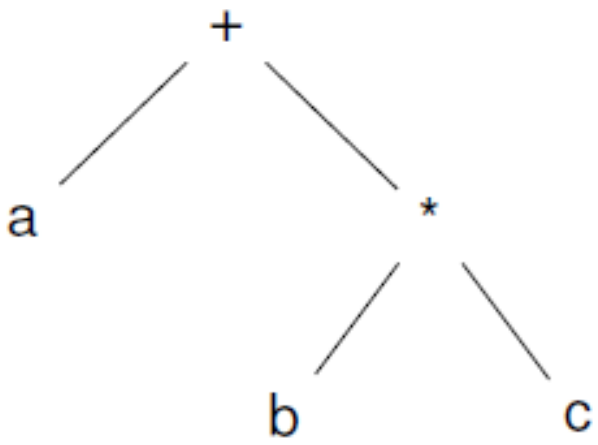


Representing Structures as Trees

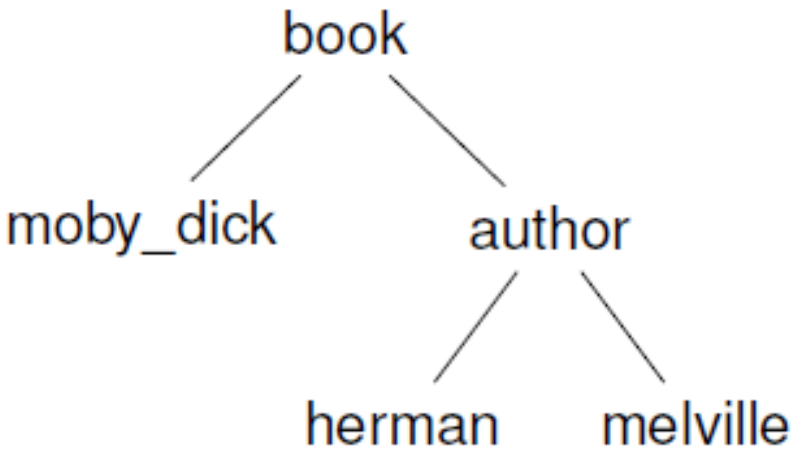
Branch may point to another structure: nested structures.

Example

a+b*c.



book(moby_dick,author(herman, melville)).



Parsing

Represent a syntax of an English sentence as a structure.

Simplified view:

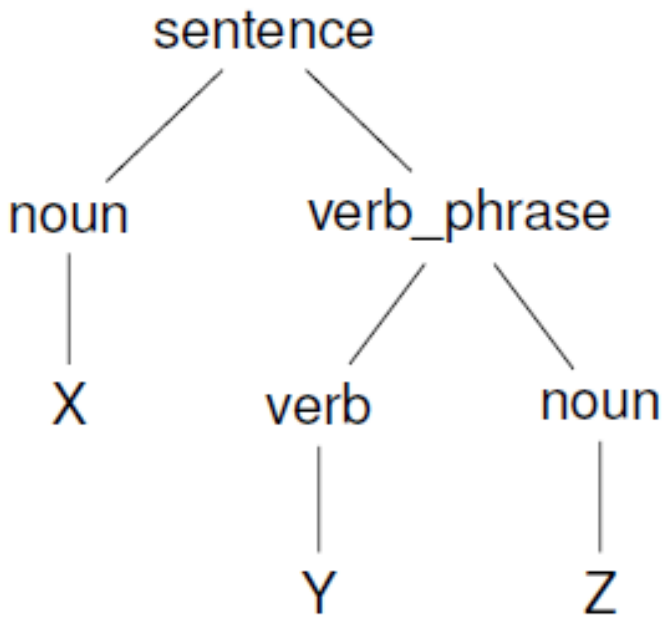
- Sentence: noun, verb phrase.
- Verb phrase: verb, noun.

