Prolog Syntax

Prolog syntax and data objects

Weekly Outcomes

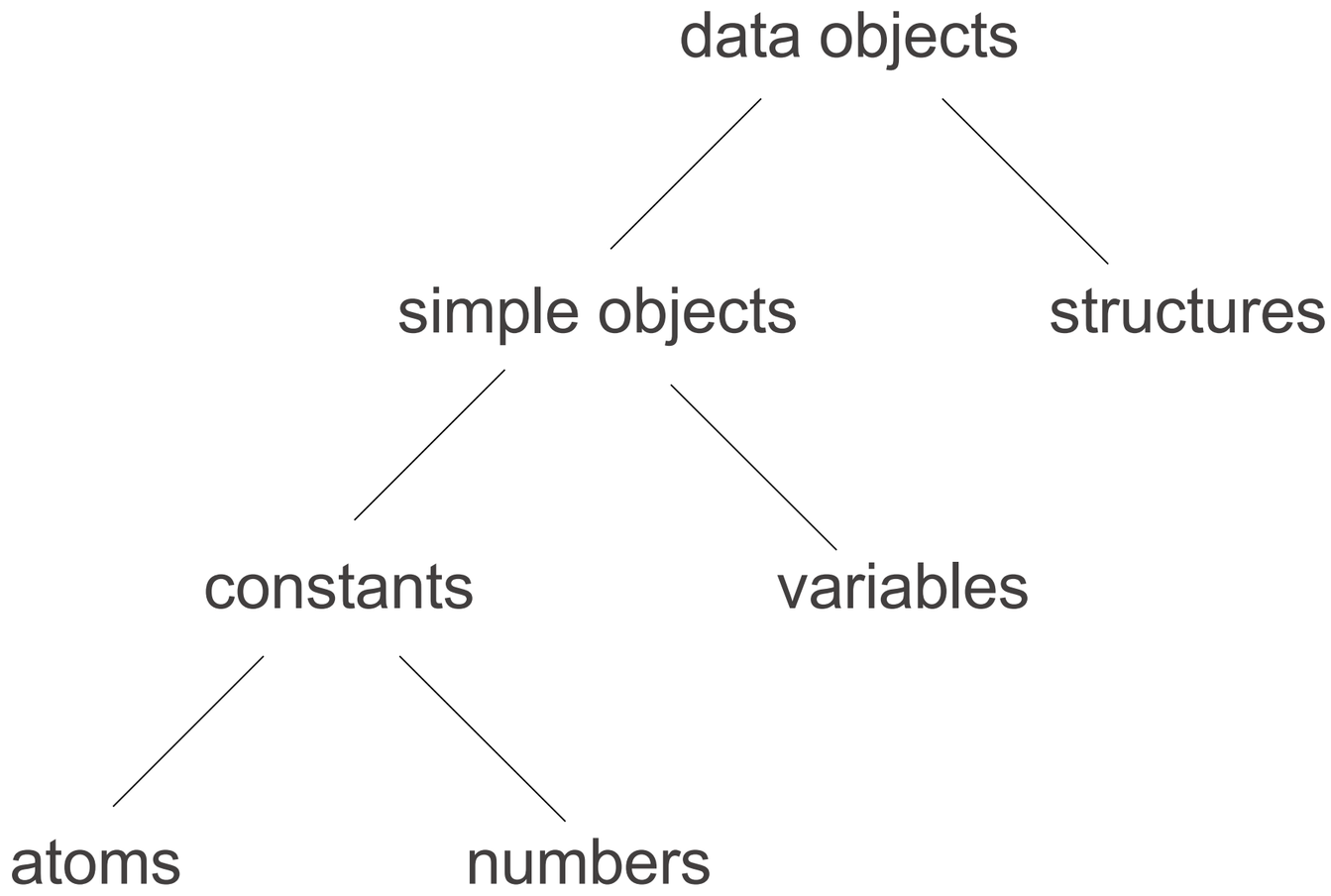
1. Become more effective with Prolog syntax and data structures
2. Explain Prolog Matching

Lesson Overview (Agenda)

In the following lesson, we will explore:

1. Prolog Syntax
2. Prolog data structures
3. Working with Prolog lists

DATA OBJECTS



SYNTAX FOR DATA OBJECTS

- Type of object always recognisable from its syntactic form

Atom Syntax (3 forms)

(1) Strings of letters, digits, and “_”, starting with lowercase letter:

x x15 x_15 aBC_CBa7

alpha_beta_algorithm taxi_35

peter missJones miss_Jones2

Atom Syntax (cont'd)

(2) Strings of special characters

---> <==> <<

. < > + ++ ! .. .::. ::= []

Atom Syntax (cont'd)

(3) Strings in single quotes

'X_35' 'Peter' 'The Beatles'

Syntax for Numbers

- Integers

  1 1313 0 -55

- Real numbers (floating point)

  3.14 -0.0045 1.34E-21 1.34e-21

Syntax for Variables

- Strings of letters, digits, and underscores, starting with uppercase letter or starting with underscore:

