

## Prolog syntax

- **Term**
  - Constant: number or name that starts with lowercase letter
    - foo
    - 42
  - Variable: name that starts with uppercase letter
    - Foo
  - Compound term: functor (name) followed by one or more arguments in parentheses
    - Ground compound term: all arguments are constants or variables
      - foo(42, 23)
    - Non-ground compound term: one or more arguments is compound
      - foo(bar(42), 23))
- **Lists**
  - The constant [] is the empty list
  - Functor "." is the cons operator. It constructs a list from an element and a list.
    - .(42, []) = [42]
    - Equivalent: [42, []]
    - .(foo, .(bar, .(baz, []))) = [foo, [bar, [baz, []]]] = [foo, bar, baz]
  - [X | Y]: The list whose head (car) is X and tail (cdr) is Y
    - [foo | [bar, baz]] = [foo, bar, baz]
- **Predicates**
  - Constant or compound term
    - it\_rains
    - lives\_in(brit, red\_house)
- **Program: Sequence of clauses**
  - Clause is either a fact or a rule:
    - Fact (single predicate)
      - lives\_in(brit, red\_house).
    - Rule (Horn clause)
      - conclusion :- condition.
        - condition is a conjunction of predicates
        - drinks(X, beer) :- smokes(X, Winfield).
        - grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
        - ancestor(X, Z) :- parent(X, Z).
        - ancestor(X, Z) :- parent(X, Y), ancestor(Y, Z).
  - In predicate logic:
    - $\forall X, Y, Z (\text{parent}(X, Y) \wedge \text{ancestor}(Y, Z) \Rightarrow \text{ancestor}(X, Z))$
  - Example programs
    - Length of a list
      - length([], 0).
      - length([head | tail], N) :- length(tail, M), N = M + 1.

- Ancestors
  - parent(tom, dick).
  - parent(dick, harry).
- Queries
  - Asking the program about information contained within its data base.
    - ?- it\_rains.
      - Yes
    - ?- ancestor(tom, harry).
      - Yes
    - ?- ancestor(tom, X).
      - X = dick
      - Yes