# **CS-205 Week3: Arithmetic**

#### Theory

- Introduce Prolog`s built-in abilities for performing arithmetiment Project Exam Help
- Apply these to simple list processing problems, using action using action using action and action and action and action actions are action as a simple list processing problems, and action are action as a simple list processing problems, and action are actions as a simple list processing problems, and action are action as a simple list processing problems, and action are action as a simple list processing problems, and action are action as a simple list processing problems, and action are action as a simple list processing problems.
- Look at tail-recursive predicates and explain why they are more efficient than predicates that are not tail-recursive

#### Exercises

- Exercises of LPN: 5.1, 5.2, 5.3
- Practical work

# **Arithmetic in Prolog**

- Prolog provides a number of basic arithmetic tools Assignment Project Exam Help
- Integer and real numbers https://powcoder.com

#### Arithmetic Add WeChat poweloger

$$2 + 3 = 5$$

$$3 \times 4 = 12$$

$$5 - 3 = 2$$

$$3 - 5 = -2$$

$$4:2=2$$

1 is the remainder when 7 is divided by 2

?-1 is mod(7,2).

# **Example queries**

```
?- 10 is 5+5.
yes
         Assignment Project Exam Help
?- 4 is 2+3.
              https://powcoder.com
no
              Add WeChat powcoder
?-X is 3 * 4.
X = 12
yes
?- R is mod(7,2).
R=1
yes
```

#### Defining predicates with arithmetic

addThreeAndDouble(X, Y):-

Y is (X+3) \* 2. Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

### Defining predicates with arithmetic

```
addThreeAndDouble(X, Y):-
Y is (X+3) * 2.
Assignment Project Exam Help
```

https://powcoder.com

```
?- addThreeAndDouble(2,X).
X=10
yes
```

- It is important to know that +, -, / and \* do not carry out any arithmetic Assignment Project Exam Help
- Expressions such as 3+2, 4-7, 5/5 are ordinary Prolog terms
  - Functor Add We Chat powcoder
  - Arity: 2
  - Arguments: integers

?-X = 3 + 2.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
?- X = 3 + 2.

X = 3+2
yes

Assignment Project Exam Help

https://powcoder.com
?-

Add WeChat powcoder
```

```
?- X = 3 + 2.
X = 3+2
yes Assignment Project Exam Help
```

 $\frac{\text{https://powcoder.com}}{\text{?- 3 + 2 = X.}}$ 

Add WeChat powcoder

```
?-X = 3 + 2.
X = 3+2
     Assignment Project Exam Help
yes
         https://powcoder.com
?-3+2=X.
X = 3+2 Add WeChat powcoder
yes
?-
```

 To force Prolog to actually evaluate arithmetic expressions, we have to use Assignment Project Exam Help

is https://powcoder.com

Add WeChat powcoder just as we did in the other examples

- This is an instruction for Prolog to carry out calculations
- Because this is not an ordinary Prolog predicate, there are some restrictions

?-X is 3 + 2.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
?-X is 3 + 2.
X = 5
     Assignment Project Exam Help
yes
          https://powcoder.com
?-
          Add WeChat powcoder
```

```
?- X is 3 + 2.

X = 5
yes

Assignment Project Exam Help

https://powcoder.com
?- 3 + 2 is X.

Add WeChat powcoder
```

```
?-X is 3 + 2.
X = 5
      Assignment Project Exam Help
yes
           https://powcoder.com
?-3+2 is X.
ERROR: is/2: Artgunivers have not sufficiently instantiated
?_
```

```
?- X is 3 + 2.

X = 5
yes

Assignment Project Exam Help

https://powcoder.com
```

?-3+2 is X.