  X Results Object2B Participant_list
  _x35 _335

- Single underscore _ the anonymous variable
- Lexical scope of variable names is one clause
  - All instances of the same non-anonymous variable name in a clause must have the same binding (value)
  - Each instance of the anonymous variable in a clause can have the same or a different binding (value)

Anonymous Variable

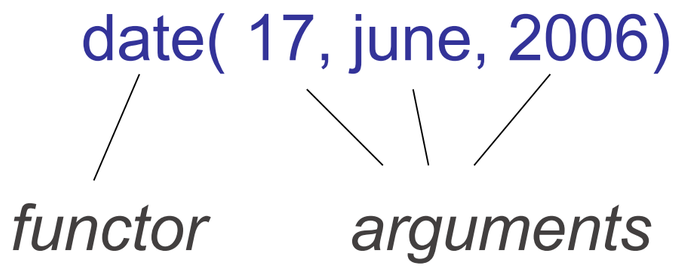
```
visible_block( B)  :-
  see( B, _, _).
```

Equivalent to:

```
visible_block( B)  :-
  see( B, X, Y).
```

Structures

- Structures are objects that have several components
- For example: dates are structured objects with three components
- Date 17 June 2006 can be represented by *term*:



- An argument can be any object, also a structure

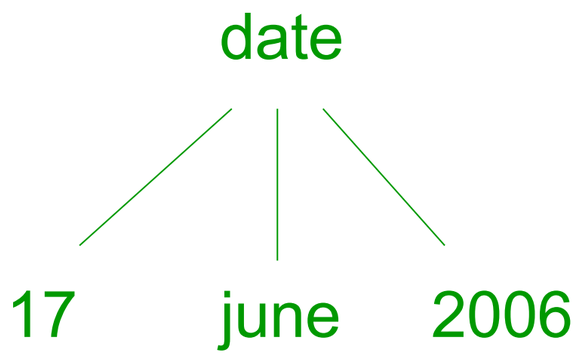
Functors

- Functor name chosen by user
- Syntax for functors: atoms
- Functor defined by name and *arity*

Tree Representation of Structures

Often, structures are pictured as trees

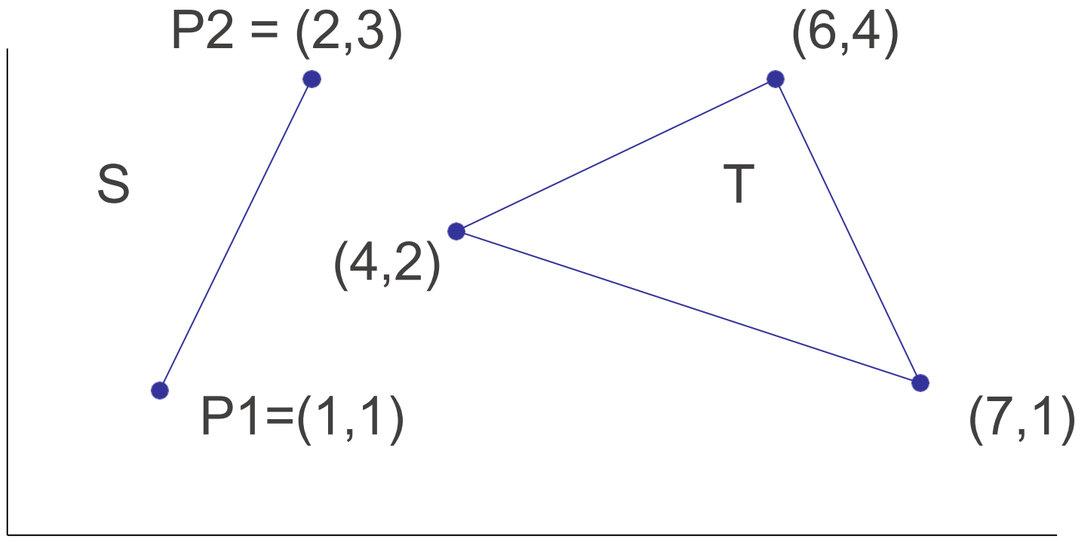
date(17, june, 2006)



Structures/Trees

- Therefore all structured objects in Prolog can be viewed as trees
- This is the *only* way of building structured objects in Prolog

Some Geometric Objects



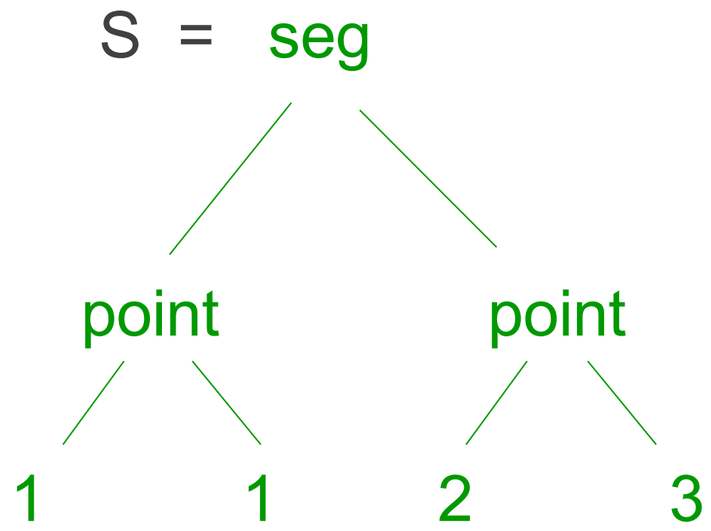
P1 = point(1, 1) P2 = point(2, 3)

