

For Wednesday

- No reading
- Prolog Handout 3 (will be available Monday)

Exam 1

- Monday
- Take home portion due on Monday
- Questions?

Some List Predicates

- member/2
- append/3

Try It

- `reverse(List,ReversedList)`
- `evenlength(List)`
- `oddlength(List)`

The Anonymous Variable

- Some variables only appear once in a rule
- Have no relationship with anything else
- Can use `_` for each such variable

Arithmetic in Prolog

- Basic arithmetic operators are provided for by built-in procedures:

$+$, $-$, $*$, $/$, `mod`, `//`

- Note carefully:

`?- X = 1 + 2.`

`X = 1 + 2`

`?- X is 1 + 2.`

`X = 3`

Arithmetic Comparison

- Comparison operators:

>

<

>=

=< (note the order: NOT <=)

== (equal values)

!= (not equal values)

Arithmetic Examples

- Retrieving people born 1950-1960:
?- born(Name, Year),
Year >= 1950,
Year =< 1960.
- Difference between = and ::==
?- 1 + 2 ::= 2 + 1.
yes
?- 1 + 2 = 2 + 1.
no
?- 1 + A = B + 2.
A = 2
B = 1

Length of a List

- Definition of length/2

length([], 0).

length([_ | Tail], N) :-
 length(Tail, N1),
 N is 1 + N1.

- Note: all loops must be implemented via recursion

Counting Loops

- Definition of sum/3
sum(Begin, End, Sum) :-
 sum(Begin, End, Begin, Sum).
sum(X, X, Y, Y).
sum(Begin, End, Sum1, Sum) :-
 Begin < End,
 Next is Begin + 1,
 Sum2 is Sum1 + Next,
 sum(Next, End, Sum2, Sum).

The Cut (!)

- A way to prevent backtracking.
- Used to simplify and to improve efficiency.

Negation

- Can't say something is NOT true
- Use a **closed world assumption**
- Not simply means “I can't prove that it is true”

Dynamic Predicates

- A way to write self-modifying code, in essence.
- Typically just storing data using Prolog's built-in predicate database.
- Dynamic predicates must be declared as such.

Using Dynamic Predicates

- assert and variants
- retract
 - Fails if there is no clause to retract
- retractall
 - Doesn't fail if no clauses