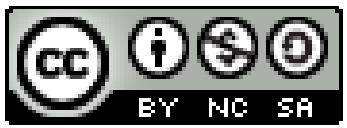
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Υπολογιστική Λογική και Λογικός Προγραμματισμός

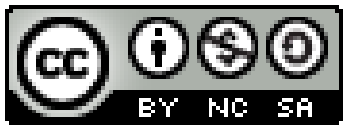
Ενότητα 4: Λογικός Προγραμματισμός: η γλώσσα Prolog: Εισαγωγή, Ιστορική Αναδρομή, Σύνταξη, Εκτέλεση Προγραμμάτων, Αναδρομή.

Νίκος Βασιλειάδης, Αναπλ. Καθηγητής
Τμήμα Πληροφορικής



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Λογικός Προγραμματισμός: η γλώσσα Prolog: Εισαγωγή, Ιστορική Αναδρομή, Σύνταξη, Εκτέλεση Προγραμμάτων, Αναδρομή.

Ιστορική Αναδρομή (1/4)

- Η *Prolog* (Programming in Logic) είναι μια συμβολική γλώσσα προγραμματισμού που βασίζεται στην κατηγορηματική λογική.
- Η ανάπτυξη της ξεκίνησε την δεκαετία του '70 στην Ευρώπη.
- Η πρώτη υλοποίηση οφείλεται στην γαλλική ερευνητική ομάδα του Alain Colmerauer στο πανεπιστήμιο Luminy της Μασσαλίας, η οποία ανέπτυξε ένα πρόγραμμα απόδειξης θεωρημάτων (theorem prover) για επεξεργασία φυσικής γλώσσας.

Ιστορική Αναδρομή (2/4)

- Βασίστηκε στις εργασίες του Robert Kowalski, ο οποίος απέδειξε ότι ένα υποσύνολο της λογική πρώτης τάξης (*προτάσεις Horn*) μπορούν να χρησιμοποιηθεί για μια νέα γενιά γλωσσών προγραμματισμού.

- Και οι δύο προηγούμενες εργασίες βασίστηκαν στην *αρχή της ανάλυσης* (Resolution Principle), μια γενική αποδεικτική διαδικασία για την λογική πρώτης τάξης, που είχε προταθεί από τον Robinson, στα μέσα της δεκαετίας του '60.

Ιστορική Αναδρομή (3/4)

- Σημαντικό ρόλο στη διάδοσή της αποτέλεσε η υλοποίηση ενός μεταφραστή / διερμηνευτή της γλώσσας από τον D.H.D Warren στο Πανεπιστήμιο του Εδιμβούργου (1977).
  - Αποτέλεσε την πρώτη αποδοτική υλοποίηση, αποδεικνύοντας ότι είναι δυνατή η χρησιμοποίηση της σαν μια γενική γλώσσα προγραμματισμού, γεγονός το οποίο ήταν υπό αμφισβήτηση από πολλούς επιστήμονες ιδίως στην Αμερική.
  - Η επιτυχία του μεταφραστή αυτού ήταν τόσο μεγάλη ώστε αποτέλεσε πρότυπο για τις επόμενες υλοποιήσεις, καθιερώνοντας παράλληλα τη σύνταξη της γλώσσας (Edinburgh Syntax).

Ιστορική Αναδρομή (4/4)

- Πολύ σημαντικός παράγοντας στην εξέλιξη της γλώσσας, ήταν και η υιοθέτηση της από τον Ιαπωνικό πρόγραμμα υπολογιστών πέμπτης γενιάς (Fifth Generation Computing).

Διαφορές διαδικαστικών γλωσσών και Prolog (1/2)

- Σε ένα οποιοδήποτε πρόγραμμα, διακρίνουμε το τμήμα της λογικής και το τμήμα του ελέγχου.
- Σύμφωνα με την κλασική εξίσωση του Kowalski:

πρόγραμμα = λογική + έλεγχος.

Διαφορές διαδικαστικών γλωσσών και Prolog (2/2)

- Στο συμβατικό (διαδικαστικό) προγραμματισμό (π.χ. C), το τμήμα της λογικής και το τμήμα του ελέγχου είναι αλληλένδετα και δεν διαχωρίζονται.
  - Ο προγραμματιστής πρέπει να καθορίσει επακριβώς τη ροή ελέγχου του προγράμματος, ανάλογα με τη λογική και τα διαθέσιμα δεδομένα του προβλήματος.
- Στην Prolog, γίνεται διαχωρισμός λογικής-ελέγχου.
  - Χρειάζεται να περιγραφεί μόνο η λογική του προς επίλυση προβλήματος ενώ ο έλεγχος αφήνεται στο σύστημα.

Προγράμματα στην Prolog

- Η λογική ενός προβλήματος, δηλαδή ενός προγράμματος, στην Prolog είναι ένα σύνολο προτάσεων που περιγράφει τα δεδομένα του προβλήματος και τις σχέσεις που τα συνδέουν.
- Οι προτάσεις αυτές λέγονται προτάσεις Horn και αποτελούν υποσύνολο της κατηγορηματικής λογικής πρώτης τάξης.
- Υπάρχουν δύο είδη προτάσεων:
  - τα γεγονότα και
  - οι κανόνες.

Γεγονότα

- Τα γεγονότα εκφράζουν σχέσεις ανάμεσα στα αντικείμενα και αποτελούν κατά ένα τρόπο τα δεδομένα του προβλήματος.
- Για παράδειγμα, αν θέλουμε να δηλώσουμε ότι "*Ο Γιώργος είναι ο πατέρας της Μαρίας*", εισάγουμε στην Prolog ένα γεγονός της μορφής:

father(george,mary).

  - Αντιστοιχεί στην ακόλουθη έκφραση στη μορφή Kowalski

→father(george,mary).

Κανόνες

- Οι κανόνες εκφράζουν γενικότερες σχέσεις ανάμεσα στα αντικείμενα, οι οποίες ορίζονται με τη βοήθεια άλλων σχέσεων.
- Για παράδειγμα, η σχέση *γονιός*, ορίζεται με δύο κανόνες:
  - "*Ο Χ είναι γονιός του Υ εάν ο Χ είναι πατέρας του Υ*" και
  - "*Η Χ είναι γονιός του Υ εάν η Χ είναι μητέρα του Υ*",
- Σε Prolog γράφονται ως εξής:

parent(X,Y) :- father(X,Y).

parent(X,Y) :- mother(X,Y).

- Τα **X**, **Y** είναι μεταβλητές.
- Το σύμβολο "**:-**" αντιπροσωπεύει το λογικό ΕΑΝ.
- Σε μορφή Kowalski

father(X,Y) → parent(X,Y) **mother(X,Y) → parent(X,Y)**

Αλληλεπίδραση με την Prolog

- Η Prolog είναι κατεξοχήν μια *διερμηνευόμενη* γλώσσα (*interpreted*).
- Ο χρήστης αλληλεπιδρά με το σύστημα, θέτοντας *ερωτήσεις* (*queries*) ή *στόχους* (*goals*), μετά το προτρεπτικό σήμα (prompt) "**?-**"
  - Το σύστημα προσπαθεί να απαντήσει βάσει γεγονότων και κανόνων.
- Οι απαντήσεις που λαμβάνει είναι είτε **yes**/**no** (Ναι / Όχι).
  - Αν η ερώτηση περιέχει μεταβλητές, τότε η απάντηση περιλαμβάνει τις κατάλληλες τιμές για τις μεταβλητές αυτές, ώστε η ερώτηση να δέχεται καταφατική απάντηση.
  - Σε περίπτωση που υπάρχουν περισσότερες από μία απαντήσεις, ο χρήστης μπορεί να τις λάβει πατώντας το πλήκτρο "**;**".

Παράδειγμα

- Σχέσεις ανάμεσα στα μέλη μιας οικογένειας.

father(george,mary).
father(george,nick).
father(peter,marina).

mother(helen,mary).
mother(helen,nick).
mother(ann,marina).

parent(X,Y) :- father(X,Y).

parent(X,Y) :- mother(X,Y).

- Έστω ότι οι παραπάνω προτάσεις είναι αποθηκευμένες σε ένα απλό αρχείο κειμένου με όνομα "**family.pl**".
- Για να "φορτωθούν" πρέπει να δοθεί η εντολή **consult**, συνοδευόμενη από το όνομα του αρχείου,:

?- consult('family.pl').

Ερωτήσεις – Απαντήσεις (1/4)

- *Είναι ο **george** πατέρας της **mary**;*

?- father(george,mary).

yes

- *Είναι ο **george** πατέρας της **marina**;*

?- father(george,marina).

no

- *Ποιος (**X**) έχει πατέρα τον **peter**;*

?- father(peter,X).

X=marina

- *Ποια (**X**) είναι η μητέρα της **marina**;*

?- mother(X,marina).

X=ann

Μορφή Kowalski

father(george,mary)→

Ερωτήσεις – Απαντήσεις (2/4)

- *Ποιανού (**X**) είναι γονιός ο **george**;*

?- parent(george,X).

X=mary;

X=nick;

no

- Στην περίπτωση αυτή υπάρχουν δύο απαντήσεις.
  - Προέρχονται από τα δύο σχετικά γεγονότα **father(george,…)**
  - Επιστρέφεται πρώτα η απάντηση **X=mary** και αν ο χρήστης πιέσει το πλήκτρο "**;**" επιστρέφεται η απάντηση **X=nick**.
  - Αν ο χρήστης πιέσει ξανά "**;**" η Prolog θα απαντήσει **no**, η οποία σημαίνει "*δεν υπάρχουν άλλες εναλλακτικές απαντήσεις*".

Ερωτήσεις – Απαντήσεις (3/4)

- *Ποιανού (**X**) είναι γονιός ο **paul**;*

?- parent(paul,X).

no

- *Ποιοι (**X**) είναι οι γονείς του **nick**;*

?- parent(X,nick).

X=george;

X=helen;

no

  - Οι δύο απαντήσεις προέρχονται από τους δύο κανόνες **parent(X,Y)**

Ερωτήσεις – Απαντήσεις (4/4)

- Ποια είναι τα ζεύγη ατόμων **X**, **Y** , για τα οποία η **X** είναι μητέρα του **Y**;

?- mother(X,Y).

X=helen, Y=mary;

X=helen, Y=nick;

X=ann, Y=marina;

no

  - Οι τιμές των X, Y πρέπει να συνυπάρχουν στο ίδιο γεγονός.

Σύνθετες Ερωτήσεις (1/3)

- Εκτός από απλές ερωτήσεις, υπάρχουν και οι σύνθετες, οι οποίες χωρίζονται μεταξύ τους με κόμμα "," το οποίο αντιστοιχεί στον λογικό τελεστή AND.
- Η απάντηση σε μια σύνθετη ερώτηση είναι καταφατική μόνο αν αληθεύουν όλες οι επιμέρους κλήσεις που την αποτελούν.
- *Οι **mary** και **nick** έχουν κοινό πατέρα; Αν ναι, ποιος (**X**) είναι αυτός;*

?- father(X,mary),father(X,nick).

X = george

Σύνθετες Ερωτήσεις (2/3)

?- father(X,mary), father(X,nick).

X = george

- Η *κοινή μεταβλητή* (*shared variable*) **X** εκφράζει τη συνθήκη ότι οι **mary** και **nick** έχουν κοινό πατέρα.
- Οι μεταβλητές στην Prolog όταν εμφανίζονται παραπάνω από μία φορά σε μία ερώτηση, παίρνουν τιμή την πρώτη φορά που εμφανίζονται και στη συνέχεια διατηρούν αυτήν την τιμή σε όλες τις επόμενες εμφανίσεις τους στην ίδια ερώτηση.

Σύνθετες Ερωτήσεις (3/3)

- *Ποια είναι τα ζεύγη ατόμων **X**, **Y** , τα οποία έχουν για μητέρα τους την **helen**;*

?- mother(helen,X),mother(helen,Y).

X=mary, Y=mary;

X=mary, Y=nick;

X=nick, Y=mary;

X=nick, Y=nick;

no

- Η χρήση δύο διαφορετικών μεταβλητών δε συνεπάγεται κατ' ανάγκη ότι αυτές πρέπει να πάρουν διαφορετικές τιμές.

Κανόνες με Σύνθετες Συνθήκες (1/2)

- Όπως υπάρχουν οι σύνθετες ερωτήσεις, έτσι και οι συνθήκες των κανόνων μπορούν να είναι σύνθετες.
  - Πρέπει να συναληθεύουν δύο ή παραπάνω προϋποθέσεις προκειμένου να αποδεικνύεται το συμπέρασμα .
  - Οι προϋποθέσεις χωρίζονται μεταξύ τους με κόμμα "," το οποίο αντιστοιχεί στον λογικό τελεστή AND.
- Παράδειγμα: Πότε δύο άνθρωποι είναι αδέλφια? Όταν έχουν κοινό γονέα.

sibling(X,Y) :- parent(Z,X), parent(Z,Y).

Κανόνες με Σύνθετες Συνθήκες (2/2)

sibling(X,Y) :- parent(Z,X), parent(Z,Y).

?- sibling(mary,george).

yes

?- sibling(mary,X).

X = mary ;

X = nick ;

X = mary ;

X = nick

no

Σύνταξη της Prolog

- Τα στοιχεία της γλώσσας είναι:
  - οι όροι,
  - τα γεγονότα,
  - οι κανόνες,
  - οι ερωτήσεις.
- Ένα Prolog πρόγραμμα είναι ένα σύνολο από προτάσεις.

A. *Γεγονός* (*fact*)

A :- B_1, B_2, ..., $B_κ$. (κ > 0) *Κανόνας* (*rule*)

?- B_1, B_2, ..., $B_κ$. (κ > 0) *Ερώτηση* (*question*)

Ατομικοί Τύποι

- Τα **A** και $\mathbf{B_i}$ ονομάζονται ατομικοί τύποι (όπως στην Κατηγορηματική Λογική) και είναι παραστάσεις της μορφής:

$$\mathbf{P(t_1, t_2, \ldots, t_n)}$$

- όπου το:
  - **P** ονομάζεται *κατηγόρημα* (*predicate*).
  - $\mathbf{t_i}$: ονομάζονται *ορίσματα* (*arguments*).
  - Ο αριθμός των ορισμάτων (*n*) ονομάζεται *τάξη* (*arity*) του κατηγορήματος.

Όροι

- Τα ορίσματα ενός κατηγορήματος είναι *όροι* (*terms*) και μπορεί να είναι:
  - *Σταθερά.*
  - *Μεταβλητή.*
  - *Σύνθετος όρος* ή *Συναρτησιακός όρος.*

Σταθερές

- Άτομα ή αριθμοί.
- Οι αριθμοί έχουν τη συνήθη μορφή.
  - Π.χ. 2, 3, 6.7, -3, -10 κλπ.
- Τα *άτομα* (*atoms*) είναι συμβολοσειρές.
  - μπορούν να περιλαμβάνουν αριθμούς.
  - πρέπει απαραίτητα να ξεκινούν από πεζό γράμμα ή να περιλαμβάνονται σε μονά εισαγωγικά.
  - Παραδείγματα αποδεκτών ατόμων είναι: **anna**, **x25**, **x_25**, **'Anna'**, κλπ.
  - Μη-αποδεκτά άτομα: **Anna**, **X25**, **25X**, **_Q**, **‘Car**

Μεταβλητές

- Οι μεταβλητές στην Prolog είναι συμβολοσειρές που μπορεί να περιέχουν ψηφία ή τον χαρακτήρα "_".
- Χρησιμοποιούνται στη θέση άγνωστων ορισμάτων σε μια πρόταση και πρέπει πάντα να ξεκινούν με κεφαλαίο γράμμα ή με τον χαρακτήρα "_".
- Παραδείγματα αποδεκτών μεταβλητών είναι: **X**, **ListOfStrings**, **List_of_strings**, **Obj2**, **_X1**.
- Μη-αποδεκτές μεταβλητές: **x**, **oBJ**, **2Obj**

Prolog και Διαδικαστικές Γλώσσες Προγραμματισμού (1/2)

- Υπάρχουν σημαντικές διαφορές ανάμεσα στις μεταβλητές των κλασσικών γλωσσών και της Prolog.
- Στις κλασσικές γλώσσες μια μεταβλητή μπορεί μετά την πρώτη ανάθεση να μεταβάλλει την τιμή που έχει πάρει (destructive assignment).
  - Αντίθετα στην Prolog, σε μια μεταβλητή η οποία έχει πάρει τιμή δεν μπορεί να δοθεί νέα (non destructive assignment).
  - Π.χ. η συνηθισμένη σε άλλες γλώσσες έκφραση N = N + 1, δεν έχει κανένα νόημα στην Prolog.

Prolog και Διαδικαστικές Γλώσσες Προγραμματισμού (2/2)

- Οι μεταβλητές στην Prolog, δεν έχουν τύπο (typeless) και άρα δεν απαιτούν δηλώσεις και μπορούν σαν τιμή οποιοδήποτε όρο.
- Υπάρχουν μόνο τοπικές μεταβλητές, των οποίων η εμβέλεια είναι μέσα στον κανόνα τον οποίο εμφανίζονται.

Σύνθετοι όροι

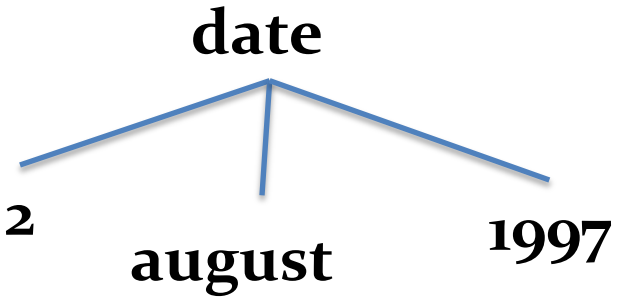
- Ο *Σύνθετος όρος* είναι δομή της μορφής $\mathbf{f(t_1, t_2, \ldots, t_\kappa)}$:
  - **f**: *συναρτησιακό σύμβολο* (*functor*).
  - $\mathbf{t_i}$: *ορίσματα* του συναρτησιακού συμβόλου και είναι όροι.
  - Ο αριθμός των ορισμάτων (κ) ονομάζεται *τάξη* (*arity*) του συναρτησιακού συμβόλου.

Παραδείγματα Σύνθετων Όρων (1/2)

- Αναπαράσταση ημερομηνίας:

date(2,august,1997)

- Πρόκειται για έναν σύνθετο όρο τάξης 3 αφού έχει τρία ορίσματα (ημέρα, μήνας, έτος).
- Στο παράδειγμα αυτό, όλα τα ορίσματα είναι απλοί όροι.

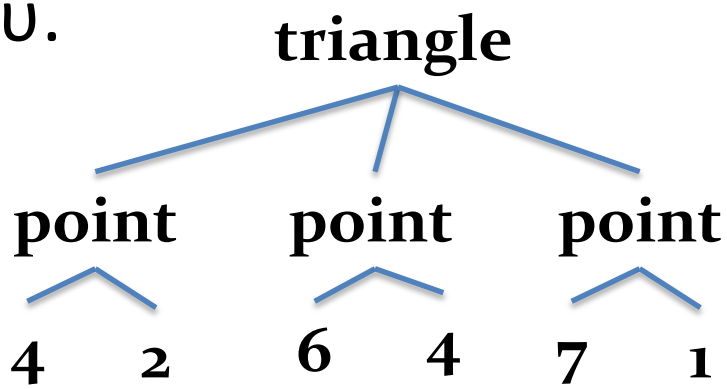


Παραδείγματα Σύνθετων Όρων (2/2)

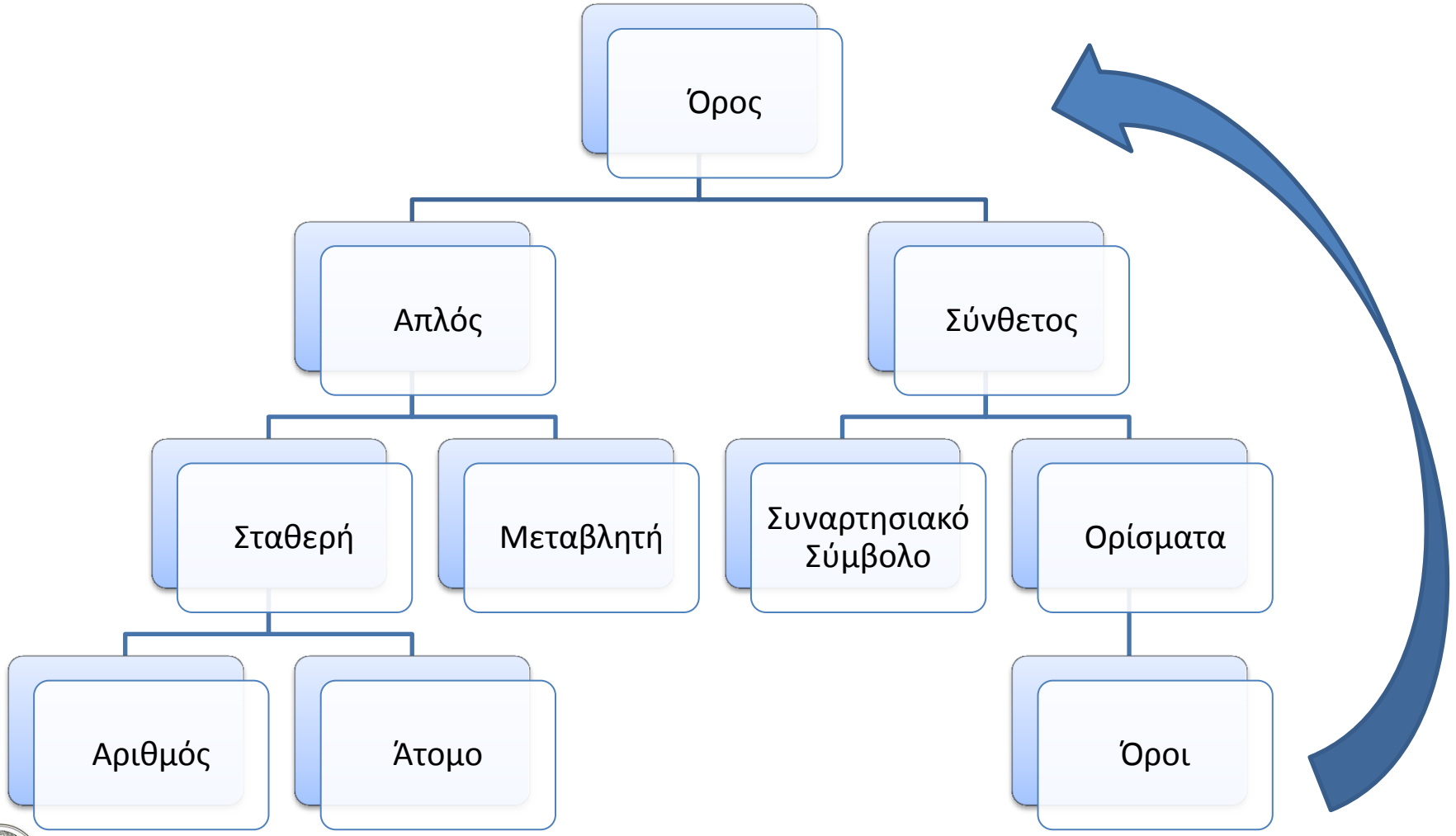
- Η αναπαράσταση ενός τριγώνου μπορεί να γίνει χρησιμοποιώντας έναν σύνθετο όρο τάξης 3.

