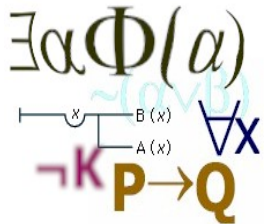


# Αναδρομή

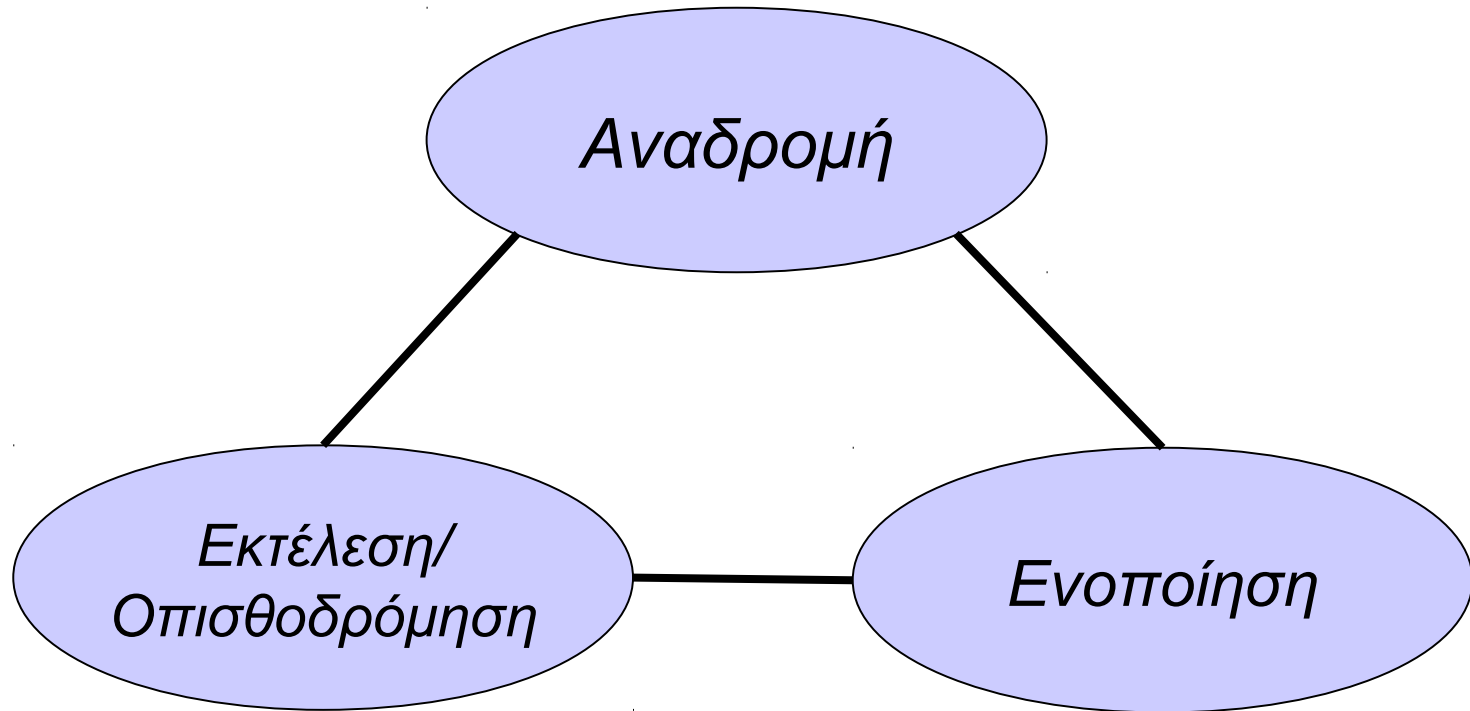
Ηλίας Σακελλαρίου



# Δομή Παρουσίασης

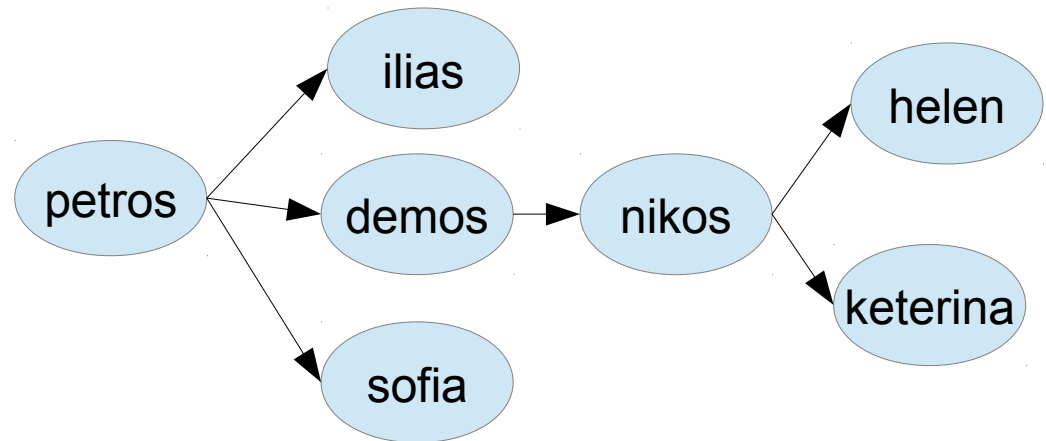
- Αναδρομικοί Ορισμοί
- Αρχή της Αναδρομής
- Παραδείγματα
- Αριθμητικές Πράξεις στην Prolog
- Παράδειγμα Συμβολικής Παραγωγήσις

# Prolog Programming Skills



# Παράδειγμα Κίνητρο: Κοινωνικό Δίκτυο

- Έστω ότι γνωρίζετε **προτάσεις** (endorsements) μεταξύ των χρηστών:
- Σχέση endorse/2.  
`endorse(petros,ilias).`  
`endorse(petros,demos).`  
`endorse(petros,sofia).`  
`endorse(demos,nikos).`  
`endorse(nikos,helen).`  
`endorse(nikos,katerina).`



# Σύνδεση μεταξύ Χρηστών: Σχέση linked\_by\_E

- Δύο χρήστες **Endorser** και **User** συνδέονται αν:
  - ο **Endorser** έχει προτείνει τον **User** (Α' Βαθμός)
  - ο **Endorser** έχει προτείνει τον **X**, ο οποίος έχει προτείνει τον **User** (Β' Βαθμός),
  - ο **Endorser** έχει προτείνει τον **X**, ο οποίος έχει προτείνει τον **Y**, ο οποίος έχει προτείνει τον **User** (Γ' Βαθμός),
  - κλπ....

# Υλοποίηση

*linked\_by\_E(Endorser, User):-  
endorse(Endorser, User).*

*linked\_by\_E(Endorser, User):-  
endorse(Endorser, X), endorse(X, User).*

*linked\_by\_E(Endorser, User):-  
endorse(Endorser, X), endorse(X, Y), endorse(Y, User).*

...

# Υλοποίηση (καλύτερη)

*linked\_by\_E(Endorser, User):-  
endorse(Endorser, User).*

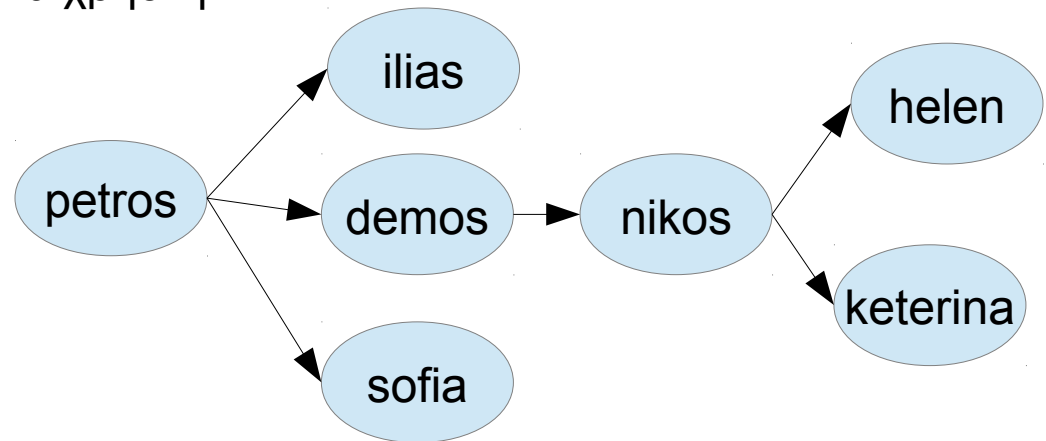
*linked\_by\_E(Endorser, User):-  
endorse(Endorser, UserX),  
linked\_by\_E(UserX, User).*