S = seg(P1, P2) = seg(point(1,1), point(2,3))

T = triangle(point(4,2), point(5,4), point(7,1))

Line Segment

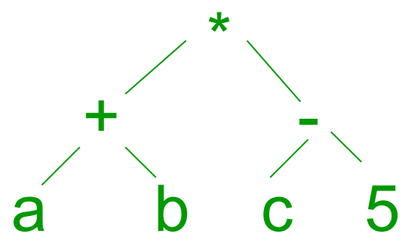
S = seg(point(1,1), point(2,3))



Arithmetic Expressions are Structures

- For example: (a + b) * (c - 5)
- Written as term with functors:

*(+(a, b), -(c, 5))



Time to check your learning!

Let’s see how many key concepts from prolog structures you recall by answering the following questions!

What is the relationship between Prolog structures, and trees?

What is special about an anonymous variable?

What are the parts of a prolog structure called?

this(that(0),theother)

Matching

- Matching is operation on terms (structures)

- Given two terms, they match if:

  (1) They are identical, or
  (2) They can be made identical by properly instantiating the variables in both terms

Matching Example

- Matching two dates:

  date(D1, M1, 2006) = date(D2, june, Y2)

- This causes the variables to be instantiated as:

  D1 = D2

  M1 = june

  Y2 = 2006

- This is the *most general instantiation*
- A less general instantiation would be: D1=D2=17, ...

Most General Instantiation

- In Prolog, matching always results in most general instantiation

- This commits the variables to the least possible extent, leaving flexibility for further instantiation if required

- For example:

```
?- date( D1, M1, 2006)  =  date( D2, june, Y2),
   date( D1, M1, 2006)  =  date( 17, M3, Y3).
```

D1 = 17, D2 = 17, M1 = june, M3 = june,

Y2 = 2006, Y3 = 2006

Matching

- Matching succeeds or fails; if succeeds then it results in the most general instantiation

- Matching algorithm on terms S and T:

  (1) If S and T are constants then they match only if they are identical

  (2) If S is a variable then matching succeeds, S is instantiated to T

  (3) if T is a variable then matching succeeds, T is instantiated to S

  (4) If S and T are structures then they match only if

  (a) they both have the same principal functor, and

  (b) all their corresponding arguments match

Matching ≈ Unification

- Unification known in predicate logic
- Unification = Matching + Occurs check
- Occurs check: does one side occur within the other side?
- Unification is the same as matching except unification fails if one side occurs within the other side
- We can turn on the occurs check if necessary, but for efficiency reasons the default is no occurs check
- What happens when we ask Prolog:

```
?-  X = f(X).
```

Matching succeeds, unification fails

Stating that two things are different

Two possibilities for stating that two things are different:

1. The negation operator, \+, is problematic because of the way Prolog deals with negation (more on this later):

```
\+ X = Y     % unless both X and Y are instantiated, this always fails
\+ X = 4     % this will always fail unless X is instantiated to something not 4
?- X = a, \+ X = Y.       % fails, because Y can match anything including a
?- X = a, Y = b, \+ X = Y.   % true, because a cannot match b
```

2. The dif/2 builtin predicate is part of the constraint system. It's better because it delays the comparison until X and Y are both instantiated

```
dif(X,Y)
```

Computation with Matching

```
% Definition of vertical and horizontal segments

vertical( seg( point( X1,Y1), point( X1, Y2))).
horizontal( seg( point( X1,Y1), point( X2, Y1))).

?- vertical( seg( point( 1,1), point( 1, 3))).
yes
?- vertical( seg( point( 1,1), point( 2, Y))).
no
?- vertical( seg( point( 2,3), P)).
P = point( 2, _173).
```

Degenerate Line Segment

- Is there a segment that is both vertical and horizontal? If vertical and horizontal do not rule out the degenerate line segment, we might have

?- vertical(S), horizontal(S).

S = seg(point(X,Y), point(X,Y))

- Note, Prolog may display this with new variables names as for example:

S = seg(point(_13,_14), point(_13, _14))

If vertical and horizontal allow the degenerate, we could use dif to address this problem:

S = seg(X,Y), dif(X,Y), vertical(S), horizontal(S).

Time to check your learning!

Let’s see how many key concepts from Prolog matching you recall by answering the following questions!

What is the difference between matching and unification?

What can an unbound Prolog variable match with?

Conclusion

In this lesson, you learned more about Prolog data structures and syntax and the matching process

In the next lesson, you will learn to use lists effectively