triangle(point(4,2),point(6,4),point(7,1))

- Το κάθε όρισμά του είναι με τη σειρά του ένας άλλος σύνθετος όρος που αναπαριστά μια κορυφή του τριγώνου.
  - Π.χ. **point(4,2)**



Περιπτώσεις Όρων



Σχόλια

- Τα σχόλια στην Prolog εισάγονται είτε με τον χαρακτήρα "%" είτε περικλείονται στους χαρακτήρες "/*" και "*/".

```
% This is a line comment
father(jim,mary).  % Jim is father of Mary
/* father connects a father and his son
   or his daughter
*/
```

Γεγονότα (1/2)

- Η πιο απλή μορφή προτάσεων Horn είναι τα γεγονότα (facts).
- Ένα γεγονός είναι η "απευθείας" αναπαράσταση μιας σχέσης που συνδέει ένα η περισσότερα αντικείμενα.
  - Π.χ. για να δηλώσουμε ότι η Ελλάδα, η Ιταλία και η Αγγλία είναι χώρες τις Ευρώπης εισάγουμε τα γεγονότα

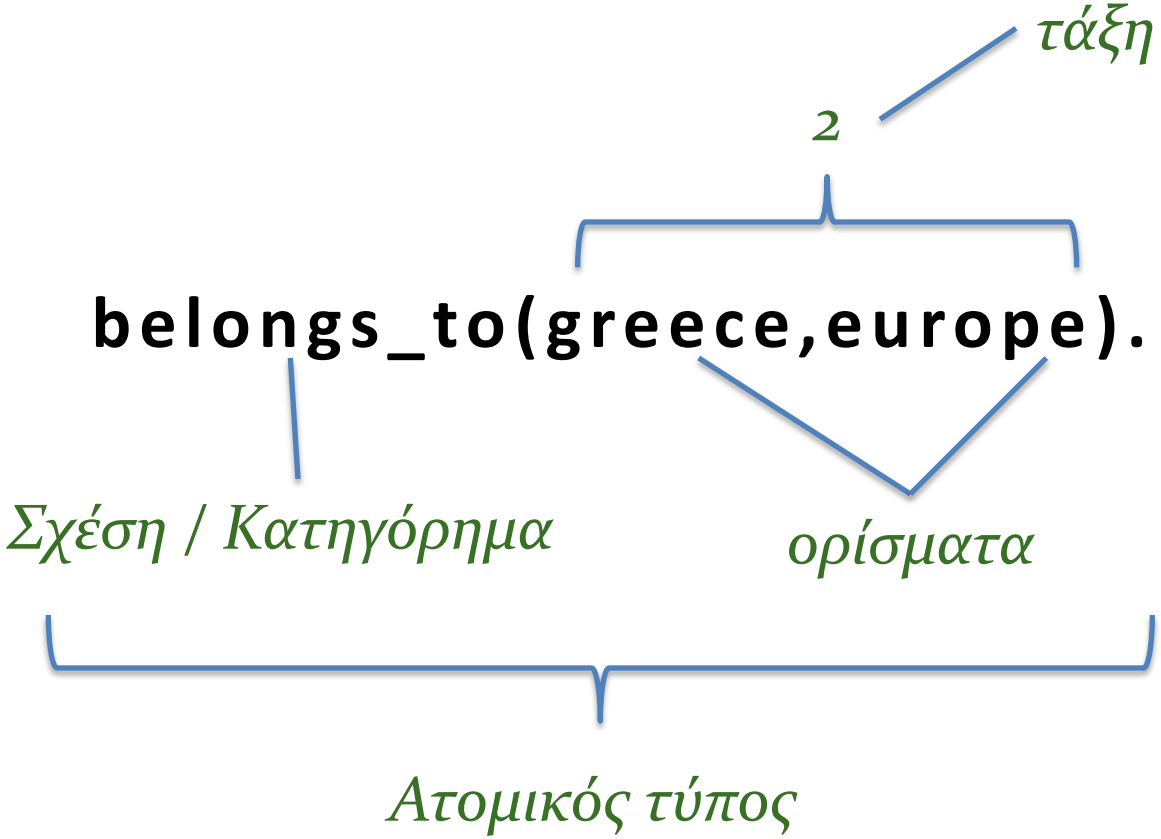
belongs_to(greece,europe).

belongs_to(italy,europe).

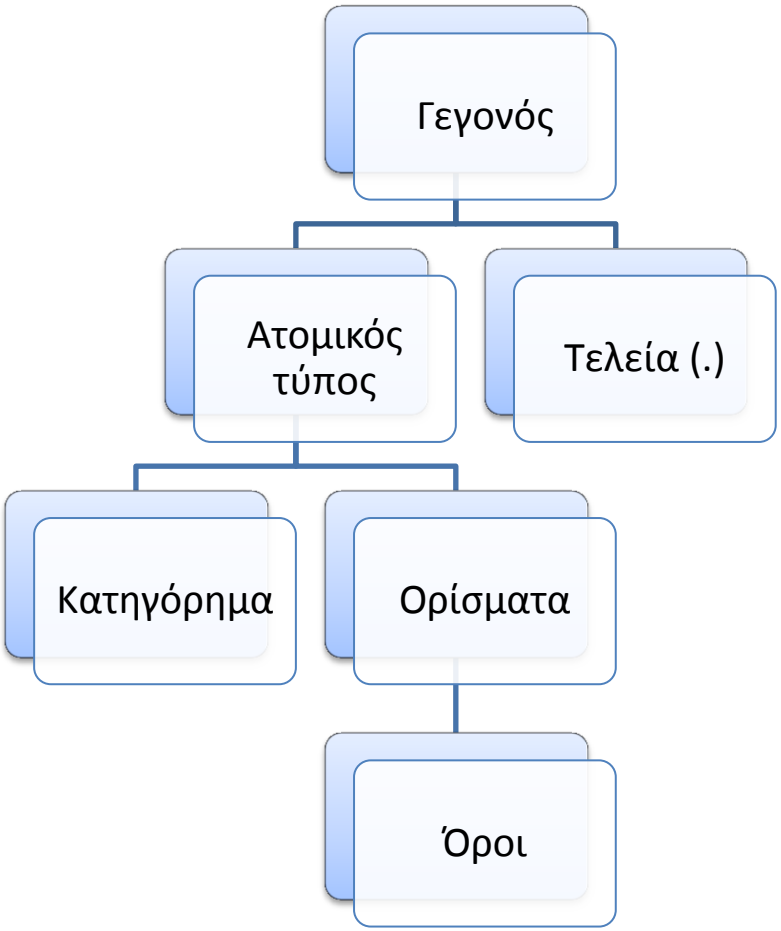
belongs_to(england,europe).

- Η σχέση (relationship) **belongs_to** συνδέει τα δύο άτομα (atoms) π.χ. **greece**, **europe**.

Γεγονότα (2/2)


τάξη
2
belongs_to(greece,europe).
Σχέση / Κατηγόρημα
ορίσματα
Ατομικός τύπος

Σύνταξη Γεγονότων



Κάθε πρόταση πρέπει να τελειώνει με το σημείο στίξης της τελείας "."

Κανόνες

- Οι κανόνες (rules) είναι προτάσεις που ορίζουν νέες σχέσεις με τη βοήθεια άλλων σχέσεων που έχουν ήδη οριστεί.
- Οι κανόνες έχουν την γενική μορφή:

A :- B_1, B_2, B_3, ..., B_n. (n >= 1).

- Το **A** ονομάζεται κεφαλή (head) του κανόνα
  - Η κεφαλή μπορεί να είναι οποιοσδήποτε ατομικός τύπος.
- Τα **B_1, B_2, B_3, ..., B_n**, ονομάζονται σώμα (body) του κανόνα, και είναι και αυτοί ατομικοί τύποι.
- Η κεφαλή διαχωρίζεται από το σώμα με τους χαρακτήρες ":-", οι οποίοι μπορούν να διαβαστούν σαν το λογικό συνδετικό της συνεπαγωγής "EAN".
- Η σημασία του σημείου στίξης του κόμματος "," μεταξύ των ατομικών τύπων στο σώμα ενός κανόνα, ή στην ερώτηση είναι αυτή της λογικής σύζευξης (AND).

Παράδειγμα Κανόνα

- Δύο άτομα (X και Y) είναι αδέλφια αν ο γονέας του ενός (Z) είναι και γονέας του άλλου.
- Αυτό μπορεί να εκφραστεί με τη βοήθεια ενός κανόνα:

```
% siblings/2

siblings(X,Y):-
    parent(Z,X),
    parent(Z,Y).
```

Συμβολισμός: Κατηγόρημα ***siblings*** *τάξης 2*

- Η κοινή μεταβλητή (shared variable) **Z** εκφράζει την συνθήκη ότι οι **X** και **Y** έχουν τον ίδιο γονέα.

Ερμηνεία Προτάσεων

- Τα προγράμματα στην Prolog μπορούν να γίνουν κατανοητά με δύο τρόπους: *δηλωτικά* (*declaratively*) και *διαδικαστικά* (*proceduraly*).
- Π.χ. πρόταση **P :- Q, R.** μπορεί να ερμηνευτεί:
- *Δηλωτικά*: Το **P** είναι αληθές εάν τα **Q** είναι αληθές και εάν το **R** είναι αληθές.
- *Διαδικαστικά*: Για να αποδειχθεί ότι το **P** είναι αληθές, πρέπει πρώτα να αποδειχθεί ότι το **Q** είναι αληθές και στη συνέχεια ότι και το **R** είναι αληθές.

Παρατηρήσεις για τα Κατηγορήματα

- Η τάξη ενός κατηγορήματος πρέπει να είναι η ίδια, οπουδήποτε και αν εμφανίζεται το κατηγόρημα αυτό μέσα στο πρόγραμμα.
- Το ίδιο πρέπει να συμβαίνει και με τη σειρά των ορισμάτων του.
- Αν τα ορίσματα ενός κατηγορήματος δεν είναι συγκεκριμένα, μπορούμε στη θέση τους να τοποθετήσουμε μεταβλητές.

Ερωτήσεις (1/2)

- Οι ερωτήσεις είναι η μέθοδος που χρησιμοποιείται στην Prolog για την εξαγωγή γνώσης από ένα πρόγραμμα.
- Η σύνταξή τους είναι: ?- **ατομικός τύπος**.
  - Ο ατομικός τύπος ονομάζεται κλήση (call).
- Ανάλογα με τον αριθμό των κλήσεων, οι ερωτήσεις διακρίνονται σε δύο κατηγορίες: τις απλές και τις σύνθετες.

Ερωτήσεις (2/2)

- Οι ερωτήσεις μπορούν να θεωρηθούν και σαν λογικές προτάσεις προς απόδειξη.
  - Η Prolog βασισμένη στα ορισμένα κατηγορήματα, προσπαθεί να αποδείξει τη αλήθεια των προτάσεων αυτών.
  - Αν η προς απόδειξη πρόταση (ερώτηση) ικανοποιείται, η απάντηση στην ερώτηση είναι θετική, και περιλαμβάνει τις τιμές για τις μεταβλητές που πιθανόν να περιλαμβάνονται σε αυτή.
  - Στην αντίθετη περίπτωση, το σύστημα απαντά αρνητικά (no).
- Ουσιαστικά η Prolog υλοποιεί την αποδεικτική διαδικασία της «αρχής της ανάλυσης» συνδυασμένη με την «απαγωγή σε άτοπο».

Απλές ερωτήσεις (1/2)

- Περιέχουν μία μόνο κλήση.

big(bear). **big (elephant).** **small(cat).**

brown(bear). **gray(elephant).** **black(cat).**

?-brown(elephant).

no

  - Η κλήση δεν ικανοποιείται αφού το αντίστοιχο γεγονός δεν υπάρχει.

?-big(elephant).

yes

  - Η κλήση ικανοποιείται αφού το σύστημα, βρίσκει το αντίστοιχο γεγονός.

Απλές ερωτήσεις (2/2)

big(bear).
brown(bear).

big (elephant).
gray(elephant).

small(cat).
black(cat).

?-big(X).

X=bear;

X=elephant;

no

- Η μεταβλητή **X** αποκτά την τιμή **bear** σύμφωνα με την πρώτη πρόταση.
- Αν ο χρήστης ζητήσει και άλλη λύση, αυτή παίρνει την τιμή **elephant** από τη δεύτερη πρόταση.

Σύνθετες ερωτήσεις (1/4)

- Περιέχουν παραπάνω από μία κλήσεις.
- Οι κλήσεις συνήθως χωρίζονται μεταξύ τους με κόμμα (",") το οποίο αντιστοιχεί στον λογικό τελεστή AND.
- Η απάντηση σε μια σύνθετη ερώτηση είναι καταφατική μόνο αν αληθεύουν **όλες** οι επιμέρους κλήσεις που την αποτελούν.

Σύνθετες ερωτήσεις (2/4)

big(bear). big (elephant). small(cat).

brown(bear). gray(elephant). black(cat).

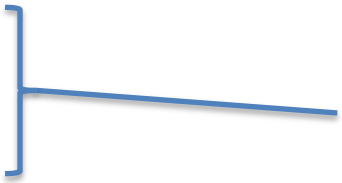
?- small(X), gray(X).

no

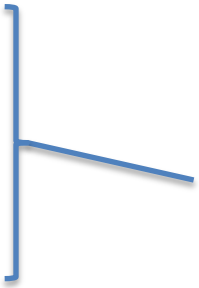
- Σύμφωνα με την τρίτη πρόταση, η μεταβλητή **X** της πρώτης κλήσης θα πάρει την τιμή **cat**.
- Τότε, και η μεταβλητή της δεύτερης κλήσης θα αποκτήσει την ίδια τιμή μιας και έχει το ίδιο όνομα με την μεταβλητή της προηγούμενης κλήσης.
- Έτσι, η ερώτηση γίνεται: **gray(cat).**
- Επειδή, όμως, το αντίστοιχο γεγονός λείπει από το πρόγραμμα, η απάντηση που θα επιστρέψει το σύστημα θα είναι αρνητική.

Σύνθετες ερωτήσεις (3/4)

```
big(bear).          big (elephant).     small(cat).
brown(bear).        gray(elephant).     black(cat).
?-big(X), gray(X).
X=elephant;
no
?-big(X), black(Y).
X=bear, Y=cat;
X=elephant, Y=cat;
no
```



Οι δύο κλήσεις συναληθεύουν για την ίδια τιμή του X.



- *Οι δύο μεταβλητές παίρνουν τιμή **ανεξάρτητα** η μία από την άλλη.*
- *Το σύνολο των απαντήσεων είναι το **καρτεσιανό γινόμενο** των συνόλων τιμών που μπορεί να πάρει η κάθε μεταβλητή ανεξάρτητα.*

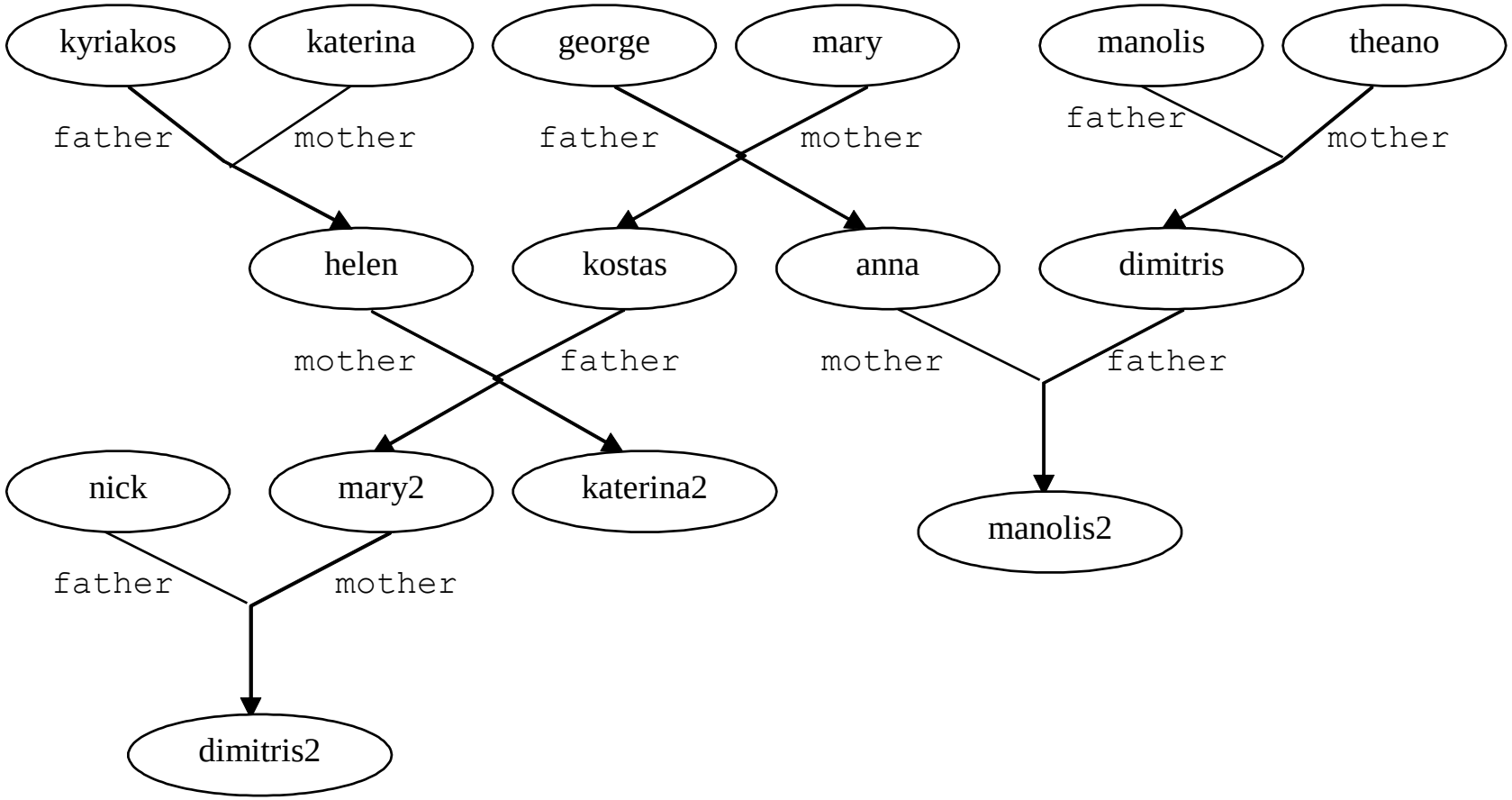
Σύνθετες ερωτήσεις (4/4)

```
big(bear).   big (elephant).
black(cat).  black(puma).
?-big(X), black(Y).
X=bear, Y=cat;
X=bear, Y=puma;
X=elephant, Y=cat;
X=elephant, Y=puma;
no
```

Η δεύτερη μεταβλητή «αλλάζει» τιμή πρώτη, γιατί η οπισθοδρόμηση είναι «χρονολογική», δηλαδή ο υπολογισμός επιστρέφει στο πιο κοντινό χρονικά σημείο που μπορεί να δώσει εναλλακτική λύση.

- *Όταν η οπισθοδρόμηση φθάσει την πρώτη μεταβλητή, η εκτέλεση ξανακυλάει με την κανονική της ροή (από αριστερά προς τα δεξιά).*
- *Οι παραπάνω τιμές της Υ έχουν ξεχαστεί και δίνονται με τη σειρά ξανά όπως την πρώτη φορά.*

Άσκηση Γενεαλογικού Δένδρου



Γενεαλογικό Δένδρο – Ερωτήσεις (1/2)

- *Είναι ο Γιώργος πατέρας της Άννας;*

?- father(george,anna).

yes

- *Είναι η Μαρία μητέρα του Δημήτρη;*

?- mother(mary,dimitris).

no

- *Ποιος είναι ο πατέρας της Ελένης;*

?- father(X,helen).

X=kyriakos

Γενεαλογικό Δένδρο – Ερωτήσεις (2/2)

- *Ποιο(-α) είναι τα παιδιά της Μαρίας;*

?- mother(mary,X).

X=kostas;

X=anna

- *Ποιοι είναι οι γονείς της Ελένης;*

?- father(X,helen) ; mother(X,helen).