ERROR: is/2: Artgunivers have not sufficiently instantiated

?- Result is 2+2+2+2+2.

```
?-X is 3 + 2
X = 5
      Assignment Project Exam Help
yes
           https://powcoder.com
?-3+2 is X.
ERROR: is/2: Add Inverte hat not sufficiently instantiated
?- Result is 2+2+2+2+2.
Result = 10
yes
?-
```

### Restrictions on use of is/2

- We are free to use variables on the right hand side of the is predicate Assignment Project Exam Help
- But when Prolog actually carries out the https://powcoder.com evaluation, the variables must be instantiated Withat variable-free Prolog term
- This Prolog term must be an arithmetic expression

#### **Notation**

- Two final remarks on arithmetic expressions Assignment Project Exam Help
  - 3+2, 4/2, 4-5 are just ordinary Prolog terms in a user-friendly notation:
    3+2 is really + (3,2) and seen.
  - Also the **is** predicate is a two-place Prolog predicate

### **Notation**

 Two final remarks on arithmetic expressions Assignment Project Exam Help

- 3+2, 4/2, 4-5 are just ordinary Prolog terms in a user-friendly notation:
3+2 is really + (3,2) and seen.

Also the **is** predicate is a two-place Prolog predicate

?- 
$$is(X,+(3,2))$$
.  
  $X = 5$  yes

### **Arithmetic and Lists**

- How long is a list?
  - The empty list has length: Assignment Project Exam Help zero;
  - A non-empty list has length: one plustle wet traft its tail der

```
len([],0).
len([_|L],N):-
len(L,X),ssignment Project Exam Help
N is X + 1. https://powcoder.com
```

```
?- Add WeChat powcoder
```

```
len([],0).
len([_|L],N):-
    len(L,X),ssignment Project Exam Help
    N is X + 1. https://powcoder.com
```

```
?- len([a,b,c,d,e,[a,t],WeChat powcoder
```

```
len([],0).
len([_|L],N):-
len(L,X),ssignment Project Exam Help
N is X + 1. https://powcoder.com
```

```
?- len([a,b,c,d,e,[a,t]],WeChat powcoder
X=7
yes
?-
```

### Accumulators

- This is quite a good program
  - Easy to understand Exam Help
- Relatively efficient <a href="https://powcoder.com">https://powcoder.com</a>
   But there is another method of finding the length of a list powcoder
  - Introduce the idea of accumulators
  - Accumulators are variables that hold intermediate results

# Defining acclen/3

- The predicate acclen/3 has three arguments Assignment Project Exam Help
  - The list whose length we want to find <a href="https://powcoder.com">https://powcoder.com</a>
     The length of the list, an integer

  - An accumulator, keeping drack of the intermediate values for the length

# Defining acclen/3

- The accumulator of acclen/3
  - Initial value of the accumulator is 0
  - Add 1 to accumulator each time we can recursively take the nead of a list
  - When wedreachtthe empty elist, the accumulator contains the length of the list

```
acclen([],Acc,Length):-
Length = Acc.
Assignment Project Exam Help
acclen([_|L],OldAcc,Length):-
NewAcc is OldAcc + 1,
acclen(L,NewAccV,Length):-
acclen(L,N
```

?-			

```
acclen([],Acc,Length):-

Length = Acc.

Assignment Project Exam Help

acclen([_|L],OldAcc,Length):-

nttps://powcoder.com

NewAcc is OldAcc + 1,

acclen(L,NewAcc\,eerlgth)powcoder
```

?-

```
acclen([],Acc,Length):-

Length = Acc.

Assignment Project ExamptHelphe list

acclen([_|L],O|dAcc,Length):-

nttps://powcoder.com

NewAcc is OldAcc + 1,

acclen(L,NewAccWeerloth)powcoder
```

```
?-
```

```
acclen([],Acc,Acc).

Assignment Project Exam Help
acclen([_|L],OldAcc,Length):-
NewAcc isl@lpAccpewcoder.com
acclen(L,NewAcc,Length).
Add WeChat powcoder
```