# Παράδειγμα: Σχέση linked\_by\_E

- **linked\_by\_E(Endorser,User)**, αληθής όταν ο Endorser συνδέεται μέσω "προτάσεων" με τον User. Άρα:

- Ο **Endorser** έχει προτείνει τον **User**
- Ο **Endorser** έχει προτείνει ένα χρήστη X που συνδέεται με τον **User**.

endorse(petros,ilias).  
endorse(petros,demos).  
endorse(petros,sofia).  
endorse(demos,nikos).  
endorse(nikos,helen).  
endorse(nikos,katerina).



linked\_by\_E(Endorser,User):- endorse(Endorser,User).

linked\_by\_E(Endorser,User):- endorse(Endorser,UserX), linked\_by\_E(UserX,User).



# Ορισμοί στα Μαθηματικά

## ■ Αναλυτικοί

- Ο ορισμός δίδεται χωρίς την χρήση του ίδιου του ορισμού.
- $n! = 1 * 2 * 3 * 4 * \dots * (n-1) * n$

## ■ Αναδρομικοί

- Ο ορισμός βασίζεται στον ίδιο τον ορισμό
- $0! = 1$  και  $n! = (n-1)! * n, n > 0$ .

# Αναδρομή

## ■ Αναδρομικοί Ορισμοί

$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! * n & n > 0 \end{cases}$$

*Βασική Περίπτωση*

## ■ Βασική Περίπτωση

□ Μη-αναδρομικό μέρος

## ■ Γενική Περίπτωση

□ Αναδρομικό μέρος

*Γενική Περίπτωση*

# Πως Λειτουργεί η Αναδρομή

## ■ Υπολογισμός του 5!

$$5 > 0 \text{ άρα } 5! = 4! * 5$$

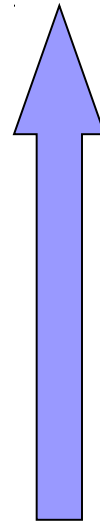
$$4 > 0 \text{ άρα } 4! = 3! * 4$$

$$3 > 0 \text{ άρα } 3! = 2! * 3$$

$$2 > 0 \text{ άρα } 2! = 1! * 2$$

$$1 > 0 \text{ άρα } 1! = 0! * 1$$

$$0! = 1$$



$$5! = 24 * 5 = 120$$

$$4! = 6 * 4 = 24$$

$$3! = 2 * 3 = 6$$

$$2! = 1 * 2 = 2$$

$$1! = 1 * 1 = 1$$

# Αρχή της Αναδρομής

- Για την επίλυση ενός προβλήματος τάξης  $N$ :
  - Υποθέτουμε ότι η λύση για **τάξη  $N-1$**  υπάρχει και “χτίζουμε” την λύση για **τάξη  $N$**  βασιζόμενοι σε αυτή (**γενική περίπτωση**).
  - Δηλώνουμε τη λύση για την **βασική περίπτωση** (-εις).
    - Συνήθως για τάξη  $N=1$ .

# Ιδιότητες/Χαρακτηριστικά της Αναδρομής

- Οι αναδρομικοί ορισμοί βασίζονται στην **επαγωγή (induction)**.
- Δηλωτικοί
  - Δηλώνουμε ΠΩΣ ορίζεται η έννοια/σχέση και όχι πως θα υπολογιστεί.
  - ... κατάλληλη για λογικό προγραμματισμό
- Μεγάλη εκφραστικότητα - απλότητα

# Μεθοδολογία

- Αναλύουμε τις περιπτώσεις του προβλήματος σε δύο ομάδες
  - “τετριμμένες” / “οριακές” περιπτώσεις.
  - “γενικές” περιπτώσεις, όπου οι λύσεις δημιουργούνται ως συνδυασμός απλών περιπτώσεων

# Υλοποίηση (με αναδρομή)

*linked\_by\_E(Endorser, User):-  
endorse(Endorser, User).*

Οριακή  
Περίπτωση  
(Βασική -  
ΜΗ αναδρομική)