X=kostas;

X=anna

Σύμβολο της λογικής διάζευξης (OR)

Γενεαλογικό Δένδρο
Ερωτήσεις + Κανόνες

- *Ποιοι είναι οι γονείς της Ελένης;*
- Προσθέτουμε στο πρόγραμμα τους ακόλουθους κανόνες που ορίζουν το κατηγόρημα **parent/2**.

```
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

*Η διάζευξη είναι «κρυμμένη» μέσα στους πολλαπλούς εναλλακτικούς κανόνες για το **parent/2.***

```
?- parent(X,helen).
X=kostas;
X=anna
```

Γενεαλογικό Δένδρο
Σχέση grandfather (1/2)

– *Ποιο(-α) είναι τα εγγόνια του Κυριάκου;*

Το εγγόνι του Κυριάκου

Το παιδί του Κυριάκου

- Με σύνθετη ερώτηση:

?- father(kyriakos, X), (father(X, Y) ; mother(X, Y)).

X=helen, Y=mary2;

X=helen, Y=katerina2

ή

?- father(kyriakos, X), parent(X, Y).

Γενεαλογικό Δένδρο

Σχέση grandfather (2/2)

- Με κανόνα:

grandfather(Z, Y) :- father(Z, X), parent(X, Y).

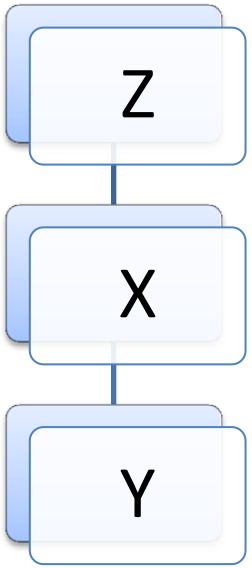
grandparent(Z, Y) :- parent(Z, X), parent(X, Y).

- Ερώτηση:

?- grandfather(kyriakos, Y).

Y=mary2;

Y=katerina2



Γενεαλογικό Δένδρο

Σχέση sibling (1/2)

- *Ποιο(-α) είναι τα αδέλφια του Κώστα;*
- Με σύνθετη ερώτηση:

?- parent(X, kostas), parent(X, Y).

X = george, Y = kostas ;

X = george, Y = anna ;

X = maria, Y = anna ;

X = maria, Y = kostas

Το «άλλο» παιδί του γονιού του Κώστα.

Ο γονέας του Κώστα.

Ο Κώστας είναι επίσης παιδί του γονιού του Κώστα!

- Αφού υπάρχουν δυο γονείς, όλες οι λύσεις θα δοθούν 2 φορές.

Γενεαλογικό Δένδρο

Σχέση sibling (2/2)

?- parent(X, kostas), parent(X, Y), Y\=kostas.

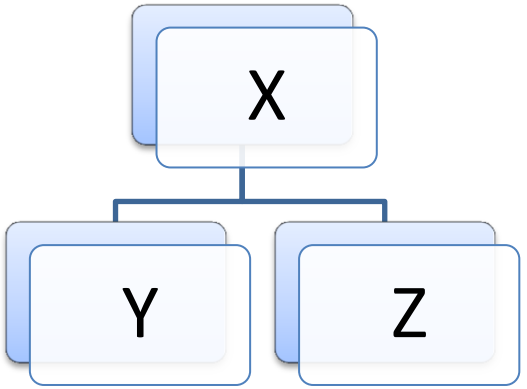
X = george, Y = anna ;

X = maria, Y = anna

Ζητάμε η τιμή που θα πάρει η μεταβλητή Y να μην ισούται με kostas.

- Δεν μπορούμε να αποφύγουμε να δοθούν 2 φορές οι λύσεις.
- Κανόνας:

```
sibling(Z,Y) :-
    parent(X,Z),
    parent(X,Y),
    Z \= Y.
```



?- sibling(kostas,Y).

Y = anna ;

Y = anna

Γενεαλογικό Δένδρο

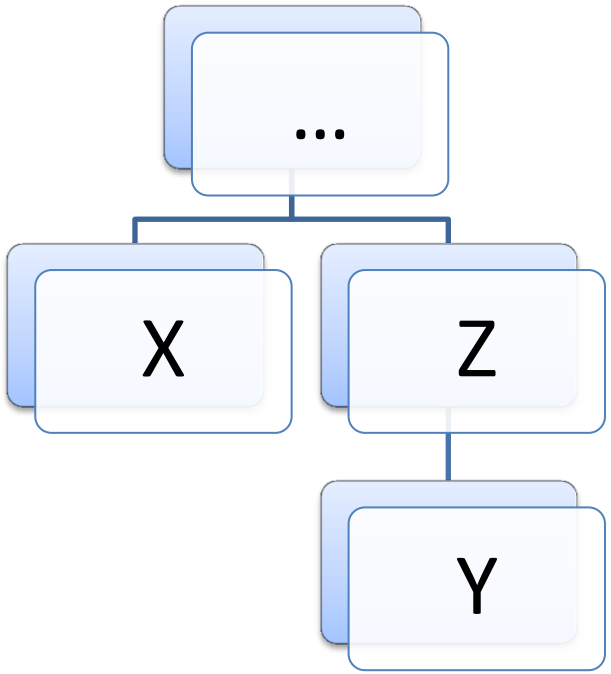
Σχέση aunt/uncle

- Η σχέση “θείος/θεία” μπορεί να οριστεί με τη βοήθεια της sibling.

aunt_uncle(X,Y) :-

sibling(X,Z),

parent(Z,Y).



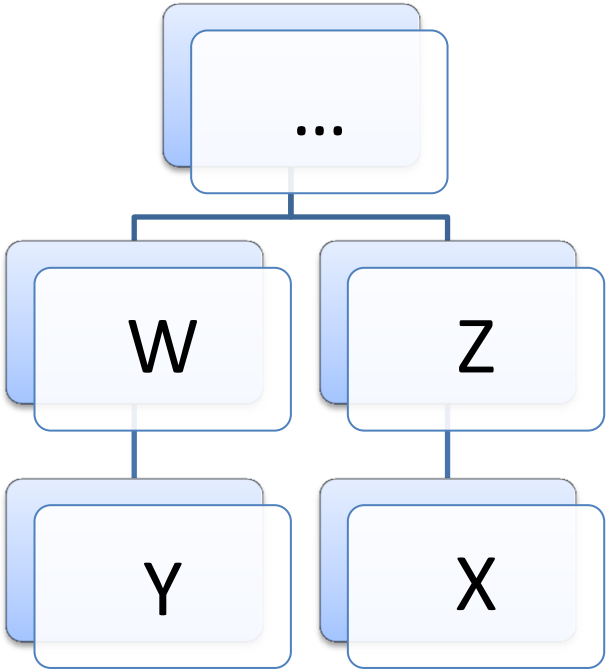
Γενεαλογικό Δένδρο - Σχέση cousin

- Η σχέση «ξάδελφος/η» μπορεί να οριστεί είτε με τη σχέση **aunt_uncle/parent**, ή με τις **sibling/parent.**

