CS2510
MODERN PROGRAMMING LANGUAGES

Logic Programming 2

Prof. Peter Edwards
p.edwards@abdn.ac.uk

Prolog – Syntax 1

- A Prolog program is a sequence of facts and rules.
- However, the logical symbols must get an ASCII representation:
  - "←" becomes "**:-**"
  - "**,**" is used to represent logical "and".
  - All clauses (fact, rules and queries) must end with "**.**"
    - To inform the interpreter that it is indeed the end!
- Prolog programs can be compiled but mostly they are interpreted:
  - Edit your program in a file (.pl) and load it in Prolog.
  - Run the program by entering a query.

Prolog – Syntax 2

- Prolog is *dynamically* typed.
- It has a single data type, the ***term***, which has several subtypes:
  - *atoms*
  - *numbers*
  - *variables*
  - *compound terms*

- **Atoms**
  - A primitive data item.
  - Begins with lowercase letter.
  - If atom contains spaces (or other special chars) then surround in single quotes ‘ ‘
  - Examples:

```
rain
socrates
‘My car’
```

- **Numbers**
  - Integers, float, scientific notation.
  - Examples:

```
63
3.14159
-13.45
1.23E+35
```

Atoms and numbers are ***constants*** in Prolog.

Prolog – Syntax 3

- **Variables**
  - Begin with a capital letter (A-Z)
    - or the "_" (underscore) symbol.
  - Examples:

    ```
    X, X1, X_1,
    ThisIsAVariable, _23
    ```

  - The *anonymous* variable "_" (underscore)

- Once a variable gets a value, there is no way to change it.
  - No destructive assignments.
- Variables can be aliased - sharing their values with other variables.
- We say a variable is ***instantiated*** when it has a value, otherwise it is ***uninstantiated***.

Prolog – Syntax 4

- **Compound Terms**
  - *Consist of:*
    FunctorName(***SubTerm***$_1$,…,***SubTerm***$_n$)
  - *FunctorName* must be an atom (not numbers!)
  - *SubTerm*$_i$ is a:
    - constant (atom, number), or
    - a variable, or
    - another (nested) term.

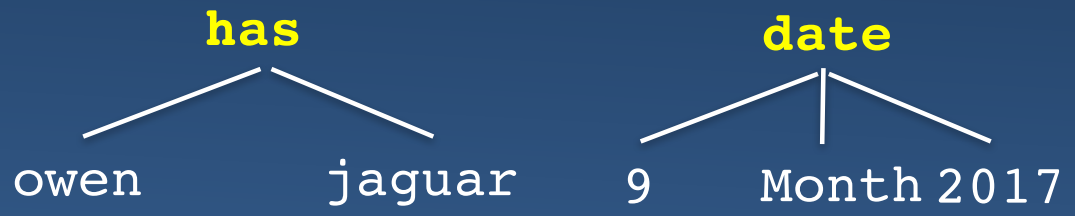
- Examples:

```
has(owen, jaguar)
date(9,Month,2017)
president(usa, trump)
s(s(s(s(s(0)))))
p(q(X,123),a45)
```

Prolog – More on *Terms*

- Compound term: *functor* and *arguments*.

- Number of arguments = ***arity***
  - atom – a compound term with *arity* 0
- Terms are stored internally as *trees*.
- Examples:



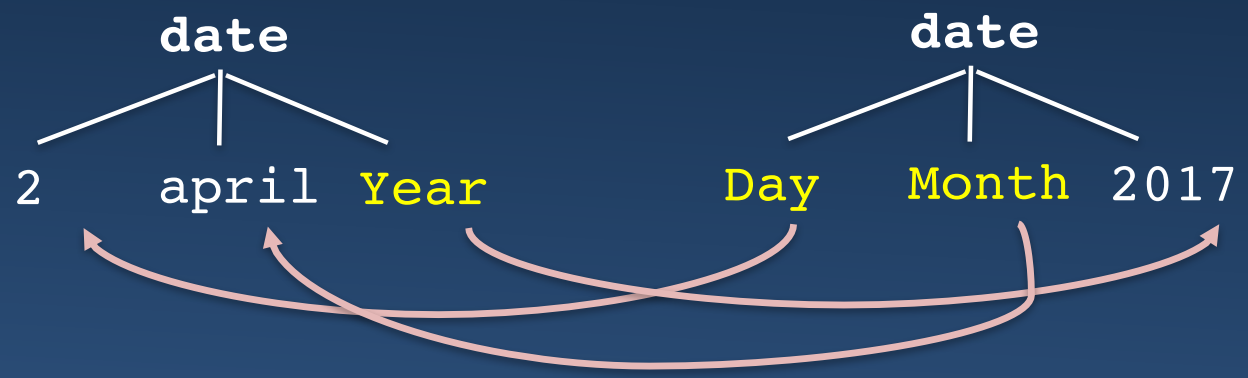
Matching Terms

- Terms can be matched, that is, compared.
- Two terms match if:
  - They are identical
    OR
  - If their variables can be assigned values so that the terms become identical.

- Two terms are matched by:
  - Traversing their trees
  - Comparing their nodes and
  - Assigning values to variables (if needed).
- Terms can be matched via the built-in "=" (*unification*) operator.
  - it compares and, if possible, assigns values to variables.

Matching Terms

- Example:
  - Does `date(2,april,Year)` match `date(Day,Month,2017)` ?



  - `Day` is instantiated to 2
  - `Month` is instantiated to april
  - `Year` is instantiated to 2017

- Do these terms match?

```
p(1,s(X,a),foo)
p(A,s(1,A),B)
```

Prolog - *Facts*

- Facts are terms which must end with "."
- Examples:

```
has(owen, jaguar).
has(owen, car(jaguar)).
has(john,book('AnimalFarm')).
date(9,march,2017).
```

- Facts establish what is true about something.
- Facts represent what is important to the problem we want to solve.
- As programmers, we create facts and their meaning:

```
loves(ann,bob).
```

Who loves whom?

Prolog - *Rules*

- Rules are of the form:

```
Term :- Term, Term, …, Term.
```

head

body

- Rules must end with "."
- We refer to the head and body of a rule.
  - A fact is a rule with an empty body (no ":-")
- The terms of a rule are also called (sub-) goals.

- Examples:

```
rich(X) :- has(X,jaguar).

goodPresident(X) :-
    hair(X, real),
    twitterUser(X, no).
```

Together, rules and facts are known as ***clauses***.

Prolog – Program Structure

- A ***predicate*** is a collection of clauses with the same *functor* (name) and *arity* (number of arguments).
- Example:

```
loves(john, mary).
loves(mary, bill).
loves(owen, X) :-
      female(X),
      rich(X).
```

- A Prolog ***program*** is a collection of predicates.
  - Predicates can be in any order.
  - Clauses within a predicate are used in the order in which they occur.
  - Comments assist readability.
    - % symbol.

```
grandfather(X,Z):-      % X is a grandfather of Z if
    father(X,Y),        %  - X is the father of Y and
    father(Y,Z).        %  - Y is the father of Z
```

Queries

- Queries are of the form
  `:- Term, Term, …, Term.`
- Queries must end with "."
- The terms of a query are also called (sub-)goals.
- A query is a clause without a head term.
- Prolog offers a prompt for us to enter queries, so we don't need to type ":-" (it is assumed).
- Example:

- A Prolog program is run/executed when we pose queries.
- An execution is a proof of a query:
  - If the query is true Prolog will return "true";
  - If not Prolog will return "false".
- We use *execution* & *proof* as synonyms in the context of Prolog programs.

Prolog - Common Mistakes

- Capitalization is *meaningful*!
- No space is allowed between a functor and its argument list:

```
        man(socrates).
not     man (socrates).
```

- Double quotes indicate a list of ASCII character values, *not* a string.
- Scope of variables: their clauses.
  - Lifespan of variables: while clauses are being proved.
- No global variables!
- Recursion is the only means to achieve iteration.

- Don't forget the full stop "."