Lists

- specific functional terms
  - for ordered sets of terms
  - form:

    [], [Arg1], [Arg1,Arg2], [Arg1,Arg2,Arg3]

    …

    Example:

    orderOfappearance([peter,husband(inge),alfons,irene]).

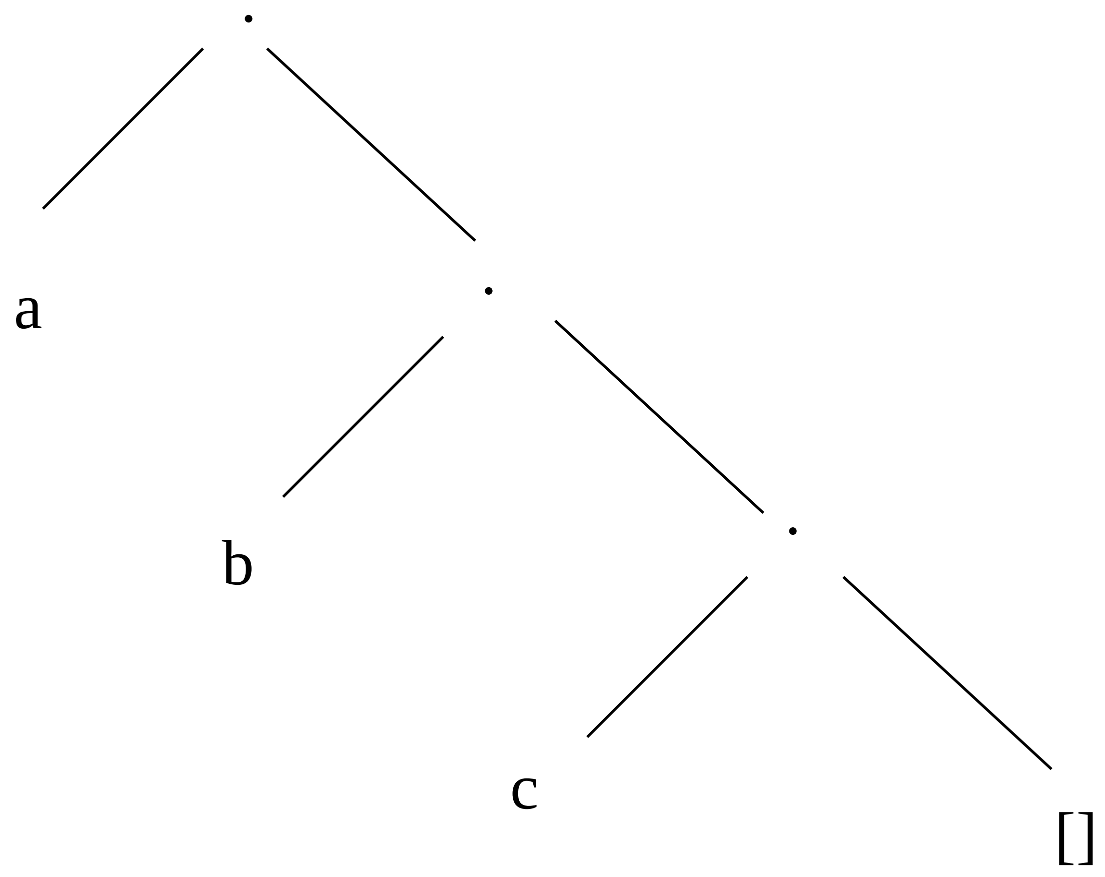
Lists

- internal Form:

```
?-  X=(.(a,.(b,[]))).

X = [a, b]


?- [a,b,c] =.. L.  (L = List of functor and args)

L = ['.', a, [b, c]]
```

