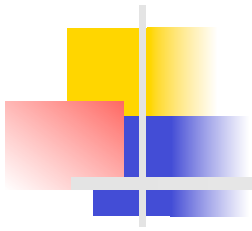
COMP3411-9814- Artificial Intelligence

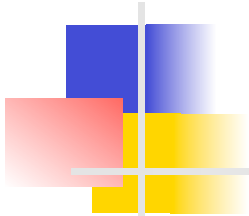


Prolog
Built-in Predicates

2019 – Summer Term

Lecture 4

Tatjana Zrimec



Groups of Built-in Predicates

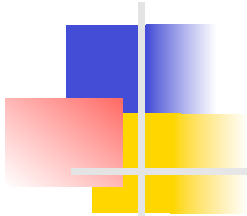
- Testing the type of terms
- Construction and decomposition of terms: =. . , functor, arg, name
- Various types of equality and comparison
- “Database manipulation”: assert, retract
- Control facilities
- *bagof*, *setof* and *findall*
- Input, output

Testing the type of terms

var(X)	succeeds if X is currently instantiated variable
nonvar(X)	X is not a variable or X is instantiated variable
atom(X)	is true if X currently is an atom
integer(X)	is true if X currently stands for an integer
float(X)	is true if X currently stands for a real number
number(X)	is true if X currently stands for a number
atomic(X)	is true if X currently stands for a number or an atom
compound(X)	is true if X currently stands for a compound term (a structure)

Example: Arithmetic Operations

```
...,
number( X),          % Value of  X number?

number( Y),          % Value of  Y number?

 Z is X + Y,         % Then addition it is possible
...
```

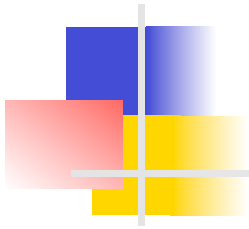


Construction and decomposition of terms: =.. , *functor, arg, name*

Term =.. [Functor, Arg1, Arg2, Arg3, ...] % „univ“

Example: Increase the geometric figure by a factor of 1.5

```
?- Figure = square( 3),      %  square side 3
   ...
   Figure =.. [ Type, Size],
   NewSize is 1.5 * Size,
   NewFigure =.. [ Type, NewSize].

   NewFigure = square( 4.5).     %  square with side 4.5
```



Substitute
the sub-phrase in the New Sub-phrase

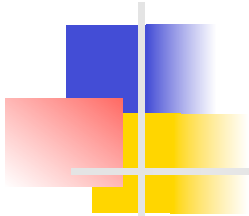
substitute(Subterm, Term, Subterm1, Term1):

if all occurrences of Subterm in Term are substituted with Subterm1 then we get Term1.

?- substitute(sin(x), 2*sin(x)*f(sin(x)), t, F).

F = 2*t*f(t)

```
% Case 1: Substitute whole term
substitute( Term, Term, Term1, Term1) :- !.

% Case 2: Nothing to substitute if Term atomic
        substitute( _, Term, _, Term) :-
            atomic( Term), !.                % Term is a constant

% Case 3: Do substitution on arguments
substitute( Sub, Term, Sub1, Term1) :-
     Term =.. [F | Args],                    % Get arguments
     substlist( Sub, Args, Sub1, Args1),     % Perform  substitution on them
    Term1 =.. [F | Args1].                   % Construct Term1

% substlist( SubTerm, Term_List, NewSubTerm, NewTerm_List)
```



Example - Use of *substitute / 4*

```
?- E0 = (a+b) * (a-b),
    substitute( a, E0, 6, E1),
    substitute( b, E1, 3, E2),
    Value is E2.
```

E1 = (6+b) * (6-b)
E2 = (6+3) * (6-3)
Value = 27

Various types of equality and comparison

X = Y	is true if X and Y match
X == Y	if X and Y are identical
X \== Y	if X and Y are not identical
X @< Y	X is lexicographically smaller then Y , term X precedes term Y by alphabetical or numerical ordering (paul @< peter)

