Introduction to Artificial Intelligence
Lecture 7: Logical Function in pyDatalog

Logical Functions in pyDatalog:
A function maps an input to an output, similar to mathematical functions.

Functions are especially useful when:

- You need **numerical results**
- You want **recursion** (factorial, Fibonacci, totals)
- You want to **use computed values inside rules**
- You can still define facts using function, but you should take into consideration that the function map only one output to each input.

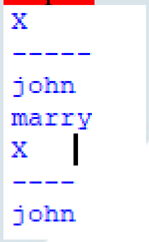
Example:

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('manager_fact,manager_function,X')
# Add some facts
+manager_fact('tom','marry')
+manager_fact('tom','john')

manager_function['tom']='marry'
manager_function['tom']='john'

print(manager_fact('tom',X))
print(manager_function['tom']==X)
```

output:



```
X
-----
john
marry
X
----
john
```

Note: for the second output, the function only outputs one result, as it replaces any old facts with the new update.

Example: Factorial Function (Numerical)

Here, Factorial[N] represents the factorial of N.

```
from pyDatalog import pyDatalog
pyDatalog.create_terms("N, Factorial")

Factorial[1] = 1
Factorial[N] = N * Factorial[N-1]

print(Factorial[5] == N)
```

Output:

```
N = 120
```

This works because:

- Factorial[1] == 1 is a **base fact**
- Factorial[N] == N * Factorial[N-1] is a **recursive rule**
- The function computes values automatically when queried

Using Functions Inside Rules

Functions can be used to help define logical conditions.

Example idea:

- A person is “senior” if their age is greater than 60

```
from pyDatalog import pyDatalog as py
py.create_terms('age,senior,X,Y')
age['john'] == 65
(senior[X]==True) <= (age[X]==Y) & (Y>60)
print(senior['john']==X)
```

Output:

True

Aggregate Functions in pyDatalog

Aggregate functions act like a predefined function in the library for a special purpose.

1. len_(Y) — Counting values

len_(Y) counts how many values Y can take.

Example: Number of people each manager manages
```
from pyDatalog import pyDatalog as py
py.create_terms('manager, number_of_managers, X, Z,Y')

+manager('tom', 'mary')
+manager('sam', 'mary')
+manager('tom', 'john')

(number_of_managers[X] == len_(Z)) <= manager(X, Z)

print(number_of_managers['tom'] == Y)
```

Output:

Y = 2

---

2. min_(Y,order_by=) — Minimum value

```
Finds the smallest value among results.

py.create_terms('price, cheapest, X, P,C')

+price('shop1', 3.5)
+price('shop1', 2.0)

(cheapest[X] == min_(P,order_by=P)) <= price(X, P)

print(cheapest['shop1'] == C)
```

Output:

C = 2.0

---

3. max_(Y,order_by=) — Maximum value

Finds the largest value among results.

```
py.create_terms('score, highest, X, S,H')

+score('alice', 70)
+score('alice', 90)

(highest[X] == max_(S,order_by=S) )<= score(X, S)

print(highest['alice'] == H)
```

```
Output:
H = 90
```