Parsing

Structure:

```
sentence(noun(X),verb_phrase(verb(Y),noun(Z))).
```

Tree representation:

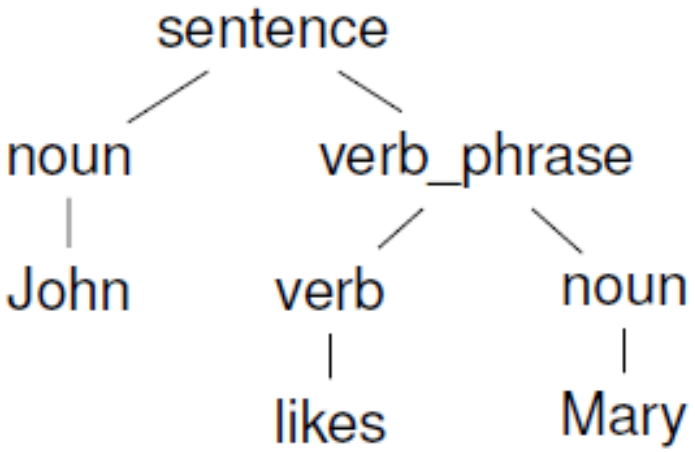


Parsing

Example

John likes Mary.

sentence(noun(John),verb_phrase(verb(likes),noun(Mary))).



Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

Lists

- Very common data structure in nonnumeric programming.
- Ordered sequence of elements that can have any length.
  - Ordered: The order of elements in the sequence matters.
  - Elements: Any terms — constants, variables, structures — including other lists.
- Can represent practically any kind of structure used in symbolic computation.
- The only data structures in LISP — lists and constants.
- In PROLOG — just one particular data structure.

Lists

A list in PROLOG is either

- the empty list [ ], or
- a structure $.(h, t)$ where h is any term and t is a list.
  h is called the head and t is called the tail of the list $.(h, t)$.

Example

- $[]$.
- $.(a, [])$.
- $.(a, .(b, []))$.
- $.(a, .(a, .(1, [])))$.
- $.(.(f(a, X), []), .(X, []))$.
- $.([], [])$.

NB. $.(a, b)$ is a PROLOG term, but not a list!

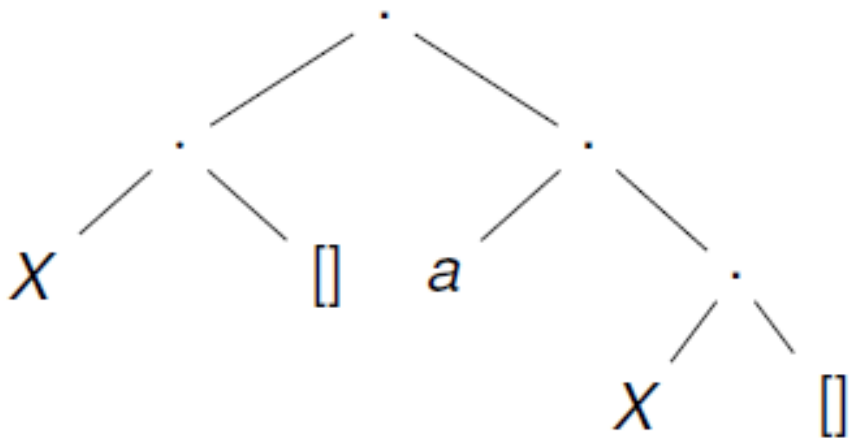
Lists as Trees

Lists can be represented as a special kind of tree.

Example

$.(a, [])$

$.(.(X, []), .(a, .(X, [])))$



List Manipulation

Splitting a list *L* into head and tail:

- Head of *L* — the first element of *L*.
- Tail of *L* — the list that consists of all elements of *L* except the first.

Special notation for splitting lists into head and tail:

- $[X|Y]$, where *X* is head and *Y* is the tail.

NB. $[a|b]$ is a PROLOG term that corresponds to $.(a, b)$. It is not a list!