*linked\_by\_E(Endorser, User):-  
endorse(Endorser, UserX),  
linked\_by\_E(UserX, User).*

Γενική  
Περίπτωση  
(Αναδρομική)

# Αναδρομή vs. Επανάληψη

- Η standard Prolog **δεν** έχει δομές επανάληψης
  - **while, for, until,...**
- Η αναδρομή αντικαθιστά την επανάληψη
  - Κάθε αναδρομικός ορισμός έχει τον αντίστοιχό του επαναληπτικό ορισμό
  - *(μόνο που συνήθως είναι περισσότερο πολύπλοκος και σαφώς λιγότερο κομψός)*
- ***Speed vs Elegance***
  - Τεχνικές όπως tail recursion optimization βελτιώνουν τουλάχιστον τις ανάγκες σε μνήμη.



# Παράδειγμα: Παραγοντικό

- Υπολογισμός Παραγοντικού

- Αναδρομικός Ορισμός

$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! * n & n > 0 \end{cases}$$

- Αναλυτικός Ορισμός (επανάληψη)

*fact:=1;*

*for i=1 to N*

*fact:=fact\*i;*

# Αφηρημένο Σχήμα Κώδικα (απλό)

*foo( X , ..., Z).*

*Βασική Περίπτωση*

*foo( X, ... ,Z):-*

*Γενική Περίπτωση*

*...*

*foo( NewX,...,Z).*

# Αφηρημένο Σχήμα Κώδικα

- Περισσότερες από μια βασικές/γενικές περιπτώσεις

*foo( X ,..., Z).*

*foo( X,... ,Z').*

*...*

*foo( X,... ,Z'').*

*foo( X,... ,Z):-*

*...*

*foo( NewX,...,Z).*

*foo( X,... ,Z):-*

*...*

*foo( NewX,...,Z).*

*Βασικές Περιπτώσεις*

*Γενικές Περιπτώσεις*

# Prolog Κώδικας

*factorial( 0,1 ).*

*factorial( N, Fact):-*

*factorial( N-1, SFact),*

*Fact = SFact\*N.*

**Error**

**Error**

$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! * n & n > 0 \end{cases}$$

# Αριθμητικές Πράξεις στην Prolog

## ■ Ενσωματωμένο κατηγορημα *is/2*

*?- X is 5 + 3.*

8

*?- X is 5 - 3.*

2

*?- X is 5 \* 3.*

15

*?- X is 5 / 3.*

1.66667

*?- X is 5 // 3*

1

*?- X is 5 mod 3*

2

...

*?- X is Y + 3.*

*instantiation fault*

*<error> !!!!*

# Άλλες αριθμητικές Πράξεις

- Τελεστές Σύγκρισης

- $>$ ,  $<$ ,  $<=$ ,  $>=$

- Τελεστές (αριθμητικής) ισότητας ανισότητας

- $=:=$ ,  $=\backslash=$

- **Προσοχή:** Όλες οι μεταβλητές πρέπει να έχουν πάρει τιμές.

- Πρόβλημα έκφρασης αριθμητικών εκφράσεων στη Λογική οδήγησε αρχικά σε Υποστήριξη περιορισμών!