```
cousin(X,Y) :-
    aunt_uncle(W,X),
    parent(W,Y).
```

ή

```
cousin(X,Y) :-
    parent(Z,X),
    sibling(Z,W),
    parent(W,Y).
```



Άσκηση

Κατηγορήματα με τάξη μηδέν

- Είναι κατηγορήματα χωρίς ορίσματα.
- Έστω ότι έχουμε τα στοιχεία (και ενώσεις) : **MgO**, **H_2**, **C** και **O_2**
- Θέλουμε να δούμε αν μπορούμε να πάρουμε **H_2CO_3** από τις εξισώσεις :

$MgO + H_2 \rightarrow Mg + H_2O$

$C + O_2 \rightarrow CO_2$

$CO_2 + H_2O \rightarrow H_2CO_3$

Πρόγραμμα Α

μαγνήσιο : - οξείδιο_μαγνησίου, υδρογόνο.
νερό : - οξείδιο_μαγνησίου, υδρογόνο.
διοξείδιο_άνθρακα : - άνθρακας, οξυγόνο.
ανθρακικό_οξύ : - διοξείδιο_άνθρακα, νερό.
οξείδιο_μαγνησίου.
υδρογόνο.
άνθρακας.
οξυγόνο.
Ερώτηση: ***?- ανθρακικό_οξύ.***
Απάντηση: ***Yes***

Πρόγραμμα Β

mg : - mgo , h2.

h2o : - mgo , h2.

co2 : - c , o2.

h2co3 : - co2 , h2o.

mgo.

h2.

c.

o2.

Ερώτηση: ***?- h2co3.***

Απάντηση: ***Yes***

Ασκήσεις (1/2)

- Ένα πρόγραμμα Prolog περιλαμβάνει γεγονότα της μορφής:

αγαπά(α1, α2) *% ο α1 αγαπά τον α2.*

- Υποβάλλετε στο πρόγραμμα τις ακόλουθες ερωτήσεις:
  - *" Υπάρχει κάποιος που αγαπά τον εαυτό του;"*
  - *"Υπάρχει κάποιος που αγαπά αυτόν που τον αγαπά;"*
  - *"Υπάρχουν 2 πρόσωπα που αγαπούν το ίδιο πρόσωπο;"*
- Απαντήσεις.

?- αγαπά(X, X). *% ανακλαστική ιδιότητα*

?- αγαπά(X, Y), αγαπά(Y, X). *% συμμετρική ιδιότητα*

?- αγαπά(X, Z), αγαπά(Y, Z).

Ασκήσεις (2/2)

- Στο προηγούμενο πρόγραμμα προσθέστε τους κανόνες:
  - *“Όλοι αγαπούν τον εαυτό τους”.*
  - *“Αν ο Α αγαπά τον Β και ο Β τον Γ, τότε ο Α αγαπά τον Γ”.*
  - *“Αν δύο άνθρωποι αγαπούν το ίδιο πρόσωπο, τότε αγαπιούνται και μεταξύ τους”.*
- Απαντήσεις

αγαπά(Χ, Χ). *% Γενικευμένο γεγονός – έχει μεταβλητές*

αγαπά(Α, Γ) : - αγαπά(Α, Β), αγαπά(Β, Γ).

αγαπά(Χ, Υ) : - αγαπά(Υ, Χ).

αγαπά(Χ, Υ): - αγαπά(Χ, Ζ), αγαπά(Υ, Ζ).

  - Πρέπει να ισχύει Χ ≠ Υ

Εκτέλεση Προγραμμάτων

- Η εκτέλεση ξεκινά με μια ερώτηση που υποβάλλει ο χρήστης.
  - Φτάνουμε σε λύση όταν έχουν εξαντληθεί όλες οι κλήσεις της ερώτησης.
  - Η απάντηση είναι το αποτέλεσμα του προγράμματος.
- Η ερώτηση μπορεί να είναι:
  - απλή, περιέχει μία μόνο κλήση, ή

?- friend(X,nick).

  - σύνθετη, αποτελείται από σύζευξη πολλαπλών κλήσεων.

?- friend(X,nick), friend(X,john).

Βήματα Εκτέλεσης (1/2)

- Έως ότου δεν υπάρχουν άλλες κλήσεις στην ερώτηση, επιλέγονται με τη σειρά, από αριστερά προς τα δεξιά, οι κλήσεις της ερώτησης.
- Για κάθε κλήση της ερώτησης, ο μηχανισμός ελέγχου αναλαμβάνει να βρει μια πρόταση του προγράμματος της οποίας η κεφαλή έχει το ίδιο κατηγόρημα και τάξη με την επιλεγμένης κλήση.
- Η ερώτηση θεωρείται ότι απαντήθηκε, όταν απαντηθούν επιτυχώς όλες οι κλήσεις της.
- Εφόσον οι κλήσεις της ερώτησης περιέχουν κοινές μεταβλητές, θα πρέπει αυτές να πάρουν την ίδια τιμή.

Βήματα Εκτέλεσης (2/2)

- Αν δεν υπάρχει άλλη κλήση στην ερώτηση του χρήστη τότε η εκτέλεση του προγράμματος τερματίζεται με επιτυχία (yes) και επιστρέφονται στον χρήστη οι τιμές των μεταβλητών που περιείχε η αρχική ερώτηση.
  - Αυτό προφανώς δεν γίνεται στην αρχική ερώτηση του χρήστη (η οποία δεν είναι κενή), αλλά είναι μια κατάσταση που προκύπτει από την επαναληπτική εφαρμογή των βημάτων εκτέλεσης.

Ενοποίηση με Γεγονός

- Όταν η κλήση που εξετάζεται ενοποιείται με ένα από τα γεγονότα του προγράμματος τότε αυτή ικανοποιείται και απομακρύνεται από την ερώτηση.
- Π.χ. **?- father(X,Y).**

father(george,nick).

father(X,Y) $\rightarrow$ *ή* $\neg$**father(X,Y)**
father(george,nick)

- Για την αντικατάσταση {**X/george, Y/nick**} η κλήση με την πρόταση (γεγονός) γίνονται ταυτόσημα και η κλήση ικανοποιείται και απομακρύνεται.
- Παραμένει κενή η ερώτηση (χωρίς κλήσεις), άρα η διαδικασία απόδειξης έχει τερματίσει επιτυχημένα.

Ενοποίηση με Κανόνα

- Αν η τρέχουσα κλήση ενοποιείται με κάποιον κανόνα, τότε αυτή απομακρύνεται από την ερώτηση και τη θέση της παίρνει το σώμα του κανόνα αυτού.
  - Για την ικανοποίηση της αρχικής κλήσης είναι απαραίτητη η ικανοποίηση των κλήσεων του σώματος του κανόνα που την αντικατέστησε.
- Π.χ. **?- grandfather(nick,W).**

grandfather(X,Y) :- father(X,Z), father(Z,Y).

- Για {**X/nick, Y/W**} η κλήση με την κεφαλή της πρότασης (κανόνας) γίνονται ταυτόσημα και η κλήση ικανοποιείται αν ικανοποιηθούν οι προϋποθέσεις (το σώμα) του κανόνα.
- Γίνεται αντικατάσταση της κλήσης στην ερώτηση με το σώμα

?- father(nick,Z), father(Z, W).

Ισοδυναμία με την Αρχή της Ανάλυσης στη μορφή Kowalski

grandfather(nick,W) →

father(X,Z), father(Z,Y) → grandfather(X,Y)

father(nick,Z), father(Z, W) →

Ισοδυναμία με την Αρχή της Ανάλυσης στην προτασιακή μορφή.

¬grandfather(nick,W)

¬ father(X,Z)∨ ¬ father(Z,Y) ∨ grandfather(X,Y)

¬ father(X,Z)∨ ¬ father(Z,Y)

Εναλλακτικές Προτάσεις

- Αν υπάρχουν περισσότερες της μιας προτάσεις με τις οποίες μπορεί να ενοποιηθεί η κλήση, τότε ενοποιείται με την πρώτη.
- Ονομάζεται *σημείο οπισθοδρόμησης* και αντιπροσωπεύει πιθανές εναλλακτικές "απαντήσεις" στην κλήση.
- Σε περίπτωση αποτυχίας εύρεσης λύσης ή σε περίπτωση που ο χρήστης ζητά και άλλη λύση, ο *μηχανισμός οπισθοδρόμησης.*
  - επιστρέφει στο τελευταίο σημείο οπισθοδρόμησης ακυρώνοντας τα υπολογιστικά βήματα που έπονται και
  - αναζητά στις επόμενες προτάσεις κάποια που να μπορεί να ενοποιηθεί με την κλήση.

Κατάληξη Εκτέλεσης

- Αν με την εξάντληση της διαδικασίας αυτής **δεν** καταλήξουμε σε ερώτηση χωρίς κλήσεις, τότε η απάντηση στην αρχική ερώτηση είναι αρνητική ή αλλιώς αποτυχία (fail).
- Σε αντίθετη περίπτωση η απάντηση στην ερώτηση είναι θετική ή αλλιώς επιτυχής (yes).
  - Συνοδεύεται από τις αναθέσεις τιμών στις μεταβλητές της αρχικής ερώτησης που προέκυψαν από τις διαδοχικές ταυτοποιήσεις.

Παράδειγμα χωρίς μεταβλητές
Ερώτηση και Πρόγραμμα

?- a, b. *Αρχική ερώτηση*

Προτάσεις προγράμματος Prolog

a :- c, d.

b :- w. **b :- z.**

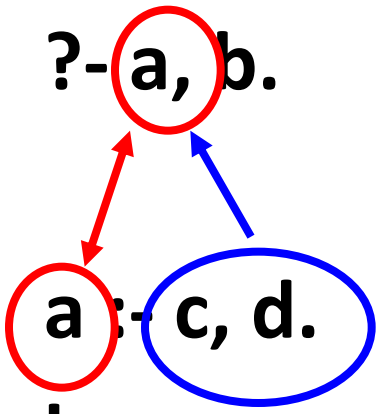
c :- e.

d :- g, h.

e. **f.** **g.** **h.** **z.**

Παράδειγμα χωρίς μεταβλητές Βήμα 1

(1/2)



Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιον κανόνα.

Το σώμα του κανόνα αντικαθιστά την κλήση.

b :- w.

b :- z.

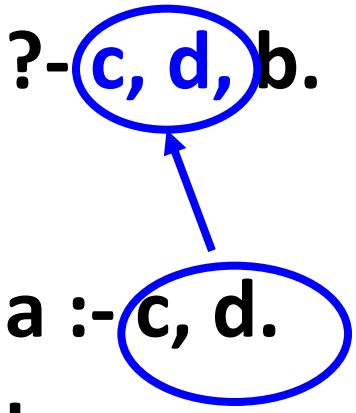
c :- e.

d :- g, h.

e. **f.** **g.** **h.** **z.**

Παράδειγμα χωρίς μεταβλητές Βήμα 1

(2/2)



Το σώμα του κανόνα αντικαθιστά την κλήση.

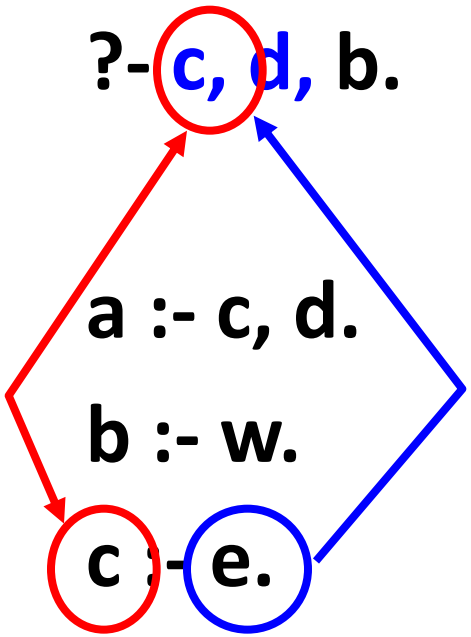
b :- w. **b :- z.**

c :- e.

d :- g, h.

e. **f.** **g.** **h.** **z.**

Παράδειγμα χωρίς μεταβλητές Βήμα 2



b :- z.

g.

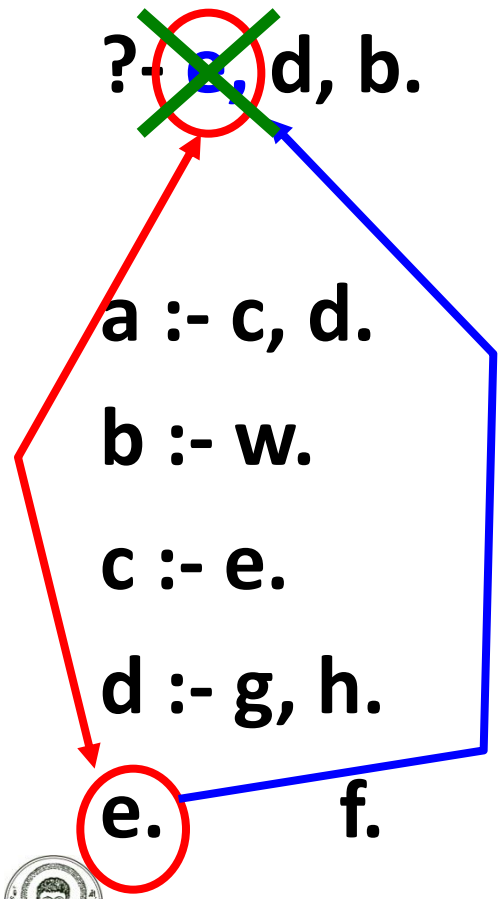
h.

z.

Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιον κανόνα.

Το σώμα του κανόνα αντικαθιστά την κλήση.

Παράδειγμα χωρίς μεταβλητές Βήμα 3

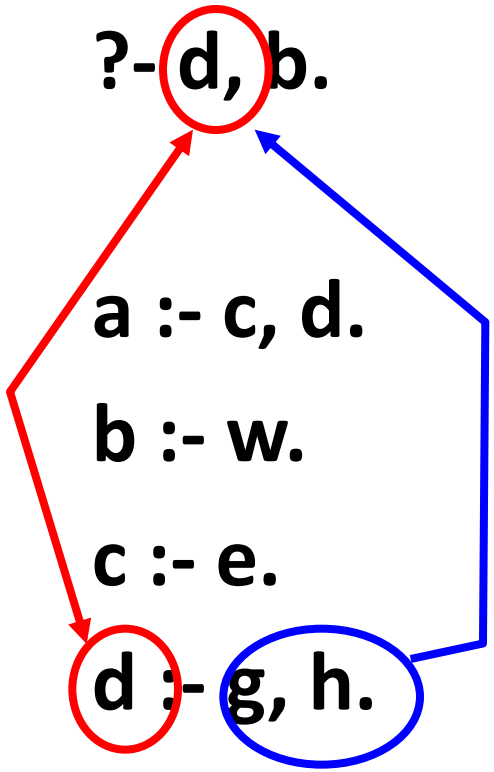


Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιο γεγονός.

Το γεγονός αυτό προκαλεί την αφαίρεση της κλήσης.

b :- z.

Παράδειγμα χωρίς μεταβλητές Βήμα 4



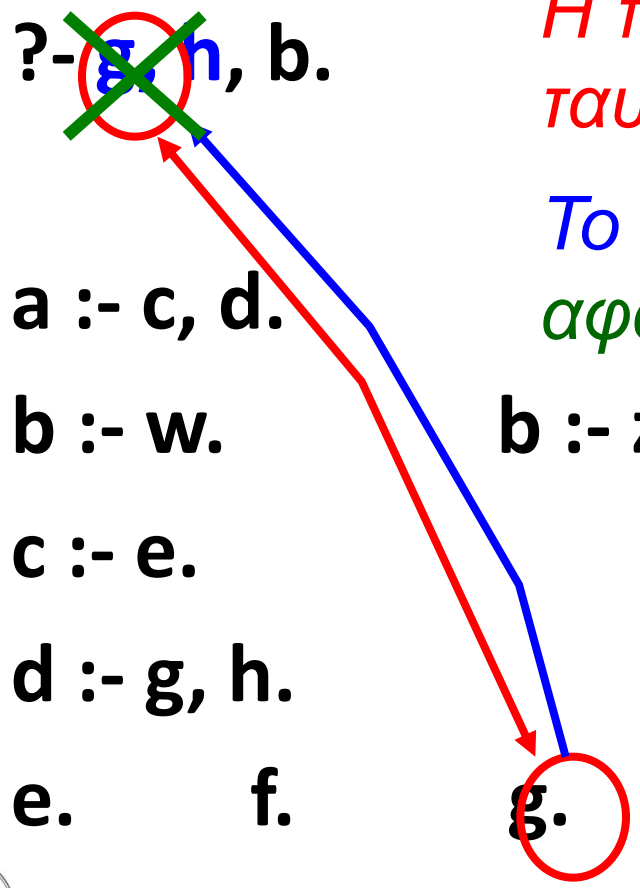
Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιον κανόνα.

Το σώμα του κανόνα αντικαθιστά την κλήση.

b :- z.

g. **h.** **z.**

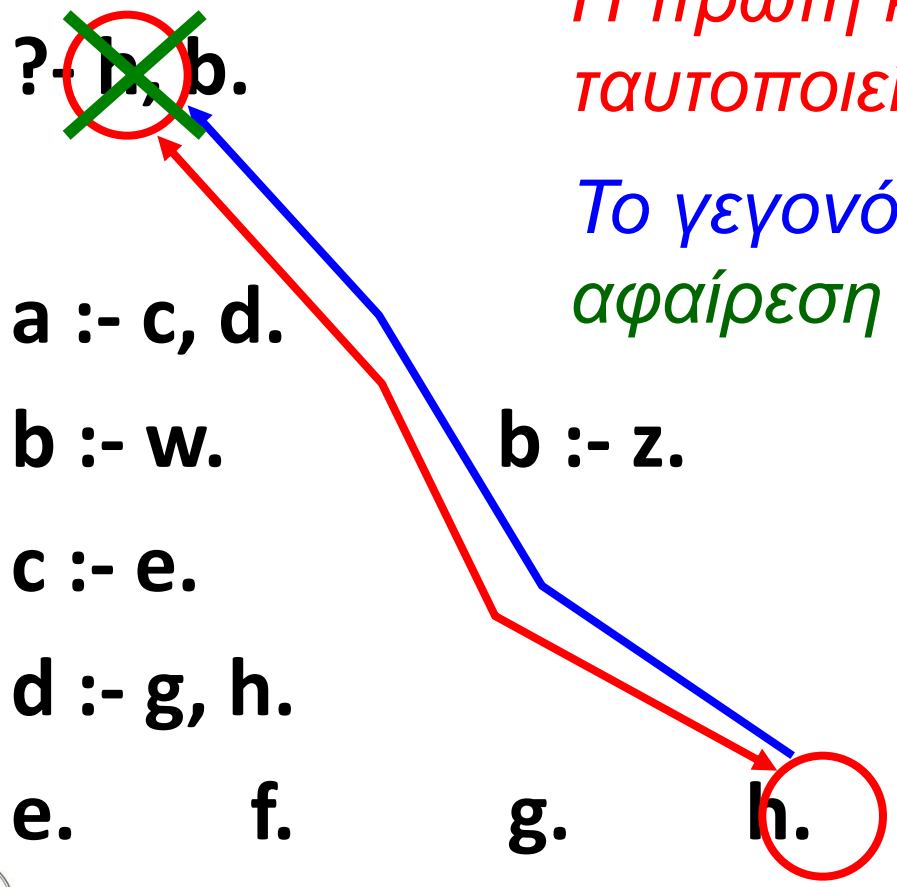
Παράδειγμα χωρίς μεταβλητές Βήμα 5



Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιο γεγονός.

Το γεγονός αυτό προκαλεί την αφαίρεση της κλήσης.

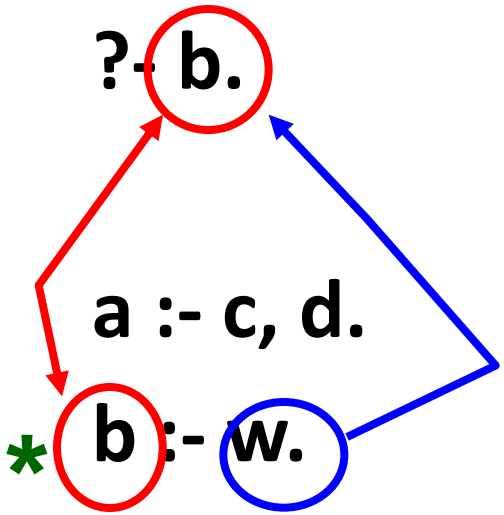
Παράδειγμα χωρίς μεταβλητές Βήμα 6



Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιο γεγονός.

Το γεγονός αυτό προκαλεί την αφαίρεση της κλήσης.

Παράδειγμα χωρίς μεταβλητές Βήμα 7



Η πρώτη κλήση (πιο αριστερή) ταυτοποιείται με κάποιον κανόνα.

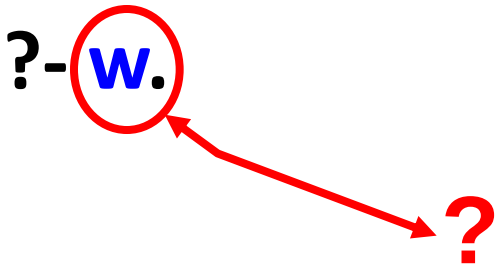
Η Prolog σημειώνει ότι υπάρχει και εναλλακτικός δρόμος, δηλαδή και άλλος υποψήφιος κανόνας για ταυτοποίηση.

b :- z.

Το σώμα του κανόνα αντικαθιστά την κλήση.

g. h. z.

Παράδειγμα χωρίς μεταβλητές Βήμα 8



a :- c, d.

b :- w. **b :- z.**

c :- e.

d :- g, h.

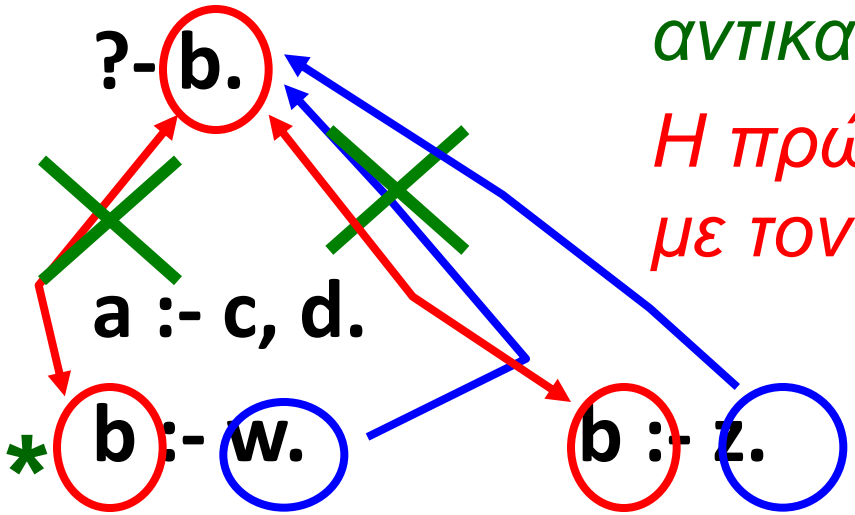
e. **f.** **g.** **h.** **z.**

Η κλήση δεν μπορεί να ταυτοποιηθεί με κάποιο γεγονός.

Προκαλείται αποτυχία στην ικανοποίηση του στόχου.

Η αποτυχία προκαλεί την οπισθοδρόμηση της διαδικασίας απόδειξης στο πιο κοντινό (χρονικά) σημείο όπου υπήρχε εναλλακτικός δρόμος στην απόδειξη.

Παράδειγμα χωρίς μεταβλητές Βήμα 9

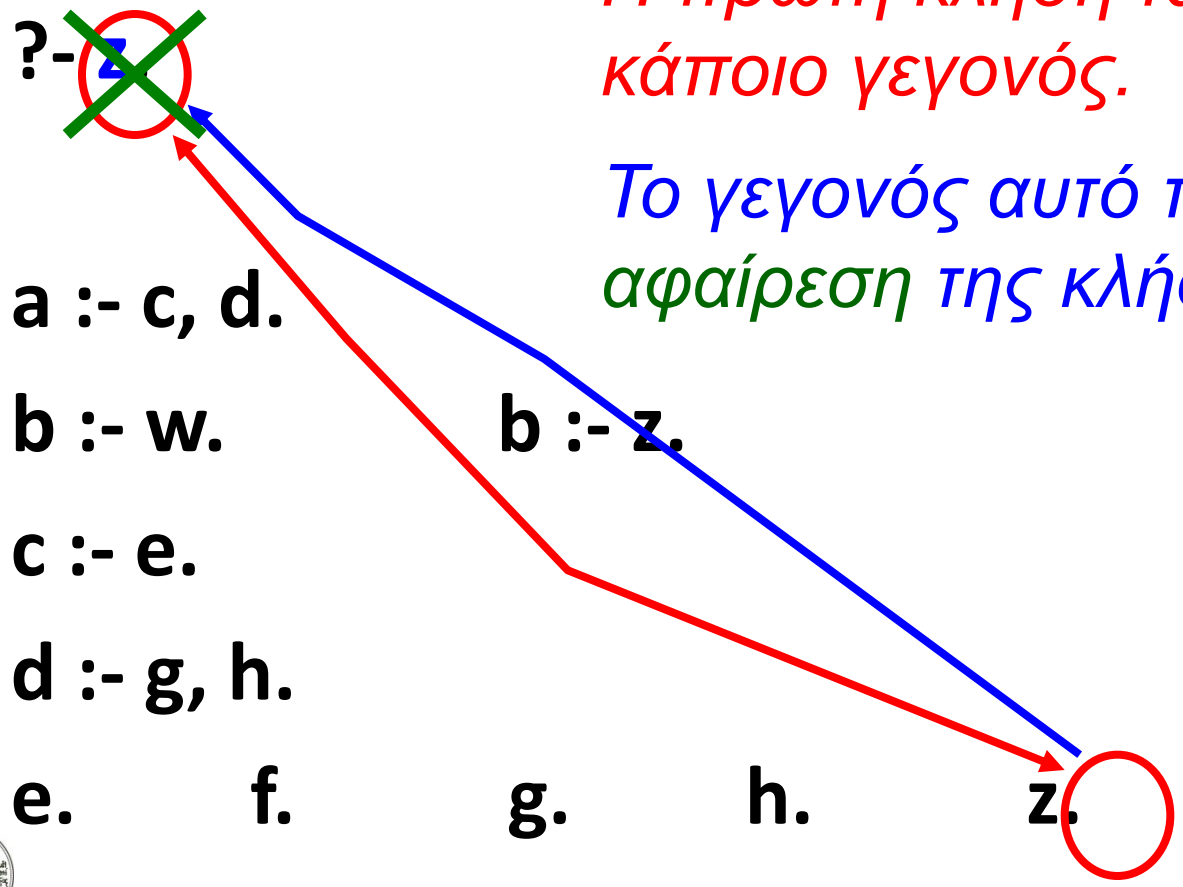


Ακύρωση της προηγούμενης αντικατάστασης.

Η πρώτη κλήση ξανα-ταυτοποιείται με τον εναλλακτικό κανόνα.

Το σώμα του εναλλακτικού κανόνα αντικαθιστά εκ νέου την κλήση.

Παράδειγμα χωρίς μεταβλητές Βήμα 10



Η πρώτη κλήση ταυτοποιείται με κάποιο γεγονός.

Το γεγονός αυτό προκαλεί την αφαίρεση της κλήσης.

Παράδειγμα χωρίς μεταβλητές Βήμα 11

?-

Δεν υπάρχουν άλλες κλήσεις (κενή κλήση).

Ο αρχικός στόχος θεωρείται ότι έχει ικανοποιηθεί.

a :- c, d.

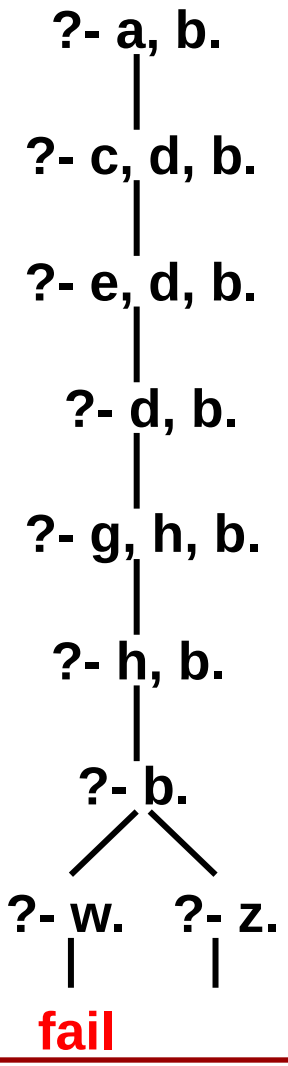
b :- w. . **b :- z.**

c :- e.

d :- g, h.

e. **f.** **g.** **h.** **z.**

Δένδρο Εκτέλεσης *ή* Δένδρο Υπολογισμού



Δένδρο Υπολογισμού (1/2)

- Κάθε κόμβος του δένδρου αντιστοιχεί στην τρέχουσα ερώτηση.
- Οι συνδέσεις (κλαδιά) μεταξύ των κόμβων αντιπροσωπεύουν ένα υπολογιστικό βήμα.
  - Χαρακτηρίζονται από τις αντικαταστάσεις των μεταβλητών που γίνονται σε αυτό.
- Κάθε διαδρομή από τον αρχικό κόμβο μέχρι κάποιο φύλλο του δένδρου (τελικός κόμβος) καταλήγει είτε σε επιτυχία είτε σε αποτυχία.

Δένδρο Υπολογισμού (2/2)

- Οι διαδρομές που καταλήγουν σε επιτυχία αντιπροσωπεύουν τις εναλλακτικές απαντήσεις στην ερώτηση του χρήστη.
- Ο τελικός κόμβος στις διαδρομές αυτές είναι η κενή πρόταση, που συμβολίζεται με το □.
- Σε περίπτωση που μια κλήση ενοποιείται με περισσότερες από μια προτάσεις του προγράμματος, τότε από τον κόμβο αυτόν ξεκινούν περισσότερα του ενός κλαδιά.
- Το σημείο αυτό είναι ένα σημείο οπισθοδρόμησης.

Ενοποίηση

- Επειδή οι κλήσεις μιας ερώτησης, τα γεγονότα και οι κεφαλές των κανόνων μπορεί να περιέχουν μεταβλητές η Prolog για να ελέγξει αν μία κλήση ικανοποιείται από μία πρόταση χρησιμοποιεί τον μηχανισμό *ενοποίησης* (*unification*).
  - παρόμοια με την κατηγορηματική λογική.
- Ο μηχανισμός προσπαθεί να καταστήσει ταυτόσημες μια κλήση και την κεφαλή μιας πρότασης εκτελώντας τις ελάχιστες αναθέσεις τιμών σε μεταβλητές.
  - *πιο γενικός ενοποιητής* (*most general unifier*)

Κανόνες Ενοποίησης (1/2)

- Μία μεταβλητή που δεν έχει πάρει τιμή μπορεί να ταυτοποιηθεί με σταθερή, μεταβλητή ή σύνθετο όρο.
- Μία σταθερά μπορεί να ταυτοποιηθεί μόνο με τον εαυτό της.
- Ένας σύνθετος όρος μπορεί να ταυτοποιηθεί με έναν άλλο σύνθετο μόνο εφόσον έχουν το ίδιο συναρτησιακό σύμβολο και την ίδια τάξη, και με την προϋπόθεση ότι τα αντίστοιχα ορίσματά τους μπορούν να ταυτοποιηθούν.

Κανόνες Ενοποίησης (2/2)

	<σταθερά> c1	<μεταβλητή> X1	<σύνθετος όρος> f(t1,…,tv)
<σταθερά> c2	επιτυχές αν η c1 είναι ίδια με τη c2	η μεταβλητή X1 παίρνει την τιμή c2 {X1=c2}	αποτυγχάνει
<μεταβλητή> X2	η μεταβλητή X2 παίρνει την τιμή c1 {X2=c1}	η μεταβλητή X1 ενοποιείται με τη X2 {X1=X2}	επιτυχές αν {X2=f(u1,…,um)}
<σύνθετος όρος> f(u1,…,uμ)	αποτυγχάνει	επιτυχές αν {X1=f(t1,…,tn)}	επιτυχές αν ν==μ και u1==t1,…,uμ==tv

Παραδείγματα Ενοποίησης Ατομικών Τύπων (1/2)

age(X, 17) – age(tom, 17)

{X=tom}

age(X, 17) - age(tom, 18)

- Δεν ενοποιούνται καθώς το δεύτερο όρισμα έχει διαφορετικές τιμές.

age(X, 17) - agel(X, 17)

- Δεν ενοποιούνται καθώς πρόκειται για διαφορετικά κατηγορήματα (functors).

a(X, X) - a(bob, Y)

{X=Y=bob}

Παραδείγματα Ενοποίησης Ατομικών Τύπων (2/2)

a(X, X) - a(bob, tom)

- Δεν ενοποιούνται . Η μεταβλητή X μπορεί να πάρει μόνο μια τιμή.

a(f (1, X) , 3) – a(f (Y, k) , Z)

{Y=1, X=k, Z=3}

a(3, X) – a(Y, f(2,Y))

{Y=3, X=f(2, 3) }

a(f(1,2) ,3) – a(f(Y, 2, 3) , Z)

- Δεν ενοποιούνται γιατί τα πρώτα ορίσματα στους δύο ατομικούς όρους έχουν διαφορετικό αριθμό ορισμάτων.

Παράδειγμα εκτέλεσης προγράμματος (1/2)

- Έστω το πρόγραμμα

  greek(socrates).

  human(turing).

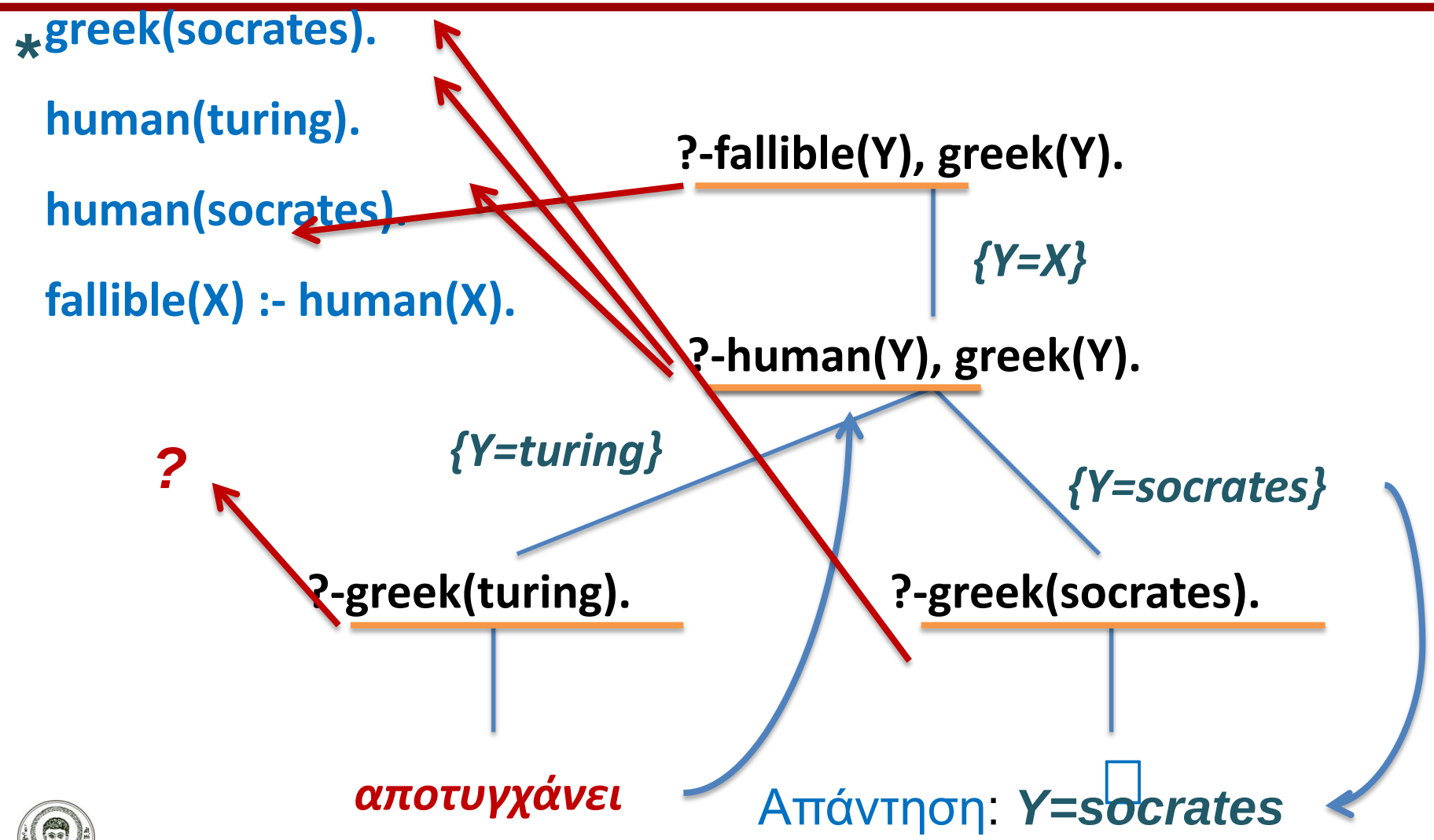
  human(socrates).

  fallible(X):- human(X).
- Τίθεται το ερώτημα.

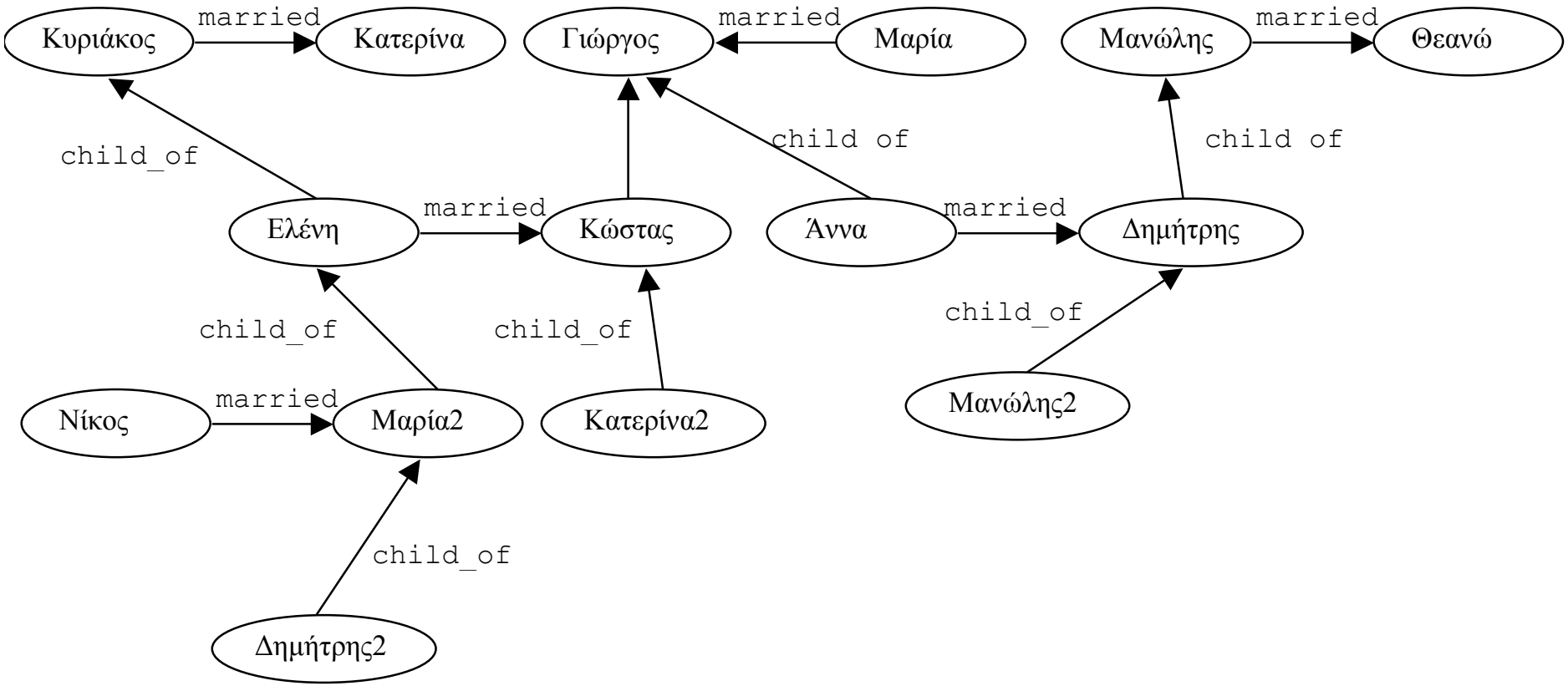
  ?- fallible(Y), greek(Y).
- Απάντηση:

  Y=socrates

Παράδειγμα εκτέλεσης προγράμματος (2/2)



Γενεαλογικό Δένδρο (εναλλακτικά)



Γενεαλογικό Δένδρο (εναλλακτικά)

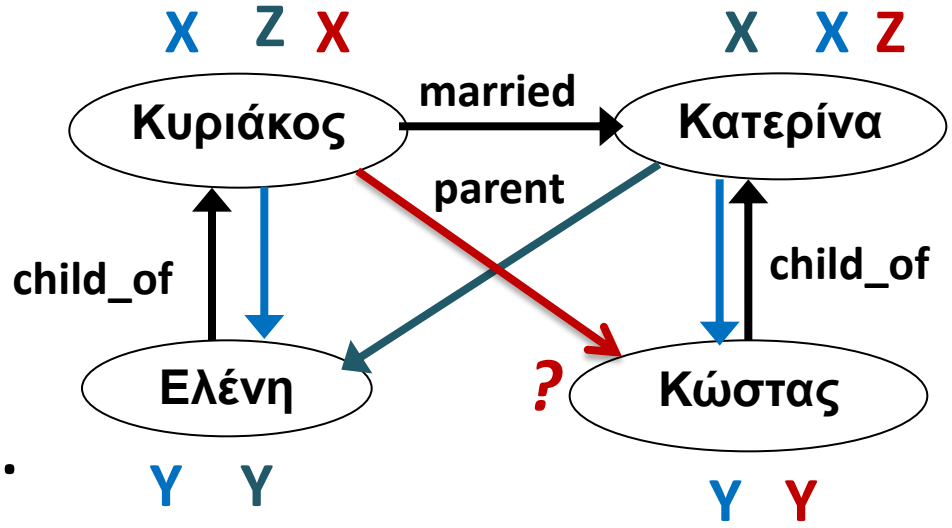
Κατηγορήματα

- Γεγονότα.