Head and Tail

Example

List	Head	Tail
$[a, b, c, d]$	a	$[b, c, d]$
$[a]$	a	$[]$
$[]$	(none)	(none)
$[[the, cat], sat]$	$[the, cat]$	$[sat]$
$[X + Y, x + y]$	$X + Y$	$[x + y]$

Unifying Lists

Example

$[X, Y, Z]$	$=$	$[john, likes, fish]$	$X = john,\ Y = likes,$ $Z = fish$
$[cat]$	$=$	$[X \mid Y]$	$X = cat,\ Y = []$
$[X, Y \mid Z]$	$=$	$[mary, likes, wine]$	$X = mary,\ Y = likes,$ $Z = [wine]$
$[[the, Y], Z]$	$=$	$[[X, hare], [is, here]]$	$X = the,\ Y = hare,$ $Z = [is, here]$
$[[the, Y] \mid Z]$	$=$	$[[X, hare], [is, here]]$	$X = the,\ Y = hare,$ $Z = [[is, here]]$
$[golden \mid T]$	$=$	$[golden, norfolk]$	$T = [norfolk]$
$[vale, horse]$	$=$	$[horse, X]$	(none)
$[white \mid Q]$	$=$	$[P \mid horse]$	$P = white,\ Q = horse$

Strings are Lists

- PROLOG strings — character string enclosed in double quotes.
- Examples: "This is a string", "abc", "123", etc.
- Represented as lists of integers that represent the characters (ASCII codes)
- For instance, the string "system" is represented as [115, 121, 115, 116, 101, 109].

Membership in a List

`member(X,Y)` is true when X is a member of the list Y.

One of two conditions:

1. X is a member of the list if X is the same as the head of the list

```
member(X,[X|_]).
```

2. X is a member of the list if X is a member of the tail of the list

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

Membership in a List (Example)

a. member(X,[X|_]).

b. member(X,[_|Y]):-member(X,Y).

? member(4, [1,2,4,-5,6,8]). T

1.a member(4, [1|[2,4,-5,6,8]]). F

1.b member(4,[_|[2,4,-5,6,8]]):- 2. member(4,[2,4,-5,6,8]). T

2.a member(4,[2|[4,-5,6,8]]. F

2.b member(4,[_|4,-5,6,8]):- 3. member(4,[4,-5,6,8]). T

3.a member(4,[4|[-5,6,8]]). T

3.b member ().

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

Recursion

- First Condition is the *boundary condition*.
  (A hidden boundary condition is when the list is the empty list, which fails.)
- Second Condition is the *recursive case*.
- In each recursion the list that is being checked is getting smaller until the predicate is satisfied or the empty list is reached.

Recursion. Termination Problems

- Avoid circular definitions. The following program will loop on any goal involving `parent` or `child`:

```
parent(X,Y):-child(Y,X).
child(X,Y):-parent(Y,X).
```

- Use left recursion carefully. The following program will loop on `?- person(X):`

```
person(X):-person(Y),mother(X,Y).
person(adam).
```

Recursion. Termination Problems

- Rule order matters.
- General heuristics: Put facts before rules whenever possible.
- Sometimes putting rules in a certain order works fine for goals of one form but not if goals of another form are generated:

```
islist([_|B]):-islist(B).
islist([]).
```

works for goals like `islist([1,2,3])`, `islist([])`, `islist(f(1,2))` but loops for `islist(X)`.

- What will happen if you change the order of `islist` clauses?

Prolog as Language

- Syntax
- Equality
- Arithmetic
- Satisfying Goals
- Structures and Trees
- Lists
- Recursive Search
- Mapping

Mapping؟

- Goal: Construct a new structure from the old one.
- The new structure should be similar to the old one but changed in some way

Map a given structure to another structure given a set of rules:

1. Traverse the old structure component by component.
2. Construct the new structure with transformed components.

Mapping a Sentence to Another

Example

you are a computer maps to a reply i am not a computer.
do you speak french maps to a reply no i speak german.

Procedure:

1. Accept a sentence.
2. Change you to i.
3. Change are to am not.
4. Change french to german.
5. Change do to no.
6. Leave the other words unchanged.

Mapping a Sentence. PROLOG Program

Example

```
change(you,i).
change(are,[am,not]).
change(french,german).
change(do,no).
change(X,X).

alter([],[]).
alter([H|T],[X|Y]) :-
   change(H,X),
   alter(T,Y).
```

Boundary Conditions

- Termination: `alter([],[]).`
- Catch all (If none of the other conditions were satisfied, then just return the same): `change(X,X).`