# Prolog Code (corrected)

*factorial( 0, 1 ).*

*factorial( N, Fact):-*

*NN is N-1,*

*factorial( NN, SFact),*

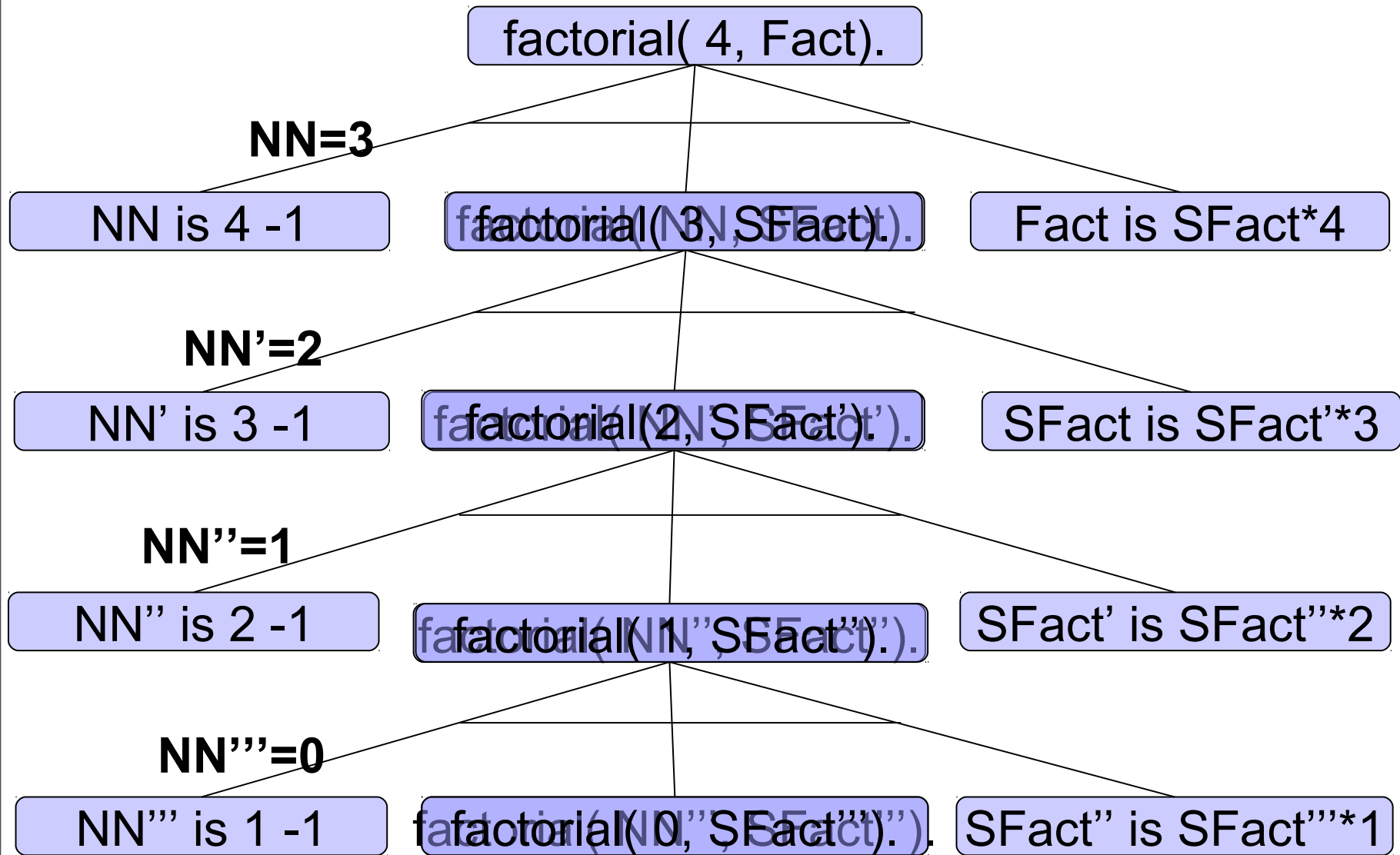
*Fact is SFact\*N.*

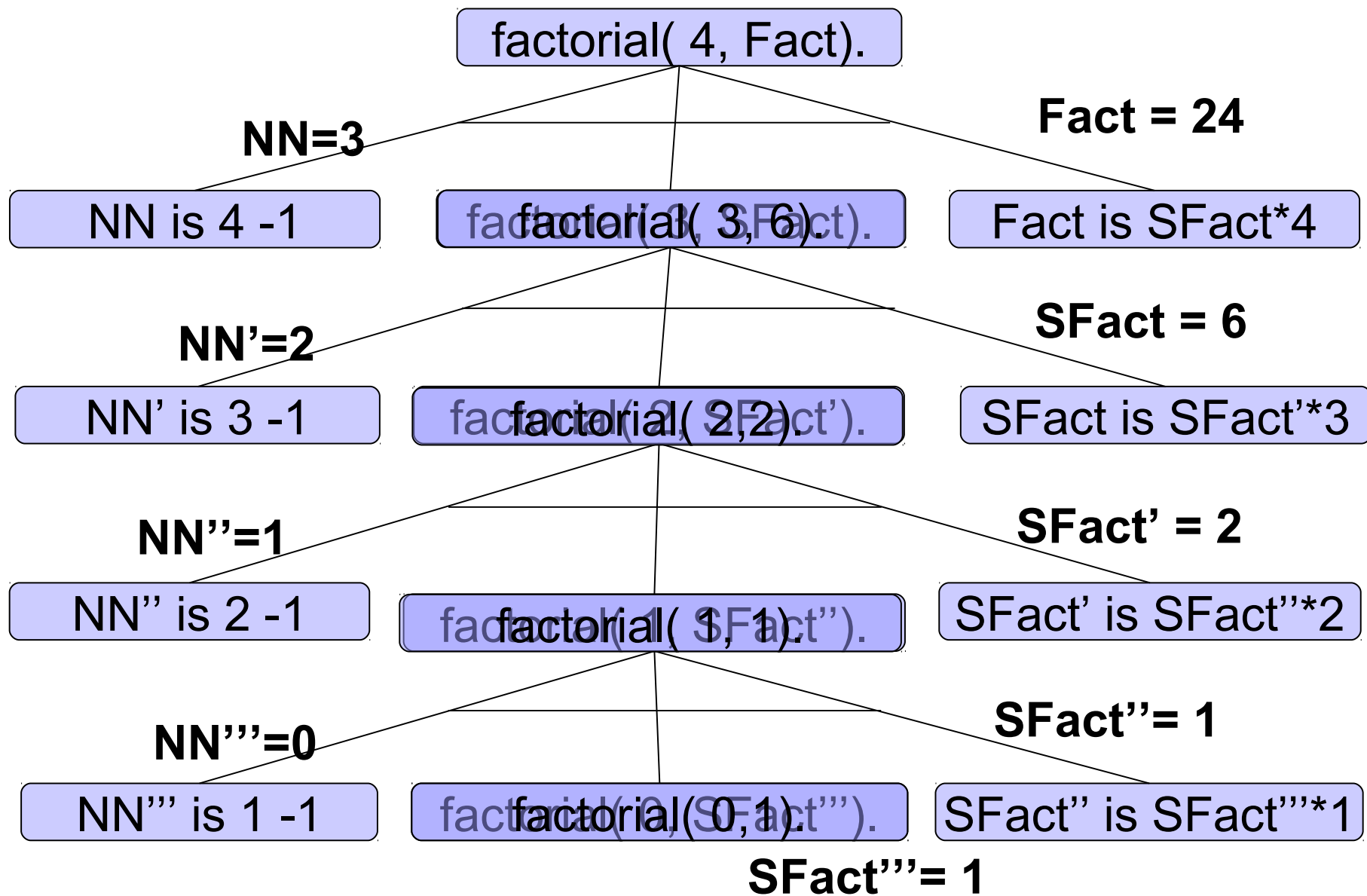
$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! * n & n > 0 \end{cases}$$

# Tracing μιας αναδρομικής κλήσης

- ?-factorial( 4, Fact).







```
factorial( 4, 24).
```

Fact=24

yes

# Ατέρμονες Βρόγχοι

- Ιδιαίτερη προσοχή θα πρέπει να δοθεί στη σειρά των προτάσεων του κατηγορήματος.

*factorial( N, Fact):-*

*NN is N-1,*

*factorial( NN, SFact),*

*Fact is SFact\*N.*

← *Infinite Loop*

*factorial( 0, 1 ).*

# “Χρυσός” κανόνας της Αναδρομής

Πάντα τοποθετούμε στον κώδικα τις προτάσεις (clauses) που περιγράφουν τις βασικές περιπτώσεις, **ΠΡΙΝ** από εκείνες που περιγράφουν την γενική περίπτωση.