```
married('Κυριάκος','Κατερίνα').
...
child_of('Ελένη','Κυριάκος').
...
```

- Να ορισθεί η σχέση **parent/2.**

```
parent(X,Y) :-
    child_of(Y,X).
parent(X,Y) :-
    child_of(Y,Z),
    married(Z,X).
```




```
parent(X,Y) :-
        child_of(Y,Z),
        married(X,Z).
```

Γενεαλογικό Δένδρο (εναλλακτικά)

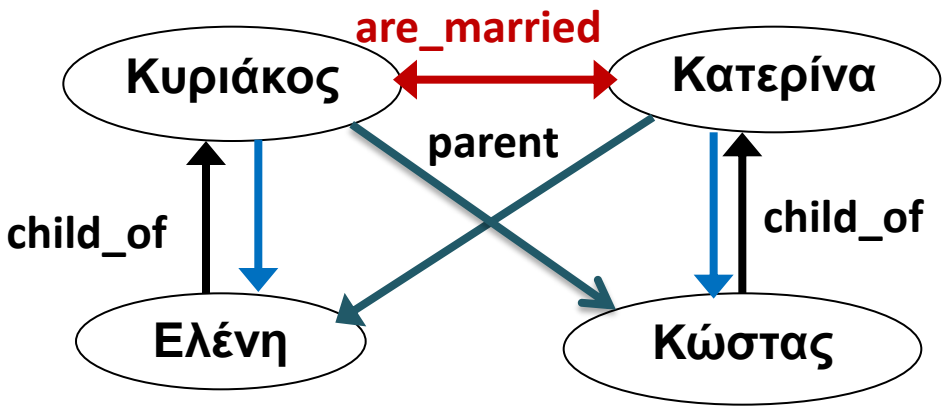
Αντιμεταθετική σχέση

- Η σχέση **married/2** είναι μονής κατεύθυνσης.
  - Να ορισθεί μία ανάλογη αντιμεταθετική σχέση.

```
are_married(X,Y) :- married(X,Y).
are_married(X,Y) :- married(Y,X).
```

- Η σχέση **parent/2** μπορεί να γραφεί

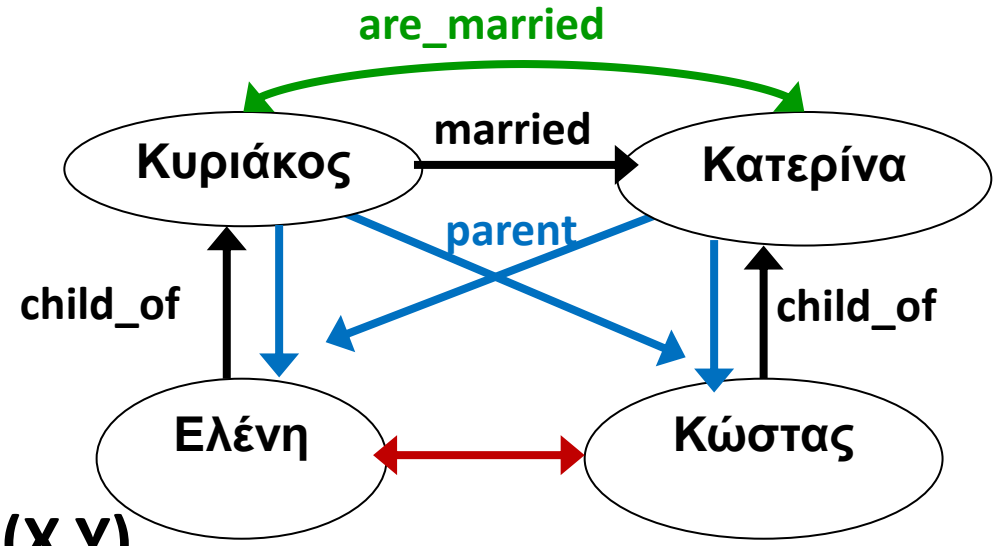
```
parent(X,Y) :-
    child_of(Y,X).
parent(X,Y) :-
    child_of(Y,Z),
    are_married(Z,X).
```



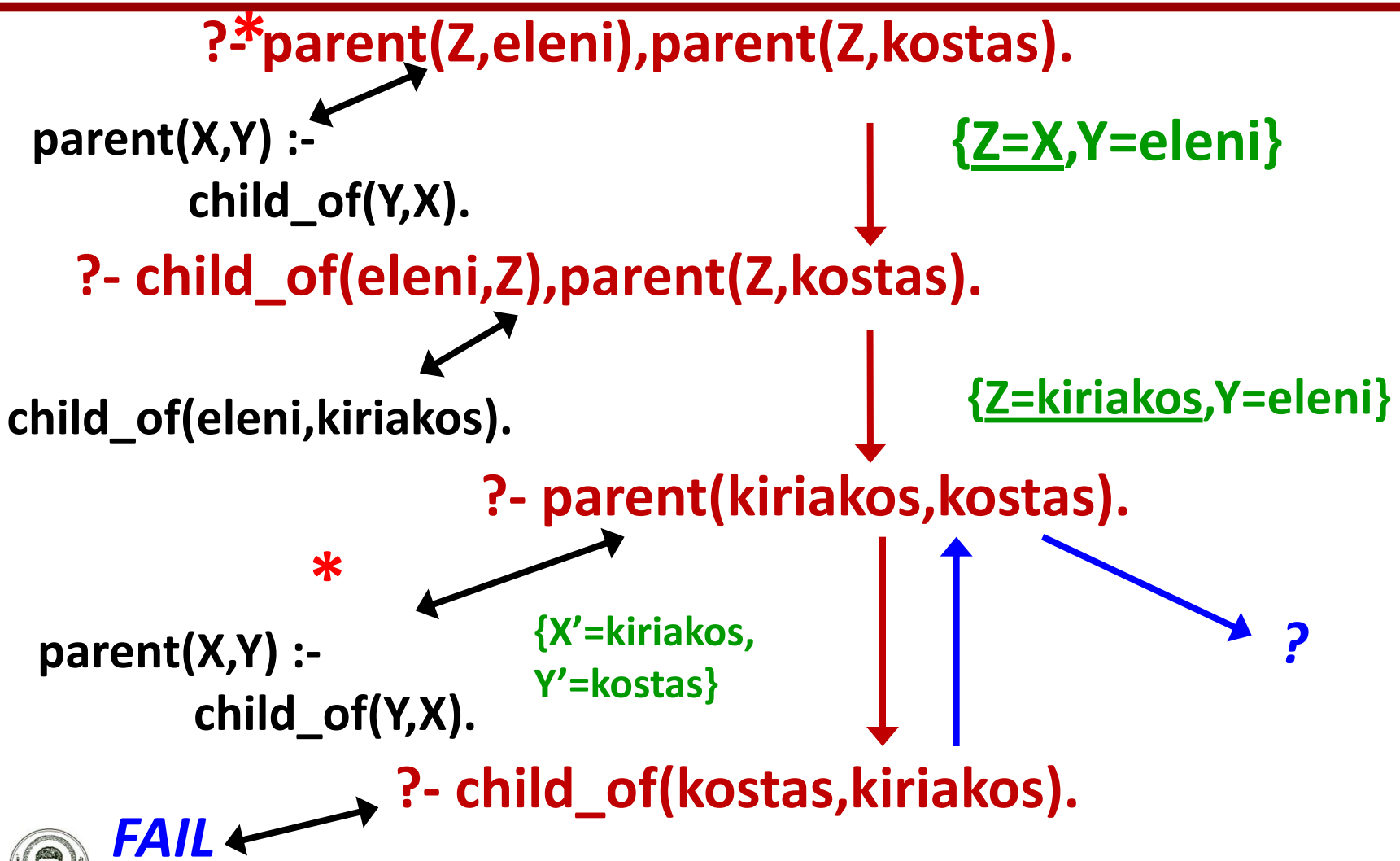
Γενεαλογικό Δένδρο (εναλλακτικά)

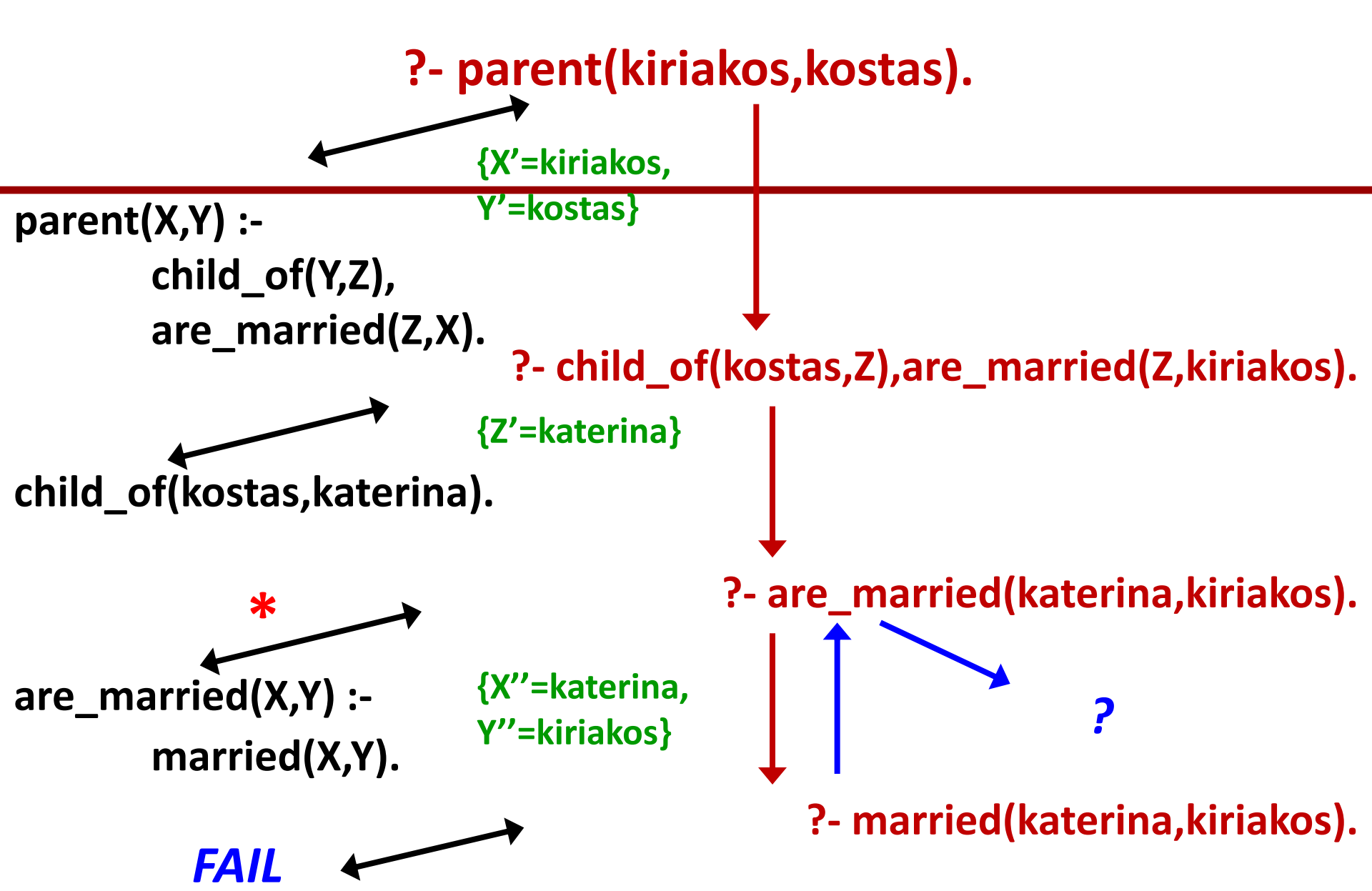
Μηχανισμός εκτέλεσης

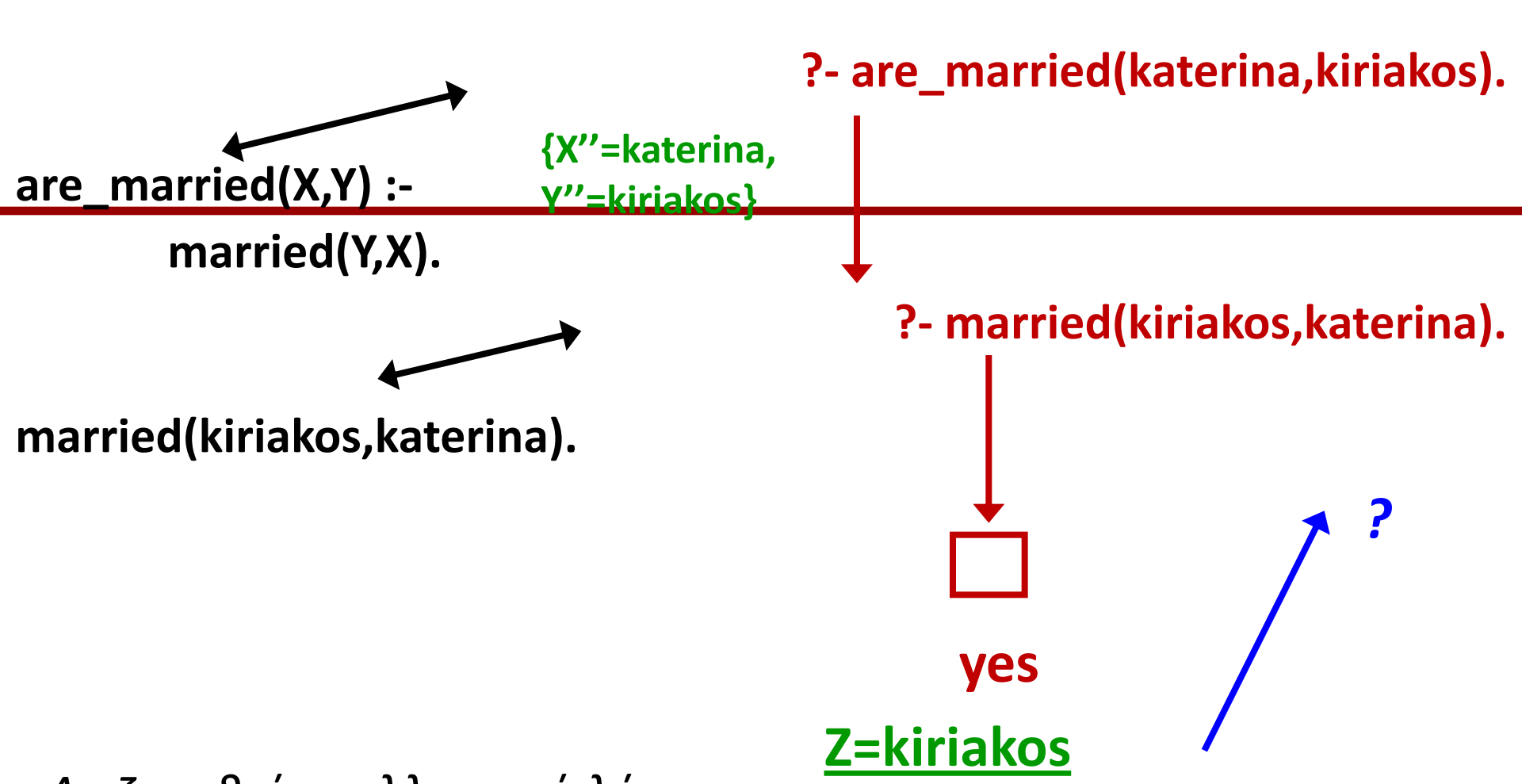
```
parent(X,Y) :-
    child_of(Y,X).
parent(X,Y) :-
    child_of(Y,Z),
    are_married(Z,X).
are_married(X,Y) :- married(X,Y).
are_married(X,Y) :- married(Y,X).
married(kiriakos,katerina).
child_of(eleni,kiriakos).
child_of(kostas,katerina).
?- parent(Z,eleni),parent(Z,kostas).
```



titlos


?-parent(Z,eleni),parent(Z,kostas).
parent(X,Y) :-
child_of(Y,X).
{Z=X,Y=eleni}
?- child_of(eleni,Z),parent(Z,kostas).
child_of(eleni,kiriakos).
{Z=kiriakos,Y=eleni}
?- parent(kiriakos,kostas).
parent(X,Y) :-
child_of(Y,X).
{X'=kiriakos,
Y'=kostas}
?
?- child_of(kostas,kiriakos).
FAIL


?- parent(kiriakos,kostas).
{X'=kiriakos,
Y'=kostas}
parent(X,Y) :-
child_of(Y,Z),
are_married(Z,X).
?- child_of(kostas,Z),are_married(Z,kiriakos).
{Z'=katerina}
child_of(kostas,katerina).
?- are_married(katerina,kiriakos).
*
are_married(X,Y) :-
married(X,Y).
{X''=katerina,
Y''=kiriakos}
?
?- married(katerina,kiriakos).
FAIL



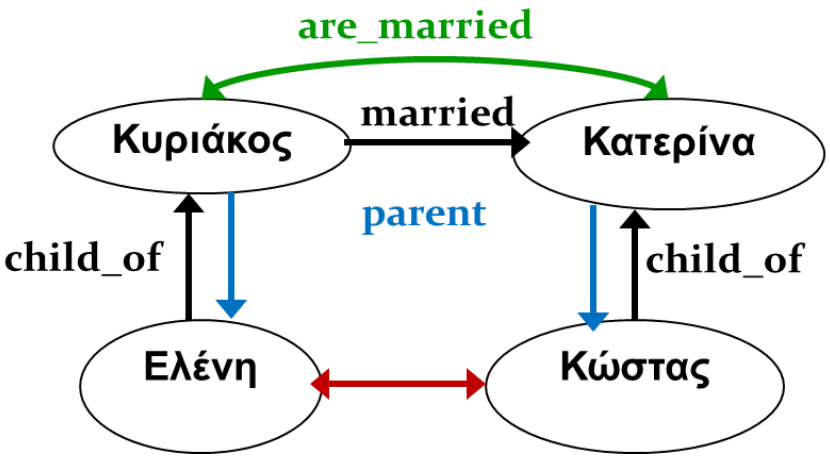
Αν ζητηθεί εναλλακτική λύση,
υπάρχει οπισθοδρόμηση στο πιο κοντινό (χρονικά) σημείο εναλλακτικής επιλογής, το οποίο ***ΔΕΝ*** *έχει χρησιμοποιηθεί σε προηγούμενη οπισθοδρόμηση.*

?- parent(Z,eleni),parent(Z,kostas).

parent(X,Y) :-
child_of(Y,Z'),
are_married(Z',X).

{Z=X,Y=eleni}

?- child_of(eleni,Z'),
are_married(Z',Z),
parent(Z,kostas).



...

{Z'=kiriakos,
Z=katerina}

yes
Z=katerina

Αναδρομή

- Η *αναδρομή* (*recursion*) αποτελεί βασικό στοιχείο στον προγραμματισμό με Prolog.
- Με τον όρο αναδρομή εννοούμε τη δυνατότητα ένας κανόνας να περιέχει στο σώμα του μια κλήση προς τον εαυτό του.
- Οι κανόνες που χρησιμοποιούν αναδρομή χαρακτηρίζονται σαν αναδρομικοί κανόνες.
- Η χρήση της αναδρομής οδηγεί σε μικρότερα προγράμματα.

Παράδειγμα Αναδρομής (1/4)

- Έστω τα γεγονότα:

parent(john,george). **parent(john,nick).**
parent(jim,bill). **parent(jim,jack).**
parent(gregory,john). **parent(gregory,jim).**
parent(bob,gregory). **parent(joseph,bob).**

- Θέλουμε να ορίσουμε έναν κανόνα **predecessor(X,Y)**, ο οποίος να αληθεύει αν ο **X** είναι πρόγονος του **Y**.

Παράδειγμα Αναδρομής (2/4)

- Λύση με μη αναδρομικούς κανόνες:

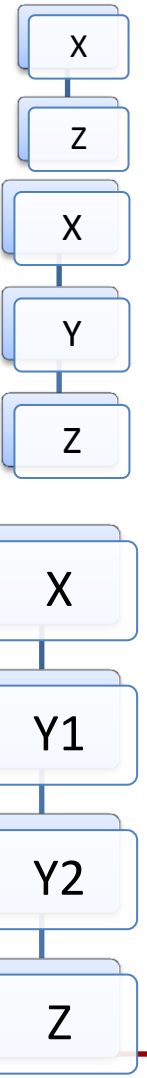
predecessor(X,Z) :- parent(X,Z).

predecessor(X,Z) :- parent(X,Y), parent(Y,Z).

predecessor(X,Z) :- parent(X,Y1),parent(Y1,Y2),
parent(Y2,Z).

…

Δεν μπορούν να γραφούν <u>άπειροι</u> κανόνες ώστε να καλύπτονται <u>όλες</u> οι γενεές.



Παράδειγμα Αναδρομής (3/4)

- Με αυτόν τον τρόπο δεν μπορούμε να ορίσουμε τη σχέση για πολύ μακρινούς απογόνους.
- Π.χ. στην ακόλουθη ερώτηση δεν δίνονται όλες οι απαντήσεις.
  - Λείπει η ***X=joseph***

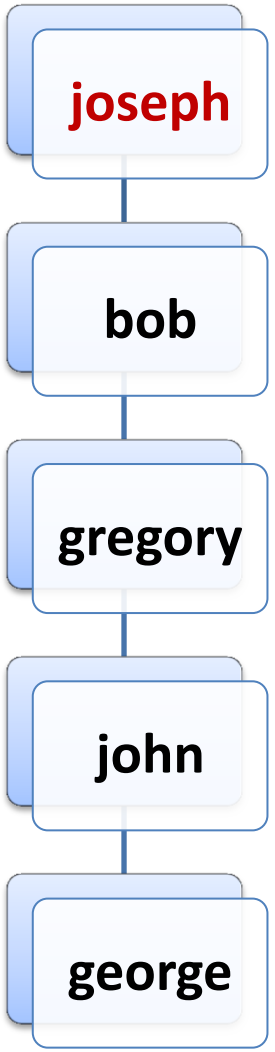
?- predecessor(X, george).

X=john;

X=gregory;

X=bob;

no



Παράδειγμα Αναδρομής (4/4)

- Λύση με αναδρομικούς κανόνες:
  - Ο αναδρομικός κανόνας διατρέχει τις γενεές ανεξάρτητα της απόστασής τους.

predecessor(X,Z):- parent(X,Z).

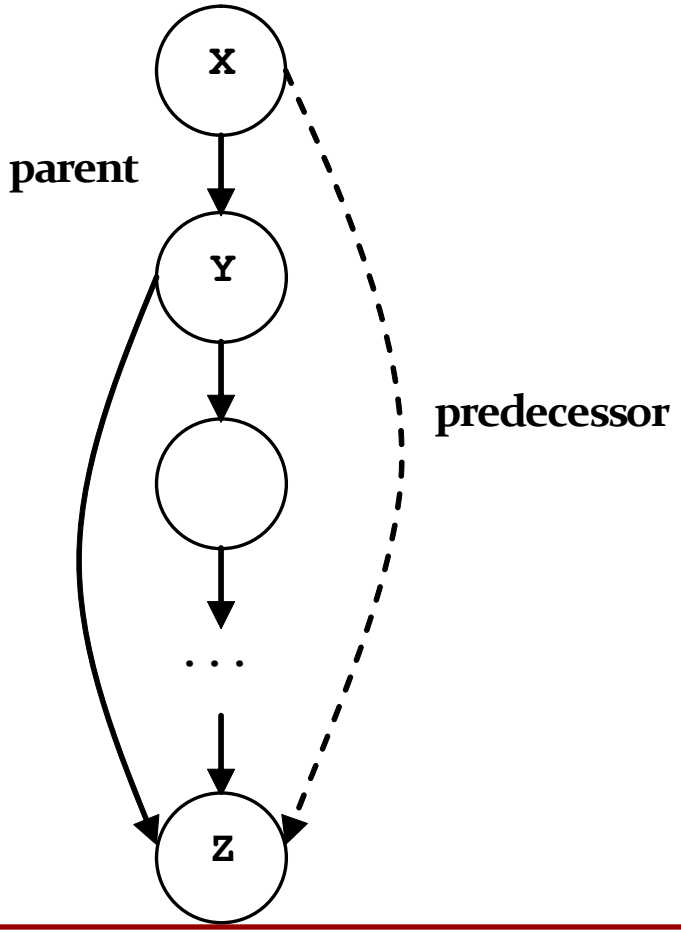
predecessor(X,Z):- parent(X,Y), predecessor(Y,Z).

- Παρατήρηση: Όταν ένα κατηγόρημα ορίζεται αναδρομικά, εμφανίζεται πάντα με τουλάχιστον δύο κανόνες.
  - Ο πρώτος συνήθως δεν περιέχει αναδρομική κλήση, λειτουργώντας σαν τερματική συνθήκη.

Αναδρομική σχέση predecessor

- Ο **X** είναι πρόγονος του **Z** όταν:
  - είτε ο **X** είναι πατέρας του **Z**,
  - είτε υπάρχει ένα **Y** τέτοιο ώστε ο **X** να είναι πατέρας του **Y** και ο **Y** να είναι πρόγονος του **Z**.