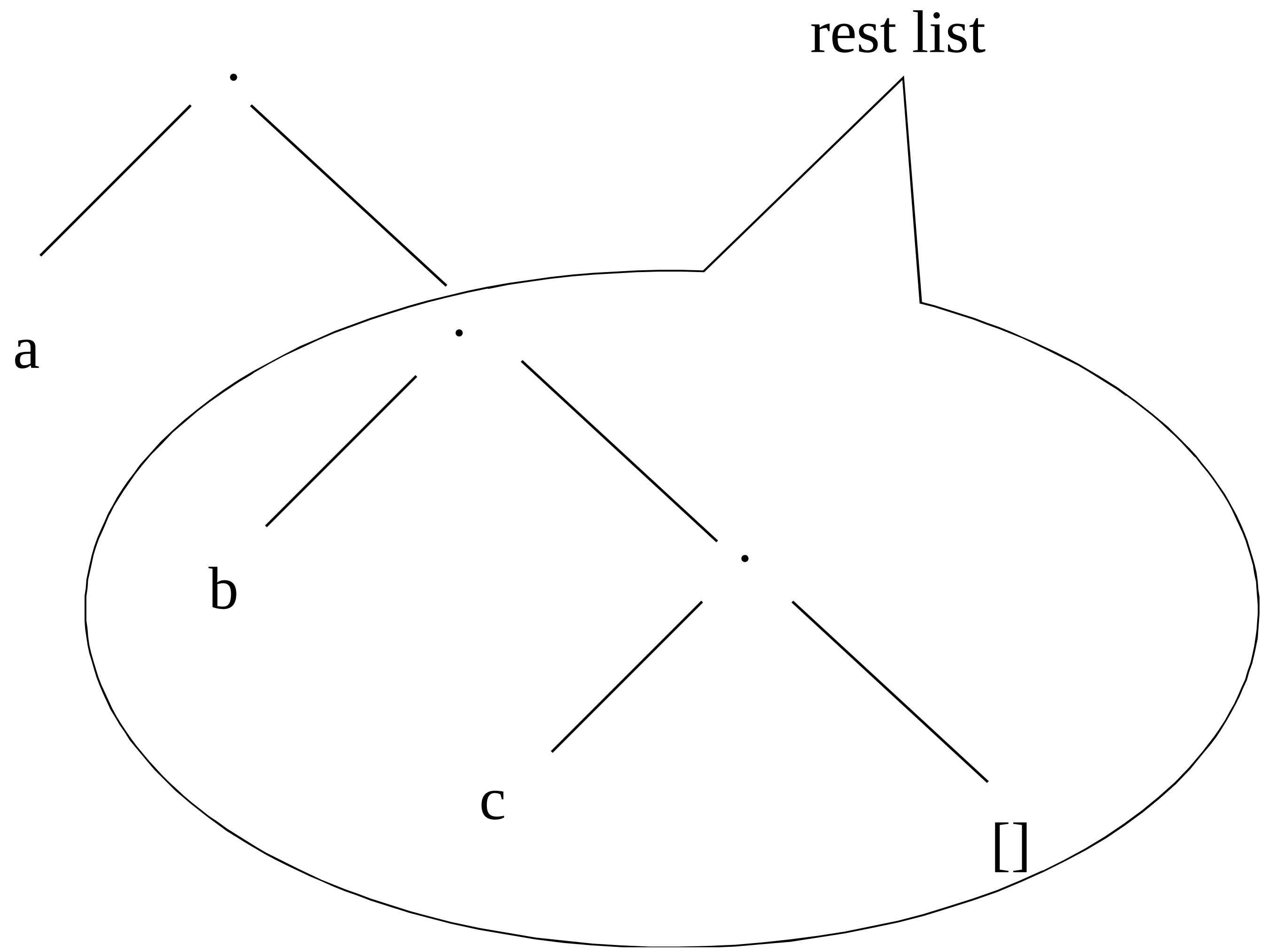
Lists

- internal form:



Lists and partial lists

- Head and Tail



Lists and partial lists

- Head and Tail
- [a|RL]
- '|'   = List separator
- adds rest list as tail to list
- Examples:

```
?- [a,b] = [A|B].      →   A=a,  B=[b]
?- [a,b,c] = [A|B].    →   A=a,  B=[b,c]
```

Lists and partial lists

other examples….

- list items can be lists:

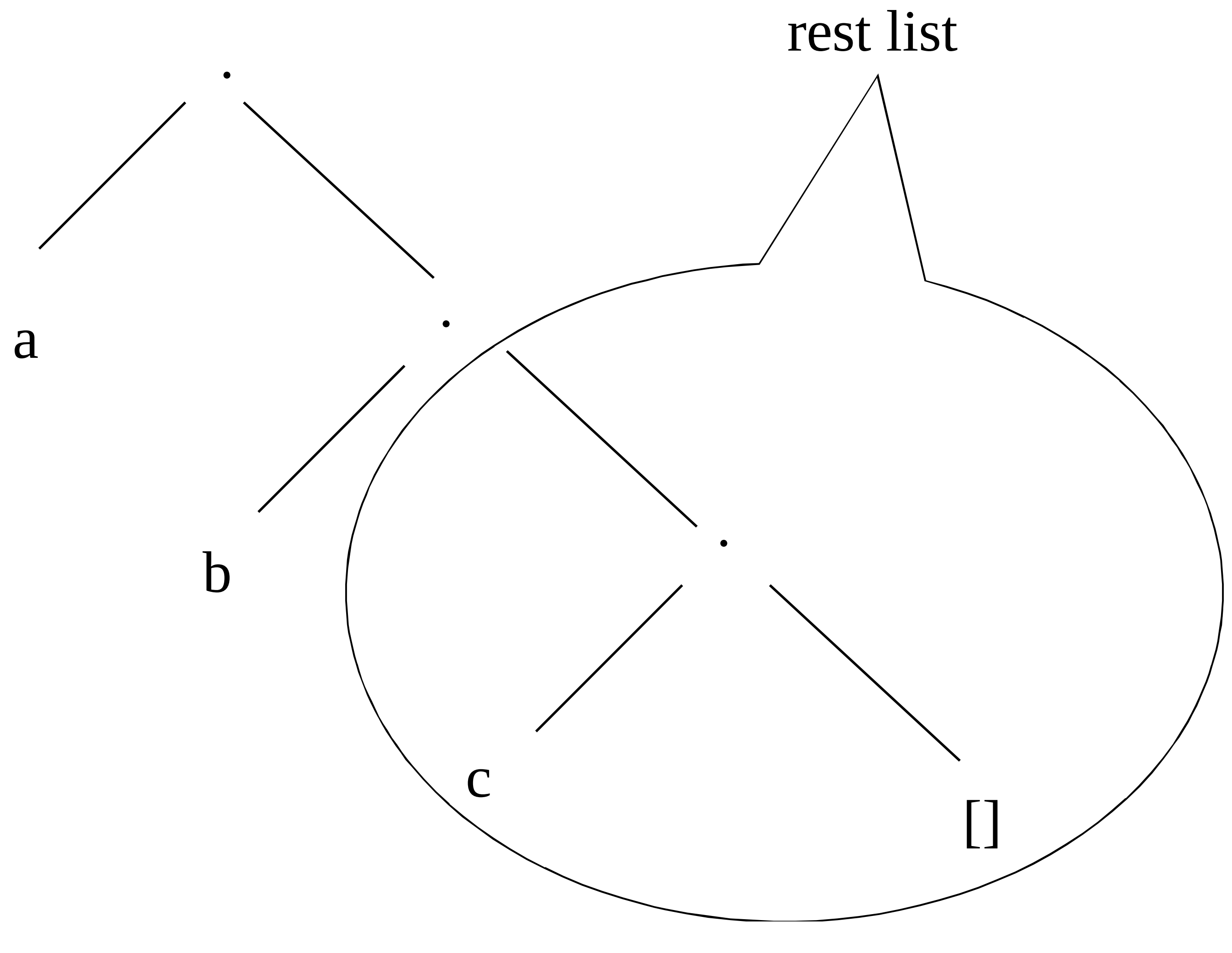
  [alf,[],berta,[o(eva),xx],z]
- or variables….