# Ατέρμονες Βρόγχοι

- Ένας καλά ορισμένος κανόνας δεν έχει προβλήματα

*factorial( N, Fact):-*

***N>0**, NN is N-1,*

*factorial( NN, SFact),*

*Fact is SFact\*N.*

*factorial( 0,1 ).*

# Ατέρμονες Βρόγχοι:

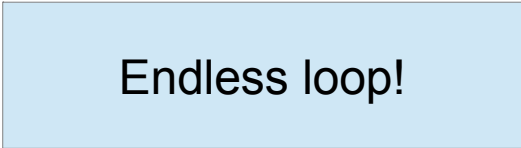
## Σχέση linked\_by\_E

- Προσοχή στην σειρά των κλήσεων!

```
endorse(petros,ilias).  
endorse(petros,demos).  
endorse(petros,sofia).  
endorse(demos,nikos).  
endorse(nikos,helen).  
endorse(nikos,katerina).
```

```
linked_by_E_loop(Endorser,User):-  
    endorse(Endorser,User).
```

```
linked_by_E_loop(Endorser,User):-  
    linked_by_E_loop(Endorser,UserX),  
    endorse(UserX,User).
```



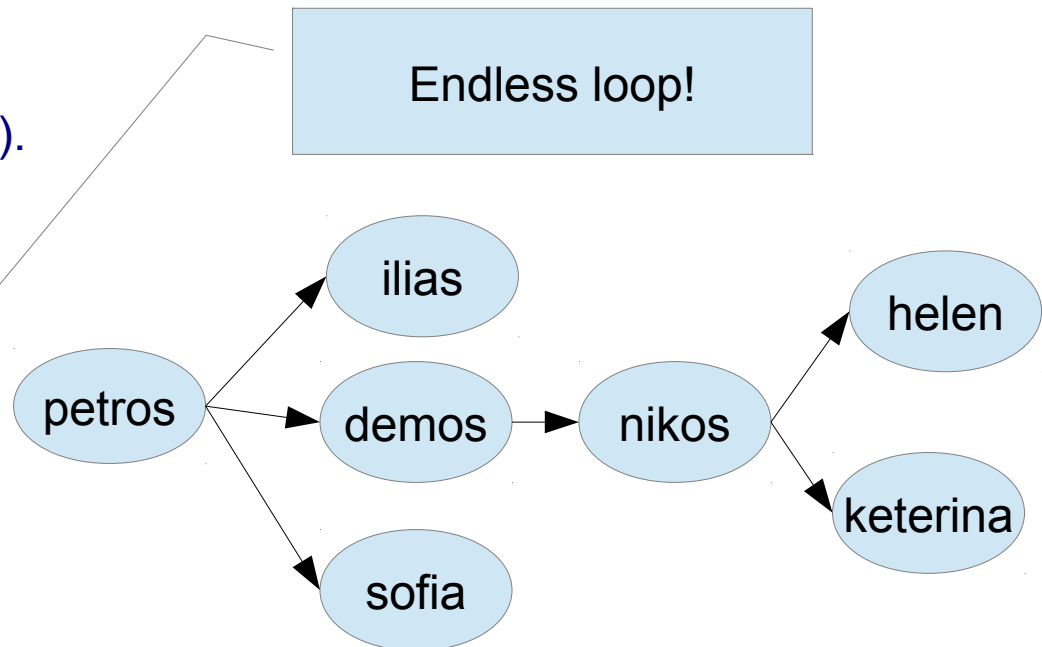
Endless loop!

# Ατέρμονες Βρόγχοι: Σχέση endorse\_link/2

- Μπορεί να γραφεί η σχέση με ένα μόνο κατηγορημα;

```
endorse_link(petros,ilias).  
endorse_link(petros,demos).  
endorse_link(petros,sofia).  
endorse_link(demos,nikos).  
endorse_link(nikos,helen).  
endorse_link(nikos,katerina).
```

```
endorse_link(X,Y):-  
    endorse_link(X,Z),  
    endorse_link(Z,Y).
```





# Συμβολική Παραγωγή

- Εύρεση της πρώτης παραγώγου μιας συνάρτησης (απλές περιπτώσεις):
  - $\sin(x)/dx = \cos(x)$
  - $(x*y)/dx = x*0 + y*1 = y$
  - $(x+2)/dx = 1$
  - $(2*x+x+\cos(x))/dx = 2 + 1 - \sin(x)$