```
predecessor(X,Z):-
    parent(X,Z).
predecessor(X,Z):-
    parent(X,Y),
    predecessor(Y,Z).
```



Απαντήσεις αναδρομικής σχέσης

?- predecessor(X, george).

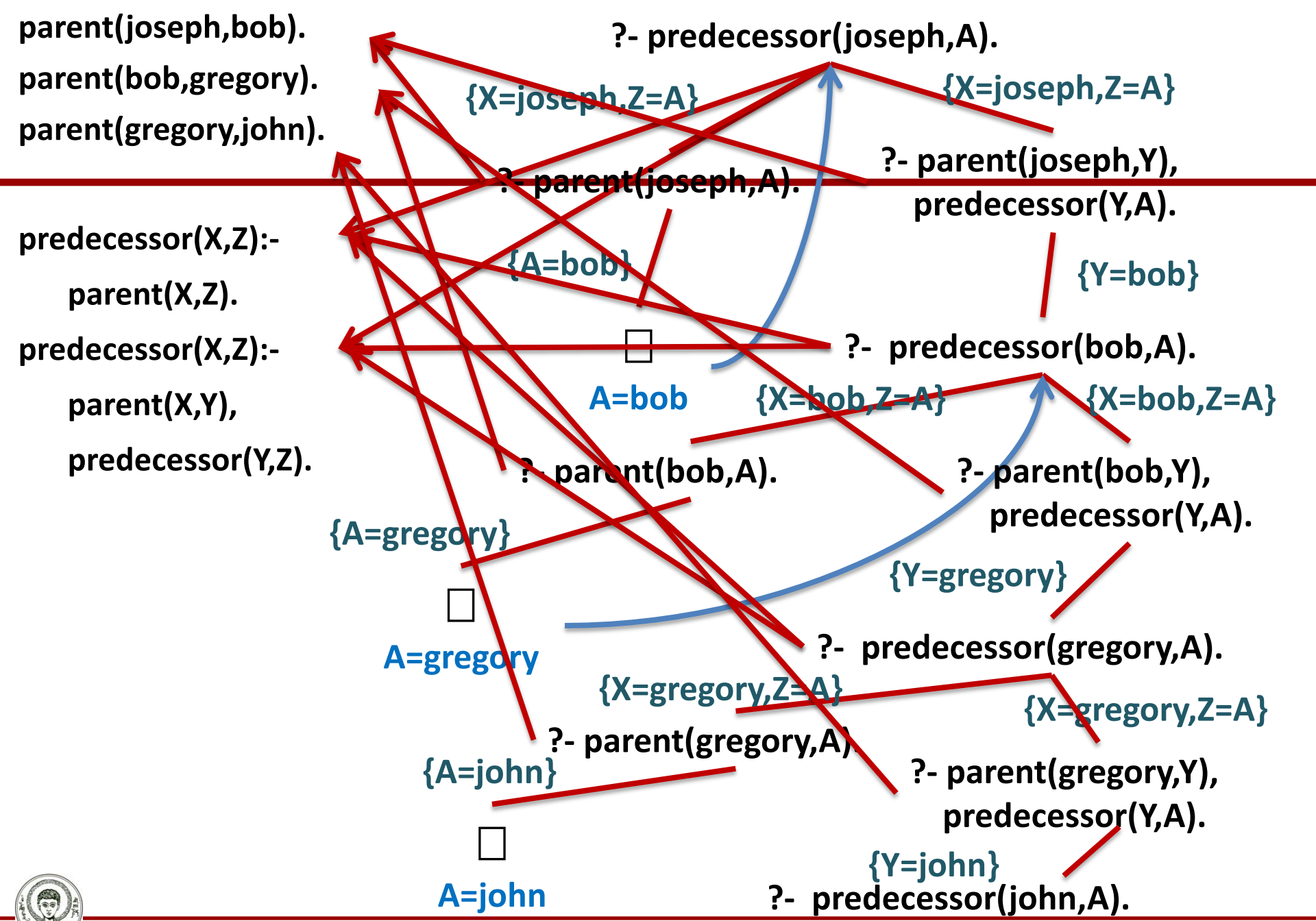
X=john;

X=gregory;

X=bob;

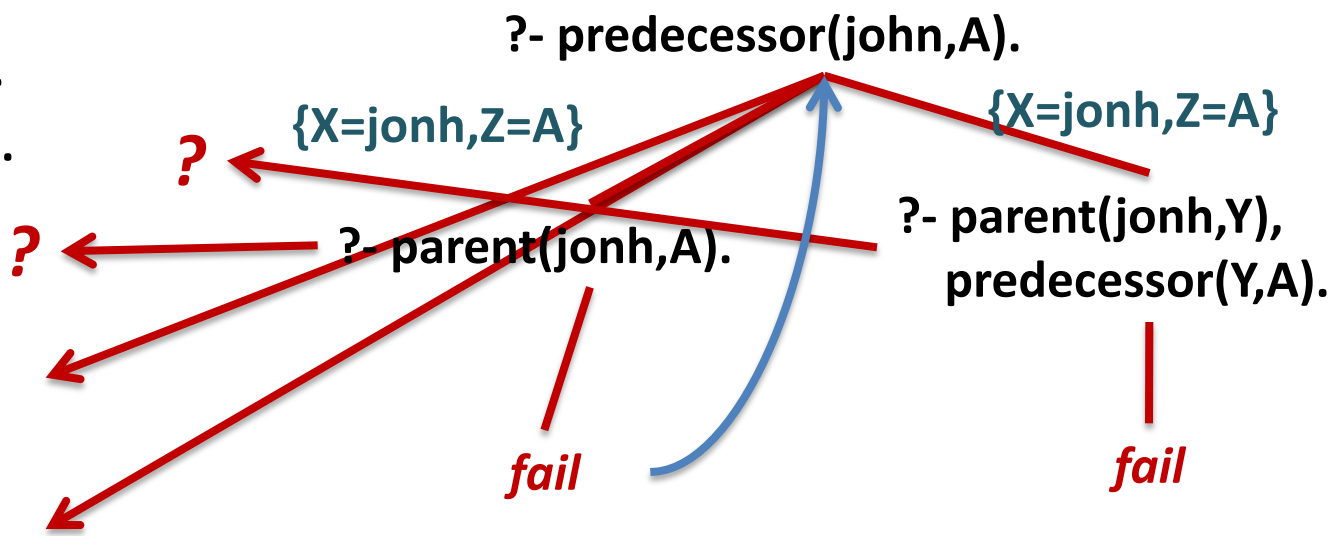
X=joseph;

no

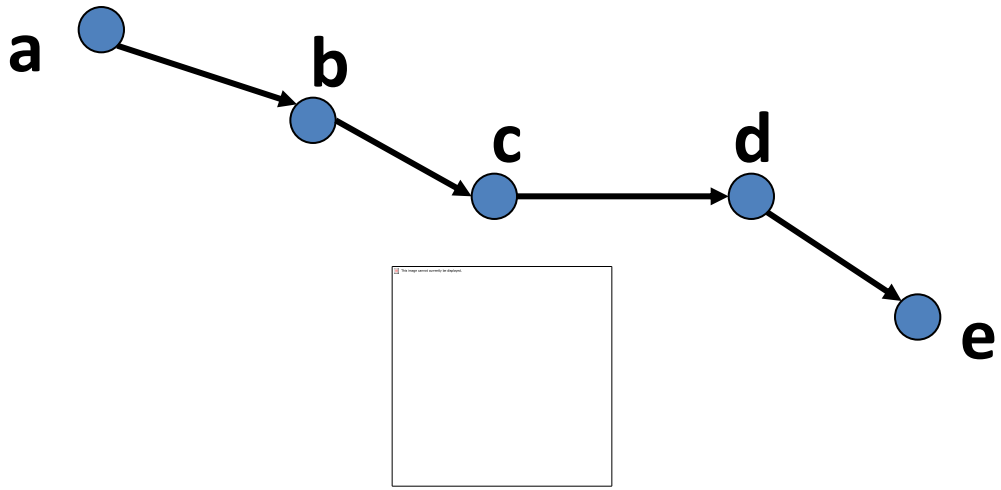

parent(joseph,bob).
parent(bob,gregory).
parent(gregory,john).
predecessor(X,Z):-
parent(X,Z).
predecessor(X,Z):-
parent(X,Y),
predecessor(Y,Z).
?- predecessor(joseph,A).
{X=joseph,Z=A}
{X=joseph,Z=A}
?- parent(joseph,A).
?- parent(joseph,Y),
predecessor(Y,A).
{A=bob}
{Y=bob}
?- predecessor(bob,A).
A=bob
{X=bob,Z=A}
{X=bob,Z=A}
?- parent(bob,A).
?- parent(bob,Y),
predecessor(Y,A).
{A=gregory}
{Y=gregory}
A=gregory
?- predecessor(gregory,A).
{X=gregory,Z=A}
{X=gregory,Z=A}
?- parent(gregory,A).
?- parent(gregory,Y),
predecessor(Y,A).
{A=john}
{Y=john}
A=john
?- predecessor(john,A).

```
parent(joseph,bob).
parent(bob,gregory).
parent(gregory,john).

predecessor(X,Z):-
    parent(X,Z).
predecessor(X,Z):-
    parent(X,Y),
    predecessor(Y,Z).
```



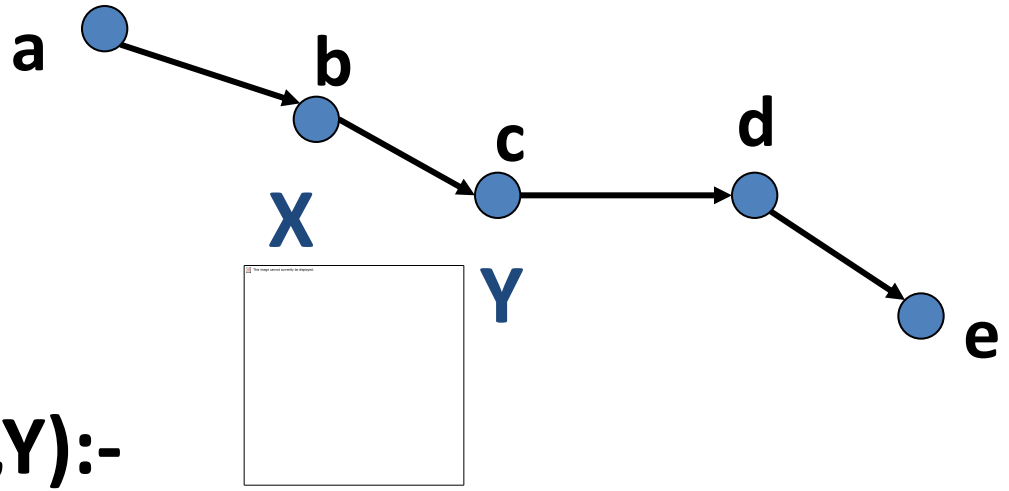
Παράδειγμα Γράφου Πόλεων (1/8)



road(a,b). road(c,d).

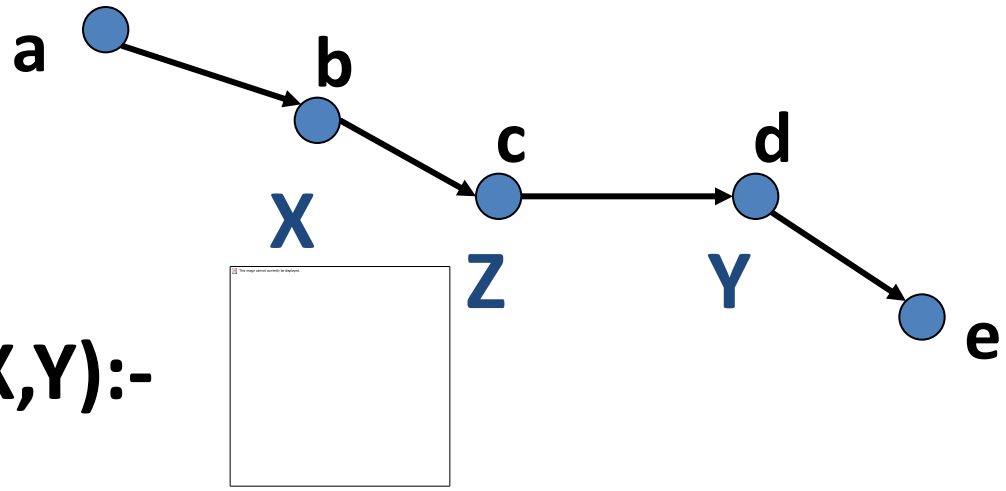
road(b,c). road(d,e).

Παράδειγμα Γράφου Πόλεων (2/8)



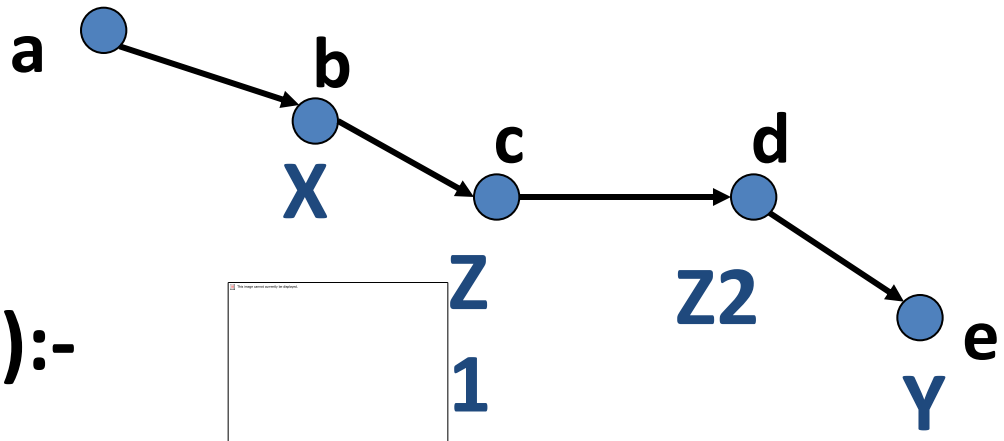

```
connected(X,Y):-
    road(X,Y).
```

Παράδειγμα Γράφου Πόλεων (3/8)



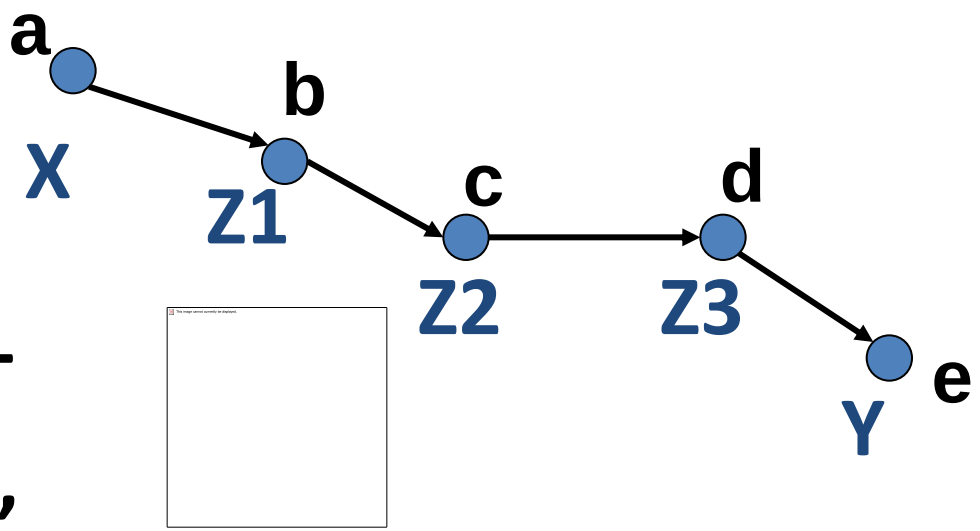

```
connected(X,Y):-
        road(X,Z),
        road(Z,Y).
```

Παράδειγμα Γράφου Πόλεων (4/8)



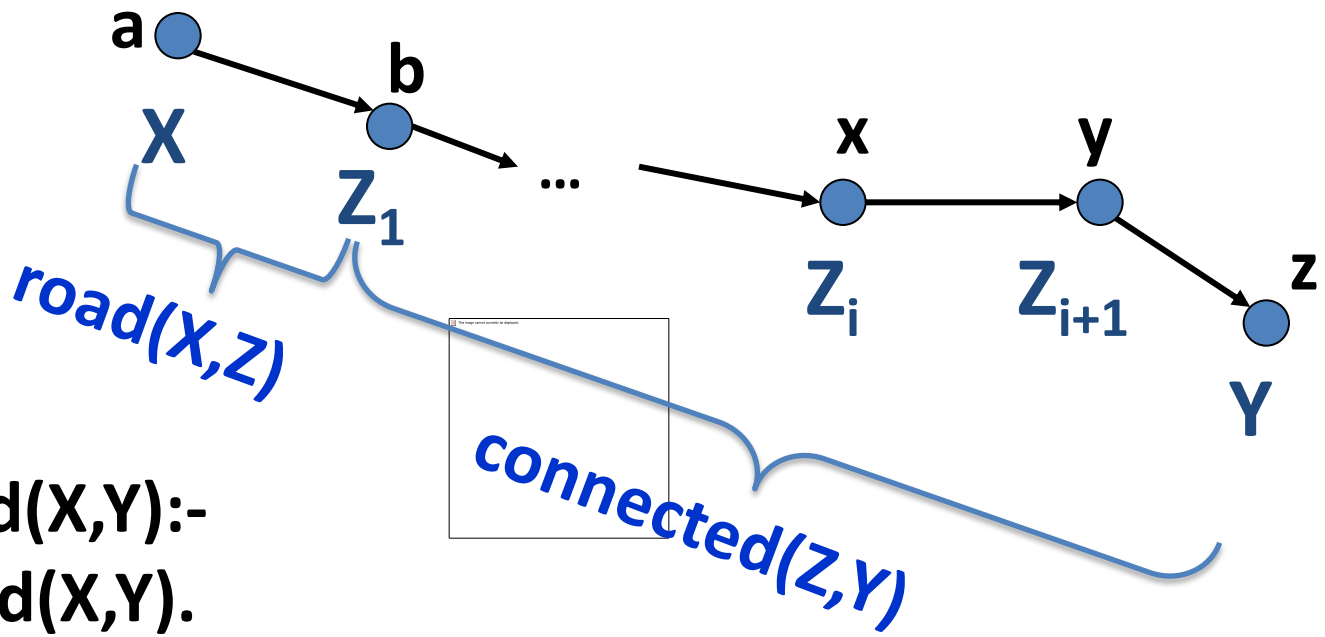