„Database Manipulation“

assert(Clause) **%** add – assert **Clause** to the DB

asserta(Clause) **%** assert **Clause** at the beginning

assertz(Clause) **%** assert **Clause** at the end

retract(Clause) **%**remove **Clause** from the DB

Example: robot world (see Lecture1)

```
% move( X, Y, Z): move block X from Y to Z
move( X, Y, Z) :-                  %move X from Y to Z
      retract( on(X,Y)), !,        % X is no longer on Y
       assertz( on(X,Z)).          % now X is on Z
```

Množice Rešitev - findall, bagof in setof

findall(Object, Condition, List)
List = list of Object objects that satisfy the Condition

bagof(Object, Condition, List) % produce a List of all Objects that satisfy Condition

setof(Object, Condition, List) % produce a sorted List of all Objects that satisfy Condition

Example: robot world (see Lecture1)

```
?- findall( B, on(B,_), L).          % L is a List of all blocks
   L = [a,b,c,d,e]

?- setof( Z:B, B2^( on(B,B2), z(B,Z)), L).    % Block are ordered on Z coord
   L = [ 0:c, 0:d, 0:e, 1:b, 2:c]
```

Procedure findall, bagof in setof

Examples:

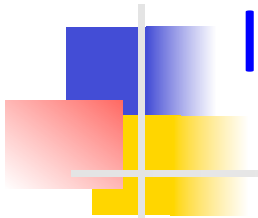
child(joze, ana). child(miha, ana).
child(lili, ana). child(lili, andrej).

?- findall(X, child(X, ana), S).
S = [joze, miha, lili]

?- setof(X, child(X, ana), S).
S = [joze, lili, miha]

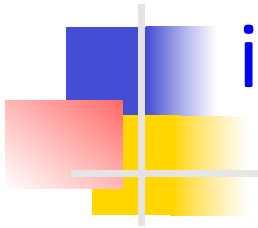
?- findall(X, child(X, Y), S).
S = [joze, miha, lili, lili]

?- bagof(X, child(X, Y), S).
S = [joze, miha, lili]
Y = ana;



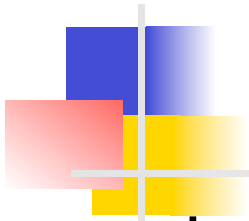
Input, output

```
?- consult(File).
?- see(File).          % File becomes the current input stream
?- see(user).          % user input
?- seen.               % close the current input stream
?- seeing(X).          %binds X to the current input file

?- tell(File).         % File becomes the current output  stream

?- tell(user).         % user output
?- told.               % close the current output stream
?- telling(X).         % binds X to the current output file
```



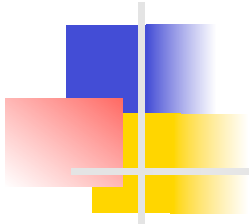
Working with input, output and files

```
?- open/4.                  %
?- close(Datoteka).         %

?- get0(C).                 %.
?- get(C).                  %
?- put(C).                  % write C on the output.

?- read(I).                 % input  to I
?- write(I).                % output I  .
```



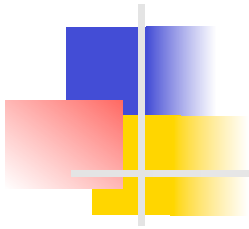
Example

```
write_char(Dat) :-
   see(Dat),
   get0(Char),
   put(Char),
   see(user).

input_char(Dat) :-
    get0(Char),
    tell(Dat),
    put(Char),
    tell(user).
```



Example

```
process(Dat) :-
    seeing(OldStream),
    see(Dat),
    repeat,                          % Repeat procedure !
    read(T),
    process(T),
    T == end_of_file, !,    %
    seen,
    see(OldStream).
```



SWI Prolog Manual - links

SWI Prolog Manual

- 4.17 Input and output

http://www.swi-prolog.org/pldoc/man?section=IO

- 4 Built-in Predicates

http://www.swi-prolog.org/pldoc/man?section=builtin

- 4.39 Debugging and Tracing Programs

http://www.swi-prolog.org/pldoc/man?section=debugger