?-

```
acclen([],Acc,Acc).

Assignment Project Exam Help
acclen([_|L],OldAcc,Length):-
NewAcc ishthpaccpowcoder.com
acclen(L,NewAcc,Length).
Add WeChat powcoder
```

```
?-acclen([a,b,c],0,Len).
Len=3
yes
?-
```

acclen([],Acc,Acc).

```
?- acclen([a,b,c],0,Len).

Assignment Project Exam Help NewAcc is OldAcc + 1, acclen(L,NewAcc,Length).
```

https://powcoder.com

Add WeChat powcoder

```
?- acclen([a,b,c],0,Len).

/ Assignment Project Exam Help, NewAcc is OldAcc + 1, acclen(L,NewAcc,Length).

https://powcoder.com
```

Add WeChat powcoder

```
?- acclen([a,b,c],0,Len).

/ Assignment Project Exam Help acclen([,L],OldAcc,Length):-
NewAcc is OldAcc + 1,
acclen(L,NewAcc,Length).

/ Add WeChat powcoder
```

```
?- acclen([a,b,c],0,Len).

/ Assignment Project Exam Help acclen(L,NewAcc,Length):-

no ?- acclen([boolcoder):-

Add WeChat powcoder

no ?- acclen([c],2,Len).
```

```
acclen([],Acc,Acc).
?- acclen([a,b,c],0,Len).
         Assignment Project Exam Help ([ |L],OldAcc,Length):-
                                            acclen(L, NewAcc, Length).
            ?-hapen/bookcoden.com
  no
               Add WeChat powcoder ?- acclen(Icl.2,Len).
                                     ?- acclen([],3,Len).
                       no
```

```
acclen([],Acc,Acc).
?- acclen([a,b,c],0,Len).
         Assignment Project Exam Help ([ |L],OldAcc,Length):-
                                            acclen(L, NewAcc, Length).
            ?- hapen/bookcoden.com
  no
               Add WeChat powcoder ?- acclen(Icl.2,Len).
                                     ?- acclen([],3,Len).
                      no
                                     Len=3
                                                          no
```

## Adding a wrapper predicate

```
acclen([],Acc,Acc).

acclen([_|L|,Cki/gc/inergth):roject Exam Help

NewAcc is OldAcc + 1,
acclen(L,NewAcc,Length):

Add WeChat powcoder

length(List,Length):-
acclen(List,0,Length).
```

```
?-length([a,b,c], X).
X=3
yes
```

## **Tail recursion**

- Why is acclen/3 better than len/2?
  - acclen/3 is tail-recursive and len/2 is not

https://powcoder.com

- Difference:
  - In tail recursive predicates the results is fully calculated once we reach the base clause
  - In recursive predicates that are not tail recursive, there are still goals on the stack when we reach the base clause

## Comparison

```
Ien([],0).

len([],NewLength):-
len(L,Length),
NewLength is Aengthy-eChat poweoden(L,NewAcc,Length).
```

```
?- len([a,b,c], Len).
```

```
len([],0).
                          len([_|L],NewLength):-
Assignment Project Exame Helphyth is Length + 1.
```

https://powcoder.com

Add WeChat powcoder

```
?- len([a,b,c], Len).

/ \ len([],0).
len([],NewLength):-
len(L,Length),
xameHeigh is Length + 1.

https://powcoder.com
```