```
connected(X,Y):-
        road(X,Z1),
        road(Z1,Z2),
        road(Z2,Y).
```

Παράδειγμα Γράφου Πόλεων (5/8)



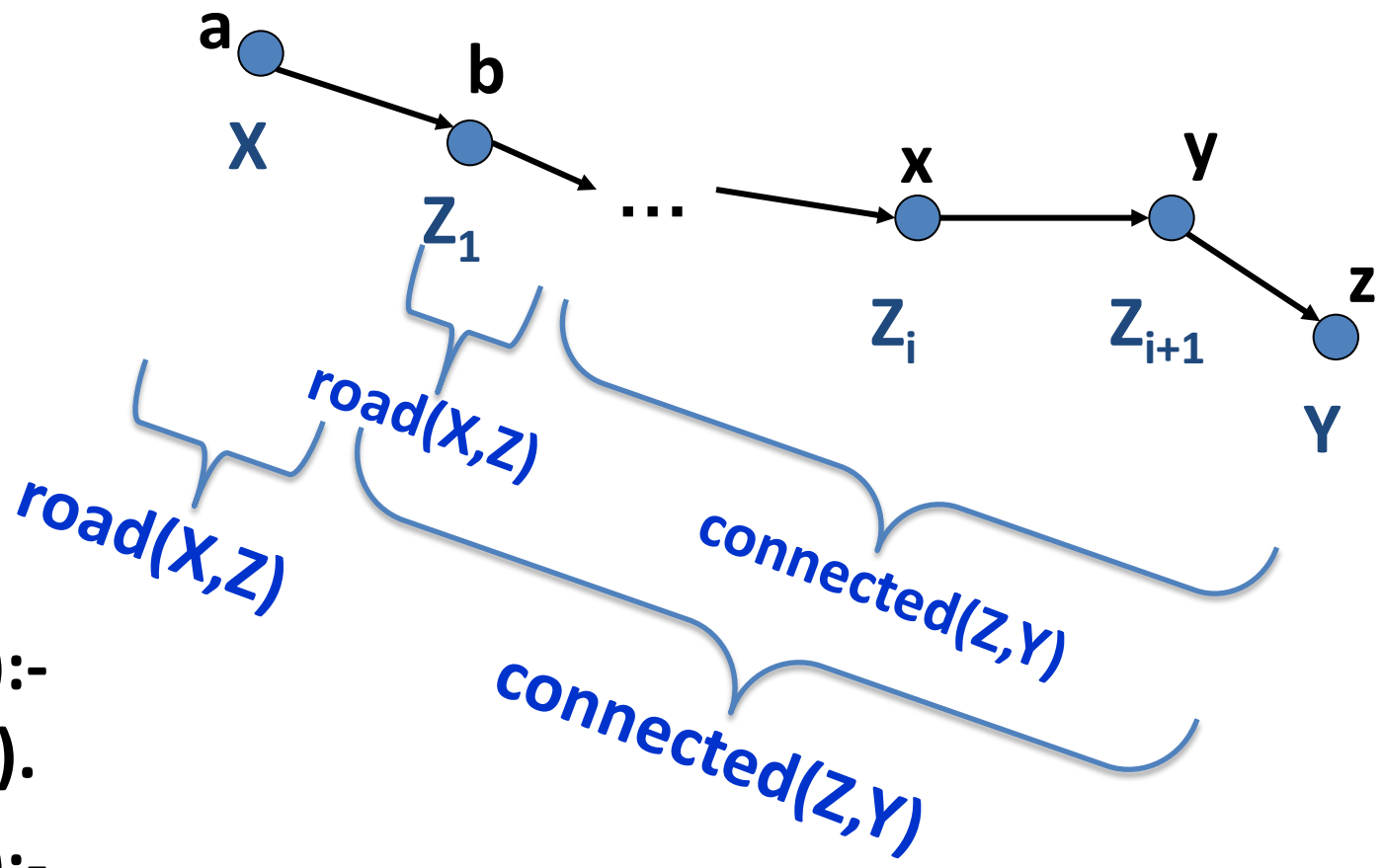

```
connected(X,Y):-
        road(X,Z1),
        road(Z1,Z2),
        road(Z2,Z3),
        road(Z3,Y).
```

Παράδειγμα Γράφου Πόλεων (6/8)



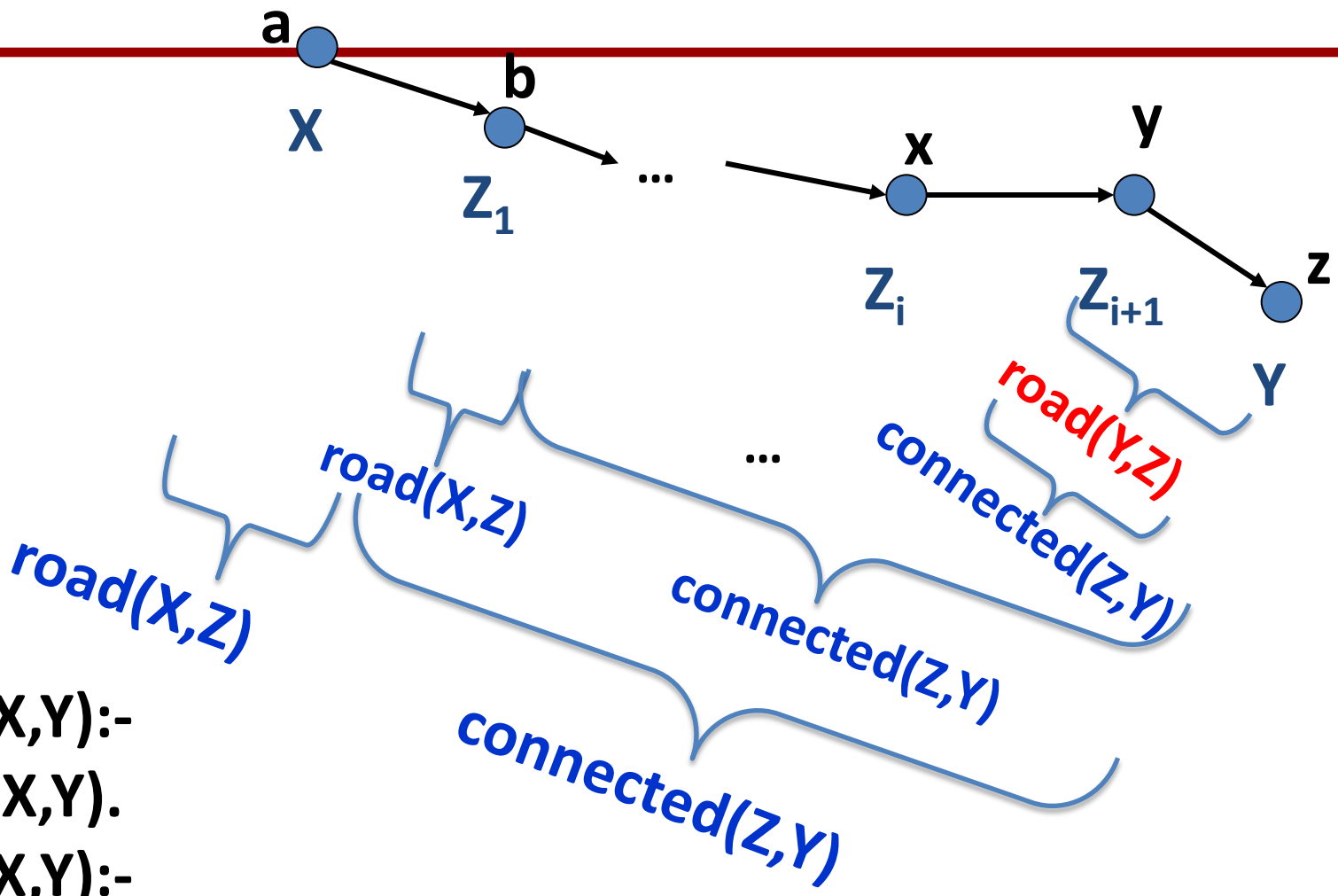

```
connected(X,Y):-
        road(X,Y).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

Παράδειγμα Γράφου Πόλεων (7/8)




```
connected(X,Y):-
        road(X,Y).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

Παράδειγμα Γράφου Πόλεων (8/8)

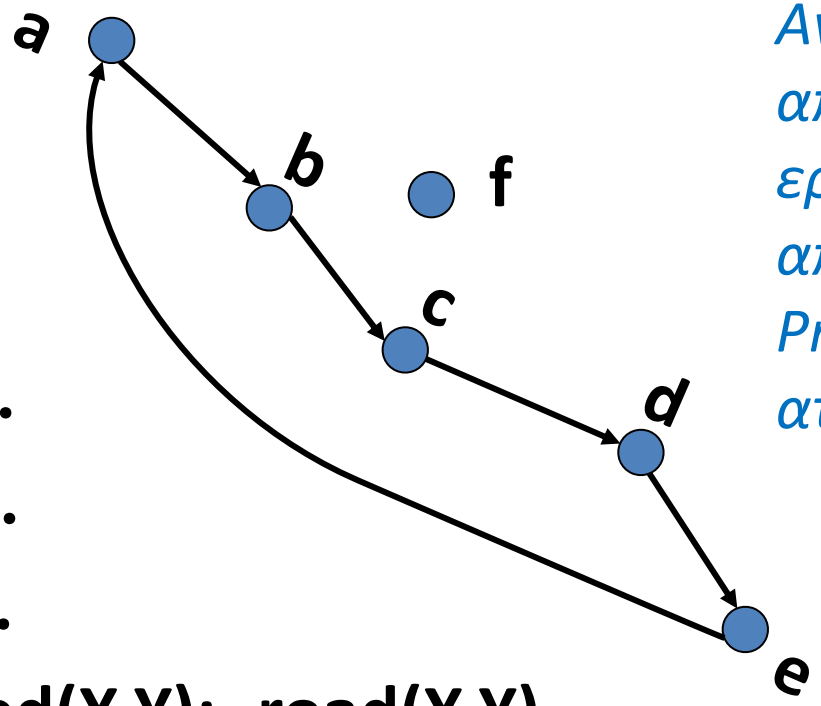



```
connected(X,Y):-
        road(X,Y).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

Κίνδυνοι Αναδρομής

- Ο βασικότερος κίνδυνος της αναδρομής είναι ο ατέρμονας βρόχος.
- Συμβαίνει όταν:
  - Δεν υπάρχει τερματικός κανόνας/γεγονός.
  - Υπάρχει τερματικός κανόνας, αλλά δεν ενοποιείται ποτέ με την αναδρομική κλήση ή αποτυγχάνει συνέχεια.
  - Τα δεδομένα πάνω στα οποία «τρέχουν» οι κανόνες έχουν κυκλική αλληλεξάρτηση.

Μη-ύπαρξη τερματικού κανόνα (1/3)



road(a,b).

road(b,c).

road(c,d).

road(d,e).

road(e,a).

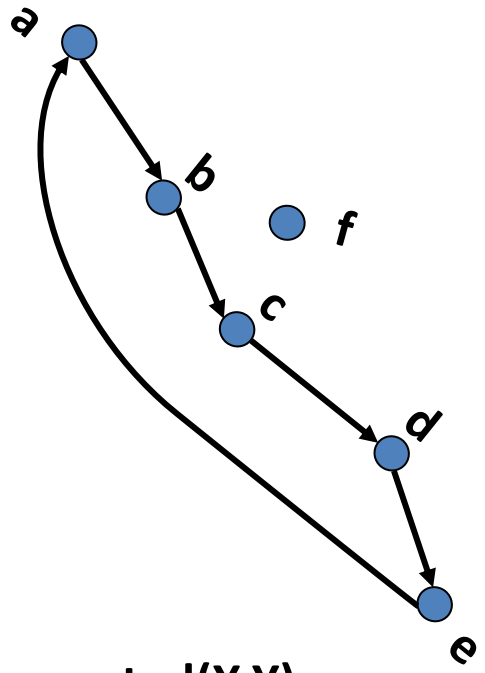
~~connected(X,Y):- road(X,Y).~~

connected(X,Y):- road(X,Z), connected(Z,Y).

?- connected(a,e).

Αν και έπρεπε να απαντηθεί yes, το ερώτημα δεν θα απαντηθεί ποτέ, γιατί η Prolog θα πέσει σε ατέρμονα βρόχο!

Μη-ύπαρξη τερματικού κανόνα (2/3)



~~connected(X,Y):-~~

~~road(X,Y).~~

connected(X,Y):-

road(X,Z),

connected(Z,Y).

?- connected(a,e).

?- road(a,Z),connected(Z,e).

{Z=b}

?- connected(b,e).

?- road(b,Z'),connected(Z',e).

{Z'=c}

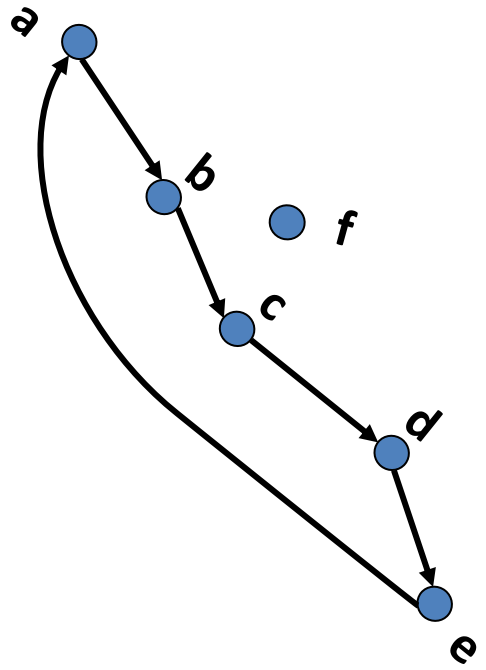
?- connected(c,e).

?- road(c,Z''),connected(Z'',e).

{Z''=d}

?- connected(d,e).

Μη-ύπαρξη τερματικού κανόνα (3/3)



~~connected(X,Y):-~~

~~road(X,Y).~~

connected(X,Y):-

road(X,Z),

connected(Z,Y).

?- connected(d,e).

?- road(d,Z’’’),connected(Z’’’,e).

{Z’’’=e}

?- connected(e,e).

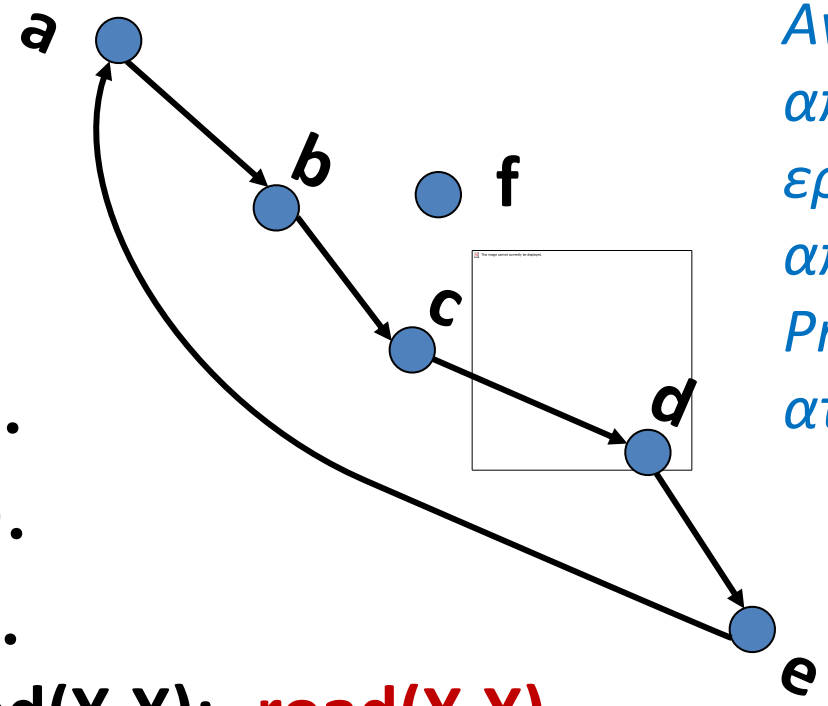
?- road(e,Z’’’’),connected(Z’’’’,e).

{Z’’’’=a}

?- connected(a,e).

- *Είναι η αρχική ερώτηση!*
- *Η εκτέλεση θα συνεχιστεί επ’ άπειρω!*

Αποτυχία τερματικού κανόνα (1/3)



road(a,b).

road(b,c).

road(c,d).

road(d,e).

road(e,a).

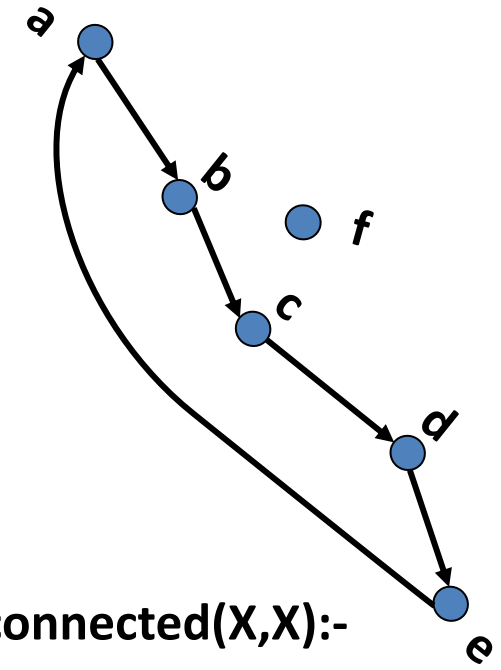
connected(X,X):- road(X,X).

connected(X,Y):- road(X,Z), connected(Z,Y).

?- connected(a,e).

Αν και έπρεπε να απαντηθεί yes, το ερώτημα δεν θα απαντηθεί ποτέ, γιατί η Prolog θα πέσει σε ατέρμονα βρόχο!

Αποτυχία τερματικού κανόνα (2/3)




```
connected(X,X):-
        road(X,X).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

?- connected(a,e).

no

?- road(a,Z),connected(Z,e).

{Z=b}

?- connected(b,e).

no

?- road(b,Z'),connected(Z',e).

{Z'=c}

?- connected(c,e).

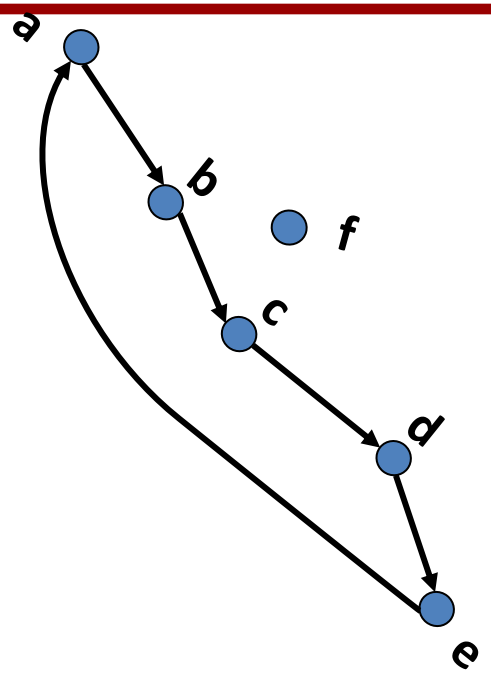
no

?- road(c,Z''),connected(Z'',e).

{Z''=d}

?- connected(d,e).

Αποτυχία τερματικού κανόνα (3/3)




```
connected(X,X):-
        road(X,X).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

?- connected(d,e).

no

?- road(d,Z'''),connected(Z''',e).

{Z'''=e}

?- connected(e,e).

?- road(e,e).

no

?- road(e,Z''''),connected(Z'''',e).

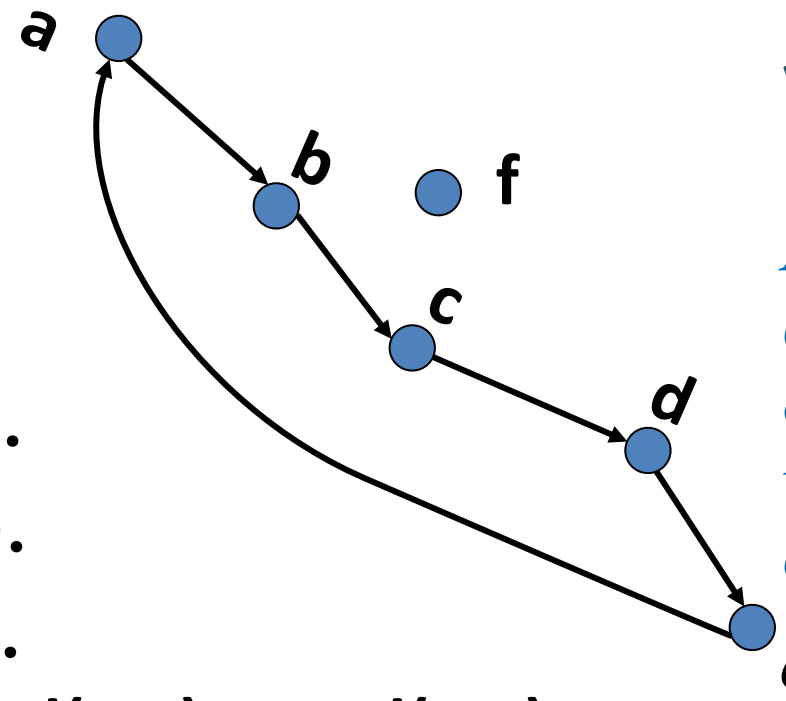
{Z''''=a}

?- connected(a,e).