# Συμβολική Παραγωγή

- Κανόνες για την εύρεση της πρώτης παραγώγου

$$dx / dx = 1$$

$$dc / dx = 0$$

$$d\sin(x) / dx = \cos(x)$$

$$d\cos(x) / dx = -\sin(x)$$

$$d(u+v) / dx = du/dx + dv/dx$$

$$d(u*v)/dx = v*du/dx + u*dv/dx$$

# Ενσωματωμένα κατηγορήματα που αφορούν “τύπο” τιμής

- Build-in Predicates

- *atomic( X )*

- Πετυχαίνει αν το  $X$  είναι άτομο (σταθερά)

- *var( X )*

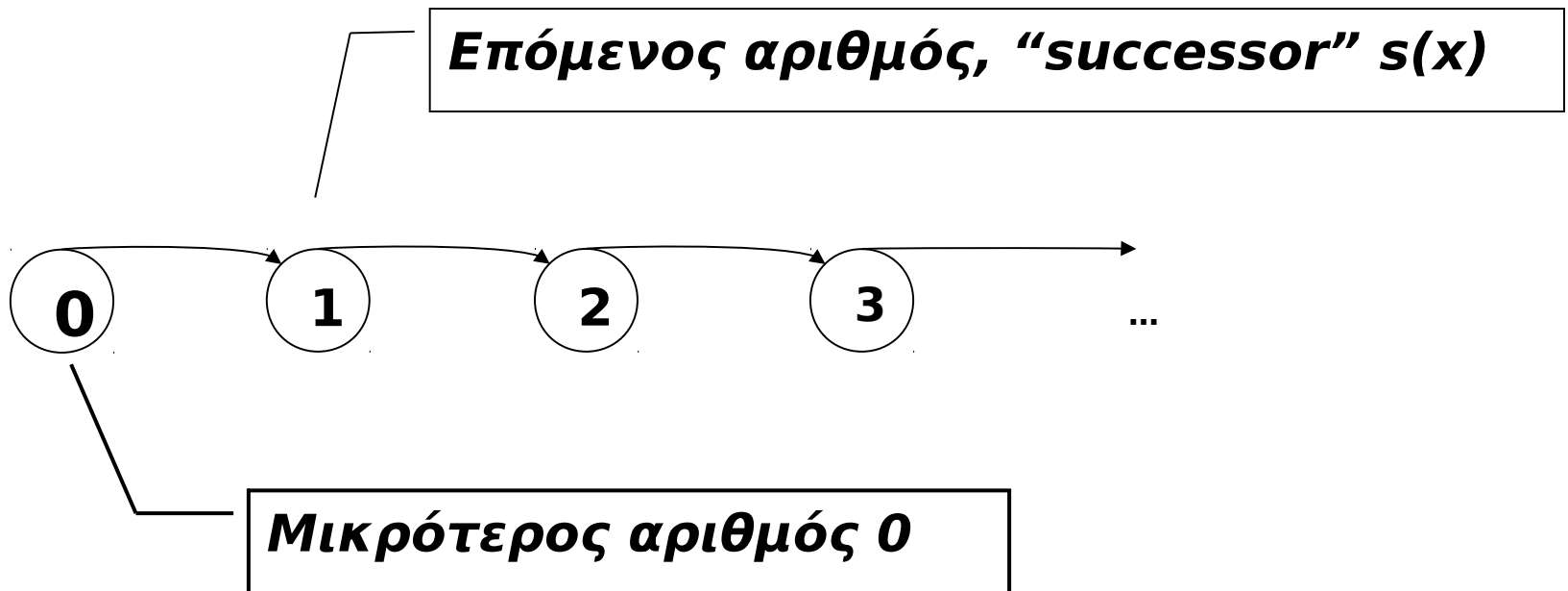
- Πετυχαίνει αν το  $X$  είναι **ελεύθερη** μεταβλητή

- *compound( X )*

- Πετυχαίνει αν το  $X$  είναι σύνθετος όρος.

# Peano Αριθμητική

- Αναπαράσταση φυσικών αριθμών στη Λογική



# Φυσικοί Αριθμοί Peano

- Ορισμός ενός φυσικού αριθμού
  - Το 0 είναι φυσικός αριθμός
  - Ένας successor (επόμενος) αριθμός ενός φυσικού αριθμού είναι φυσικός αριθμός.
- Πρόσθεση
  - $x + 0 = x$
  - $x + s(y) = s(x + y)$
- Γινόμενο
  - $x * 0 = 0$
  - $x * s(y) = (x * y) + x$