  [X,[],berta,[o(eva),X],z]

```
?- [X,[],berta,[o(eva),X],z] = [A|B].
→ A=X, B= [[],berta,[o(eva),X],z]]
```

Lists and partial lists

- Head and Tail



Lists and partial lists

- '|'   = list separator
- adds rest list as tail to any list of list items
- Examples:

```
?- [a,b,c,d] = [A,B|L].
                              →  A=a, B=b, L=[c,d]
?- [[],V,a,[a,[b,c]]] = [X,Y,Z|W].
                    →  X=[],  Y=V, Z=a, W=[[a,[b,c]]]
```

Exercise: 2nd and 4th element of a list

```
?-
 [X1,X2,X3,X4 | Tail] = [[], dead(zed), [2, [b, chopper]], [], Z].

X1 = []
X2 = dead(zed)
X3 = [2,[b,chopper]]
X4 = []
Tail = [_8910]
Z = _8910
yes
```

Exercise: 2nd and 4th element of a list

or… with anonymous variables

```
?- [_,X,_,Y|_] = [[], dead(zed), [2, [b, chopper]], [], Z].

X = dead(zed)
Y = []
Z = _9593
yes
```

Exercise: Tail of a list element which is a list

… using anonymous variables for the irrelevant items

```
?- [_,_,[_|X]|_] =
     [[], dead(zed), [2, [b, chopper]], [], Z, [2, [b, chopper]]].

X = [[b,chopper]]
Z = _10087
yes
```

List manipulation

[library(basics),library(lists)].

- member(Element,List)
- append(List1,List2,CompleteList)
- rev(List,tsil)   (reverse)
- length(List,Length)
- sort(List,Slist)

Member definition

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

```
?- member(yolanda,[yolanda,trudy,vincent,jules]).
yes
```

Member definition

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

?- member(vincent,[yolanda,trudy,vincent,jules]).
  ?- member(vincent,[trudy,vincent,jules]).
    ?- member(vincent,[vincent,jules]).
yes
```

Member definition

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

?- member(zed,[yolanda,trudy,vincent,jules]).
  ?- member(zed,[trudy,vincent,jules]).
    ?- member(zed,[vincent,jules]).
      ?- member(zed,[jules]).
      ?- member(zed,[]).
no
```

Member: Order?

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

```
?- member(a,[a,b,c]).
```

Member: Backtracking

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

member(X,[yolanda,trudy,vincent,jules]).

   X = yolanda ;

   X = trudy ;

   X = vincent ;

   X = jules ;

   no
```

Notation

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

use anonymous variable: transparency, readability

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

think recursively!

Functions:

Compare, copy, translate (parts of) lists ...

Examples:

d2e([der,mann,stirbt],[the,man,dies]).

a2([2,3,7,5,4,3],[5,12,7]).

a2b([a,a,a,c,a],[b,b,b,c,b]).

Strategy:

- What is the simplest case?   (mostly [])
- Whar happens at an intermediate step?

think recursively!

Example: a2b

- simplest case (termination condition):

  list empty:

  → a2b([],[])
- intermediate step: a list with first element X

  1) X is a: → Y is b

  2) X is not a: → Y is X

```
a2b([],[]).
a2b([a|AL],[b|BL]) :- a2b(AL,BL).
a2b([X|AL],[X|BL]) :- a2b(AL,BL).
```