Add WeChat powcoder

```
len([],0).
?- len([a,b,c], Len).
                                           len([_|L],NewLength):-
       ?- len([b,c],Len1), len(L,Length), len is een1+1. Project Exame Helpgth is Length + 1.
  no
                 https://powcoder.com
               ?- len([c], Len2),
       no
                 Actid WeCatat powcoder Len is Len1+1.
                        ?- len([], Len3),
              no
                          Len2 is Len3+1,
                          Len1 is Len2+1,
                           Len is Len1 + 1.
```

```
len([],0).
?- len([a,b,c], Len).
                                        len([_|L],NewLength):-
         len([b,c],Len1), len(L,Length), Lense gement Project Exame Helpgth is Length + 1.
       ?- len([b,c],Len1),
  no
              ?_https://paycoder.com
      no
                 Len1 is Len2+1,
                Ands Wer Chat powcoder
                      ?- len([], Len3),
             no
                        Len2 is Len3+1,
                        Len1 is Len2+1,
                        Len is Len1 + 1.
         Len3=0, Len2=1,
                                          no
          Len1=2, Len=3
```

```
acclen([],Acc,Acc).
?- acclen([a,b,c],0,Len).
         Assignment Project Exam Help ([ |L],OldAcc,Length):-
                                            acclen(L, NewAcc, Length).
            ?- hapen/bookcoden.com
  no
               Add WeChat powcoder ?- acclen(Icl.2,Len).
                                     ?- acclen([],3,Len).
                      no
                                     Len=3
                                                          no
```

#### **Exercises**

- Exercise 5.1
- Exercise 5 1 Project Exam Help
- Exercise 5.3//powcoder.com

Add WeChat powcoder

# **Comparing Integers**

- Some Prolog arithmetic predicates actually do carry out arithmetic by Assignment Project Exam Help themselves
- These are the operators that compare integers<sup>Add</sup> WeChat powcoder

**Arithmetic** 

x > y

## **Comparing Integers**

Anumenc			1 10109
x < y	Assignment Project	ct	Exxam Help
x ≤ y	https://powco	4.	X =< Y
x = y	nups://powco	ue	r <sup>xc</sup> öin <sup>4</sup>
x ≠ y	Add WeChat	nc	&ē\ēder
ху		r	X >= Y

Prolog

X > Y

## **Comparison Operators**

- Have the obvious meaning
- Forcesbeth left pand right hand argument to be evaluated https://powcoder.com

```
?- 2 < 4+1.
yes Add WeChat powcoder
?- 4+3 > 5+5.
no
```

## **Comparison Operators**

- Have the obvious meaning
- Forcesboth leftpand right hand argument to be evaluated https://powcoder.com

```
?- 4 = 4.
yes

Add WeChat powcoder

?- 2+2 = 4.
no

?- 2+2 =:= 4.
yes
```

# **Comparing numbers**

- We are going to define a predicate that takes two arguments, and is true when:
  - Assignment Project Exam Help
     The first argument is a list of integers
  - The sedompsa/pumentderthe highest integer in the list
- Basic idea WeChat powcoder
  - We will use an accumulator
  - The accumulator keeps track of the highest value encountered so far
  - If we find a higher value, the accumulator will be updated

?- accMax([1,0,5,4],0,Max).

Max=5

yes

## Definition of accMax/3

```
accMax([H|T],A,Max):-
  H > A
 accMaxAssignment Project Exam Help
accMax([H|T],A,Mtap);://powcoder.com
  H = < A,
 accMax(T,A,Max). WeChat powcoder
accMax([],A,A).
```

# Adding a wrapper max/2

```
accMax([H|T],A,Max):-
                                 ?- max([1,0,5,4], Max).
  H > A,
                                 Max=5
  accMaxAsiMaxment Project Exesm Help
accMax([H|T],A,Mtaps://powcoder.comax([-3, -1, -5, -4], Max).
  accMax(T,A,Max). MeChat powcoder yes
accMax([],A,A).
                                  ?_
max([H|T],Max):-
  accMax(T,H,Max).
```

# **Summary of this lecture**

- In this lecture we showed how Prolog does arithmetic Assignment Project Exam Help
- We demonstrated the difference https://powcoder.com between tail-recursive predicates and predicates that are more tail-recursive
- We introduced the programming technique of using <u>accumulators</u>
- We also introduced the idea of using wrapper predicates

### **Next lecture**

- Yes, more lists!
  - Defining the append/3 a predicate that concatenates two lists
  - Discuss the idea of reversing a list, first naively ausing append a then with a more efficient way using accumulators