- *Είναι η αρχική ερώτηση!*
- *Η εκτέλεση θα συνεχιστεί επ' άπειρο!*

Κυκλική Αλληλεξάρτηση Δεδομένων

Μεταβατική Ιδιότητα (1/3)




```
road(a,b).
road(b,c).
road(c,d).
road(d,e).
road(e,a).
```

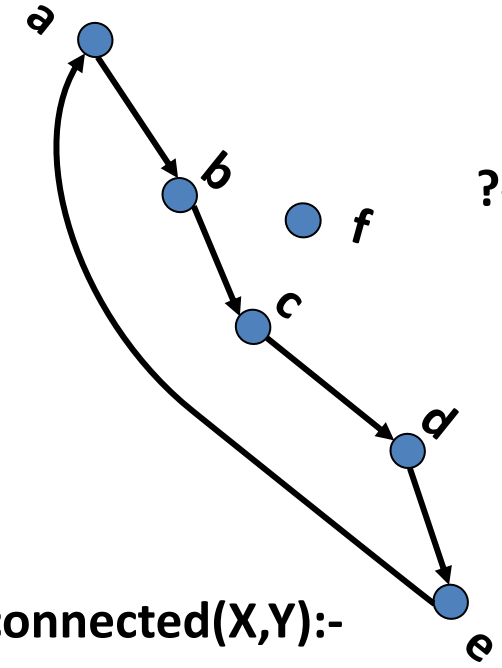
```
?- connected(a,e).
yes
?- connected(a,f).
```

Αν και έπρεπε να απαντηθεί no, το ερώτημα δεν θα απαντηθεί ποτέ, γιατί η Prolog θα πέσει σε ατέρμονα βρόχο!

```
connected(X,Y):- road(X,Y).
connected(X,Y):- road(X,Z), connected(Z,Y).
```

Κυκλική Αλληλεξάρτηση Δεδομένων

Μεταβατική Ιδιότητα (2/3)




```
connected(X,Y):-
        road(X,Y).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

?- connected(a,f).

?- road(a,f). — no

?- road(a,Z),connected(Z,f). — {Z=b}

?- connected(b,f).

?- road(b,f). — no

?- road(b,Z'),connected(Z',f). — {Z'=c}

?- connected(c,f).

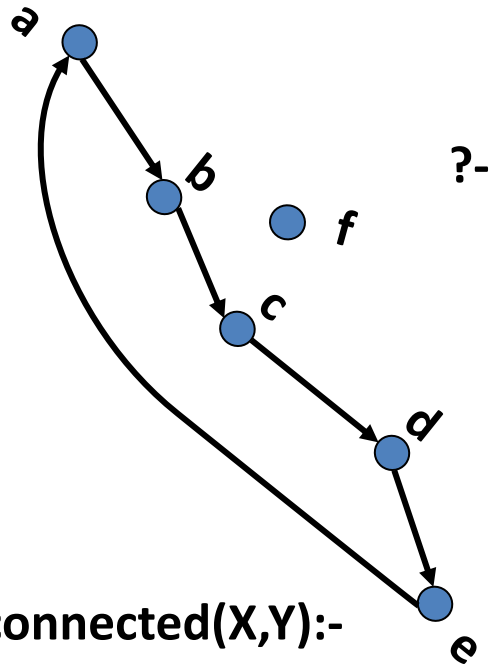
?- road(c,f). — no

?- road(c,Z''),connected(Z'',f). — {Z''=d}

?- connected(d,f).

Κυκλική Αλληλεξάρτηση Δεδομένων

Μεταβατική Ιδιότητα (3/3)




```
connected(X,Y):-
        road(X,Y).
connected(X,Y):-
        road(X,Z),
        connected(Z,Y).
```

?- connected(d,f).

?- road(d,f).

no

?- road(d,Z’’’),connected(Z’’’,f).

{Z’’’=e}

?- connected(e,f).

?- road(e,f).

no

?- road(e,Z’’’’),connected(Z’’’’,f).

{Z’’’’=a}

?- connected(a,f).

- *Είναι η αρχική ερώτηση!*
- *Η εκτέλεση θα συνεχιστεί επ’ άπειρο!*

Κυκλική Αλληλεξάρτηση Δεδομένων

Αντιμεταθετική Ιδιότητα (1/6)

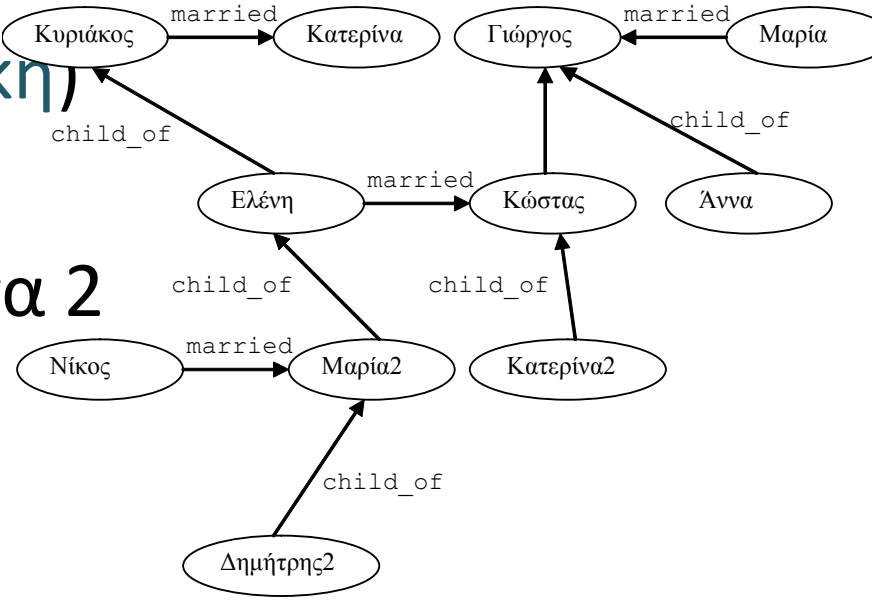
- Η σχέση **married/2** είναι μονής κατεύθυνσης.
- Πώς μπορεί να γίνει διπλής κατεύθυνσης? (Αντιμεταθετική)
- Λύσεις:

1. Να γραφούν όλα τα γεγονότα 2 φορές:

married(kiriakos,katerina).

married(katerina,kiriakos).

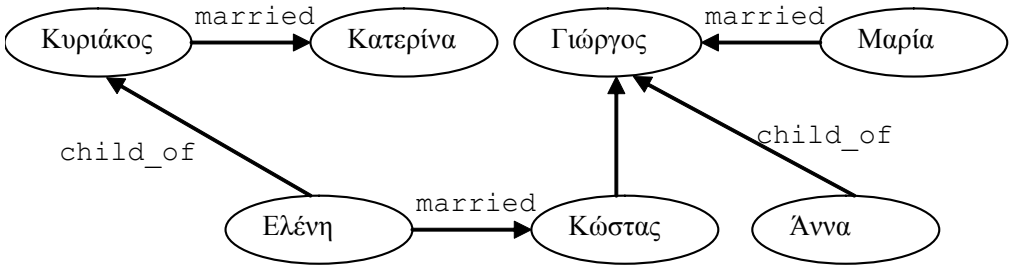
   - Μειονέκτημα: Πολύς κόπος από τον προγραμματιστή και σπατάλη χώρου



Κυκλική Αλληλεξάρτηση Δεδομένων

Αντιμεταθετική Ιδιότητα (2/6)

2. Να γραφεί αναδρομικός κανόνας που να δηλώνει ότι η σχέση είναι αντιμεταθετική.



married(kiriakos,katerina).

married(maria,george).

married(X,Y) :- married(Y,X).

- Μειονέκτημα: Η Prolog πέφτει σε ατέρμονα βρόχο λόγω κυκλικής αλληλεξάρτησης δεδομένων.

Κυκλική Αλληλεξάρτηση Δεδομένων

Αντιμεταθετική Ιδιότητα (3/6)

- Το πρόγραμμα απαντάει **yes** όταν πρέπει να απαντήσει **yes.**

married(kiriakos,katerina).

married(maria,george).

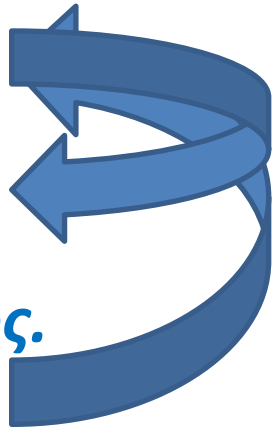
married(X,Y) :- married(Y,X).

?- **married(kiriakos,katerina).**

yes *% λόγω γεγονότος.*

?- **married(katerina,kiriakos).**

yes *% λόγω κανόνα και γεγονότος.*



{X=katerina, Y=kiriakos}

Κυκλική Αλληλεξάρτηση Δεδομένων

Αντιμεταθετική Ιδιότητα (4/6)

- Το πρόγραμμα απαντάει **no** όταν πρέπει να απαντήσει **no**?

married(kiriakos,katerina).

married(maria,george).

married(X,Y) :- married(Y,X).

- *Είναι η αρχική ερώτηση!*
- *Η εκτέλεση θα συνεχιστεί επ' άπειρο!*

Κυκλική Αλληλεξάρτηση Δεδομένων

Αντιμεταθετική Ιδιότητα (5/6)

3. Να γραφούν 2 μη-αναδρομικοί κανόνες που να δηλώνουν ότι η σχέση είναι αντιμεταθετική.

married(kiriakos,katerina).

married(maria,george).

are_married(X,Y) :- married(X,Y).

are_married(X,Y) :- married(Y,X).

?- are_married(kiriakos,katerina).

yes *% λόγω 1ου κανόνα + γεγονός*

?- **are_married(katerina,kiriakos).**

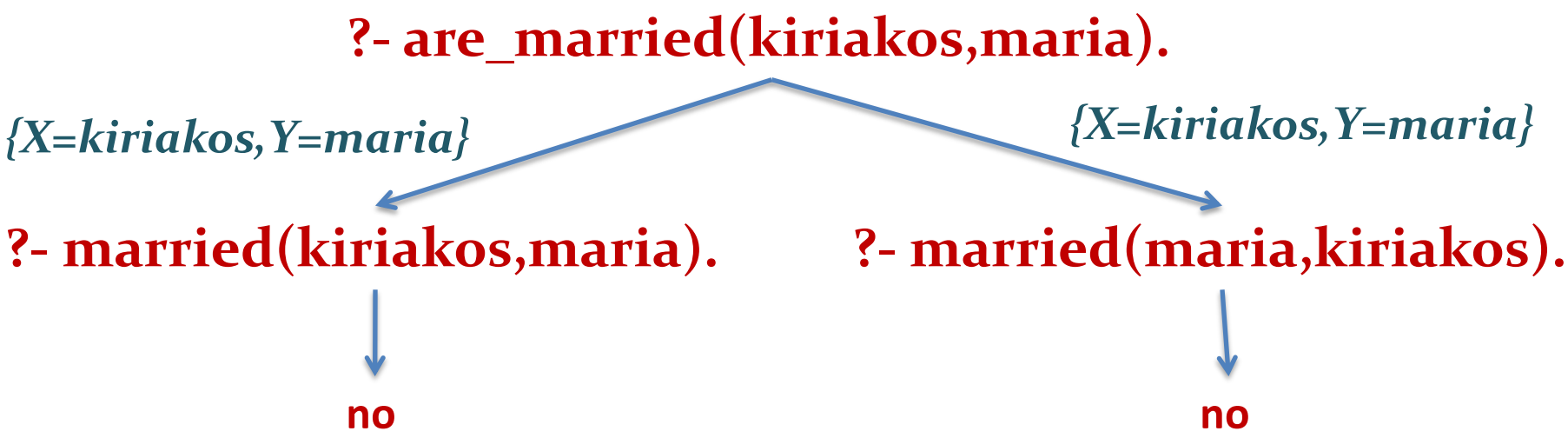
yes *% λόγω 2ου κανόνα + γεγονός*

?- married(kiriakos,katerina).

?- married(kiriakos,katerina).

Κυκλική Αλληλεξάρτηση Δεδομένων

Αντιμεταθετική Ιδιότητα (6/6)




```
married(kiriakos,katerina).

married(maria,george).

are_married(X,Y) :- married(X,Y).

are_married(X,Y) :- married(Y,X).
```

Κυκλική Αλληλεξάρτηση Δεδομένων

Μεταβατική Ιδιότητα – Εναλλακτικά (1/3)

- Γιατί η μεταβατική ιδιότητα σε γράφους απαιτεί δύο (2) κατηγορήματα;
  - Π.χ. **road/2** για τις απευθείας ακμές και **connected/2** για τις «μεταβατικές».

connected(X,Y):- road(X,Y).

connected(X,Y):- road(X,Z), connected(Z,Y).

- Γιατί δεν είναι εφικτός ο παρακάτω ορισμός;
  - Η μείξη δηλαδή άμεσων και έμμεσων ακμών;

road(a,b). road(b,c).

road(X,Y) :- road(X,Z), road(Z,Y).

Κυκλική Αλληλεξάρτηση Δεδομένων

Μεταβατική Ιδιότητα – Εναλλακτικά (2/3)

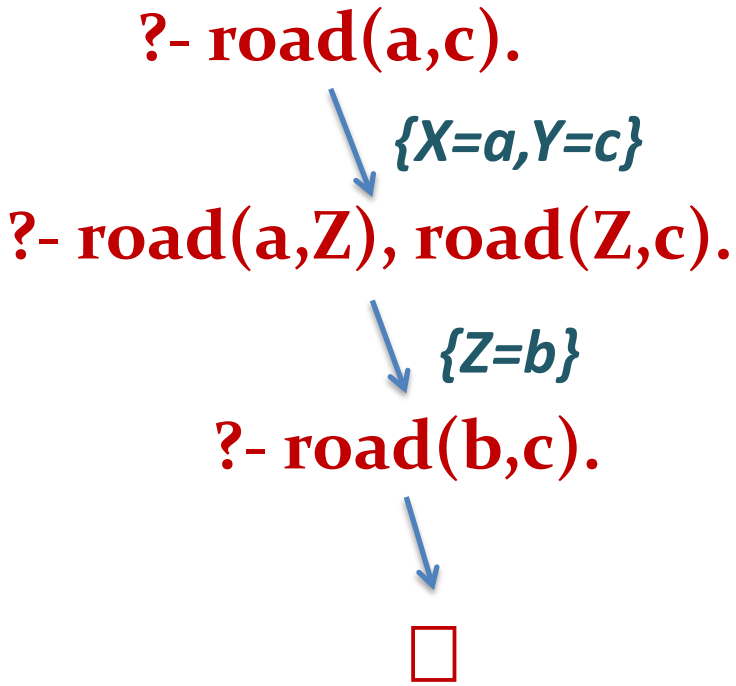
road(a,b). road(b,c). road(c,a).

road(X,Y) :- road(X,Z), road(Z,Y).

- Ένας τέτοιος ορισμός θα έδινε σωστές απαντήσεις στην καταφατική περίπτωση, αλλά θα έπεφτε σε ατέρμονα βρόχο στην αρνητική περίπτωση.
  - Ακριβώς όπως στην περίπτωση της αντιμεταθετικής ιδιότητας.

?- road(a,c).

yes



Κυκλική Αλληλεξάρτηση Δεδομένων

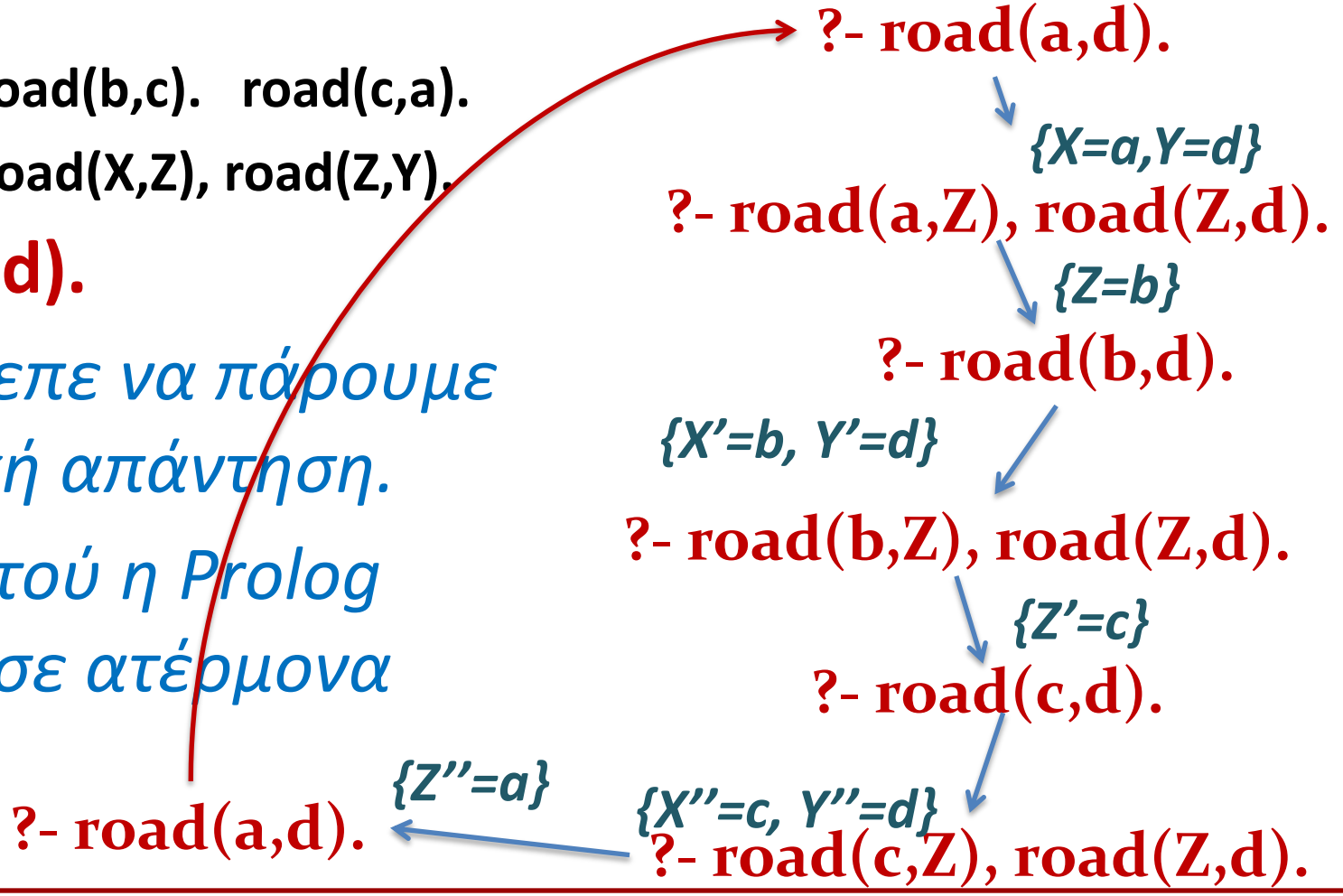
Μεταβατική Ιδιότητα – Εναλλακτικά (3/3)

road(a,b). road(b,c). road(c,a).

road(X,Y) :- road(X,Z), road(Z,Y).

?- road(a,d).

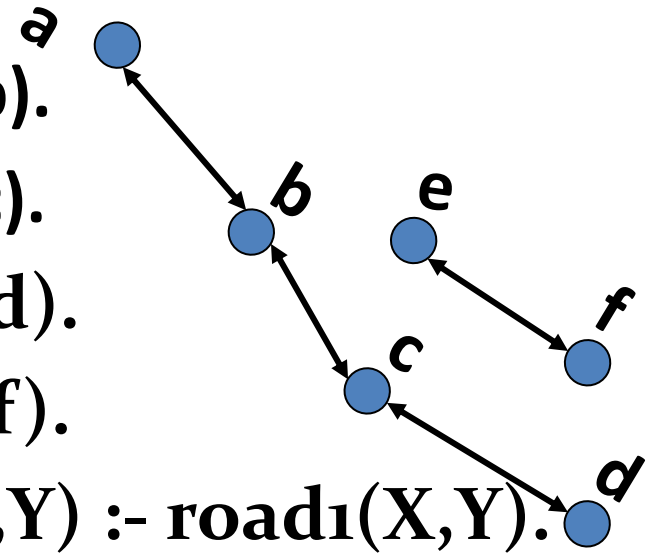
- *Θα έπρεπε να πάρουμε αρνητική απάντηση.*
- *Αντ' αυτού η Prolog πέφτει σε ατέρμονα βρόχο.*



Γενικό «Δίδαγμα»

- Όταν υπάρχει γράφος και αναδρομή καλό είναι τα κατηγορήματα που απεικονίζουν τις ακμές του γράφου (γεγονότα) να είναι διαχωρισμένα από τα αναδρομικά κατηγορήματα που διασχίζουν το γράφο.
- Παρ’ όλα αυτά δεν είναι δυνατόν πάντα να γλυτώσουμε τα προβλήματα με την αναδρομή.
  - Π.χ. κυκλικά δεδομένα.
  - Μια ακόμα περίπτωση είναι ο συνδυασμός αντιμεταθετικής και μεταβατικής ιδιότητας.

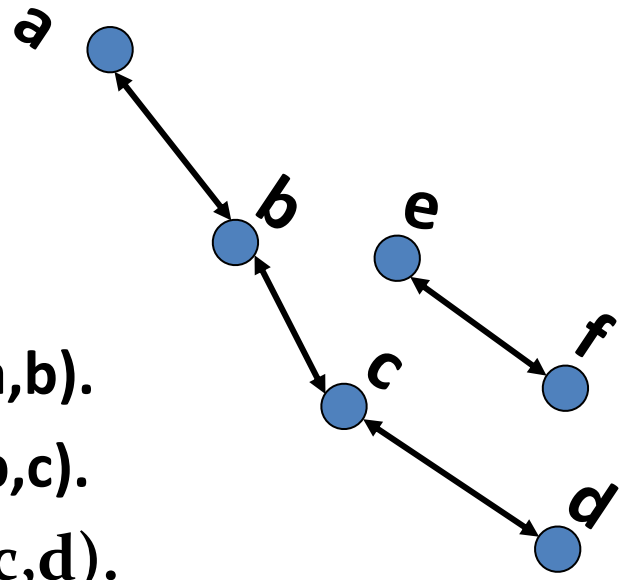
Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (1/12)




```
road1(a,b).
road1(b,c).
road1(c,d).
road1(e,f).
road2(X,Y) :- road1(X,Y).
road2(X,Y) :- road1(Y,X).
connected(X,Y):- road2(X,Y).
connected(X,Y):- road2(X,Z), connected(Z,Y).
```

- Το κατηγόρημα **road1/2** είναι μονής κατεύθυνσης, ενώ το **road2/2** αντιμεταθετικό.
- Το κατηγόρημα **connected/2** είναι σωστό;
  - Απαντάει ναι/όχι όταν πρέπει;

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (2/12)



road1(a,b).

road1(b,c).

road1(c,d).

road1(e,f).

road2(X,Y) :- road1(X,Y).

road2(X,Y) :- road1(Y,X).

connected(X,Y):- road2(X,Y).

connected(X,Y):- road2(X,Z), connected(Z,Y).

?- connected(a,c).

↓

?- road2(a,Z),connected(Z,c).

↓

?- road1(a,Z),connected(Z,c).

↓ {Z=b}

?- connected(b,c).

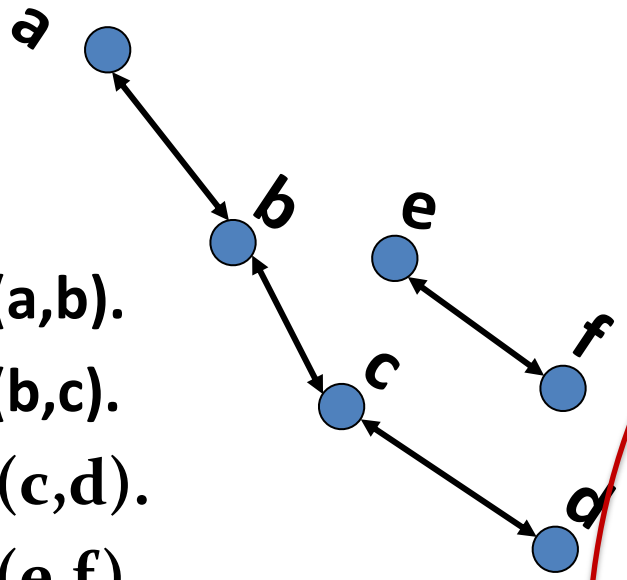
↓

?- road2(b,c).

↓

?- road1(b,c). □

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (3/12)



road1(a,b).

road1(b,c).

road1(c,d).

road1(e,f).

road2(X,Y) :- road1(X,Y).

road2(X,Y) :- road1(Y,X).

connected(X,Y):- road2(X,Y).

connected(X,Y):- road2(X,Z), connected(Z,Y).

?- connected(c,a).

↓

?- road2(c,Z),connected(Z,a).

↓

?- road1(Z,c),connected(Z,a).

↓ {Z=b}

?- connected(b,c).

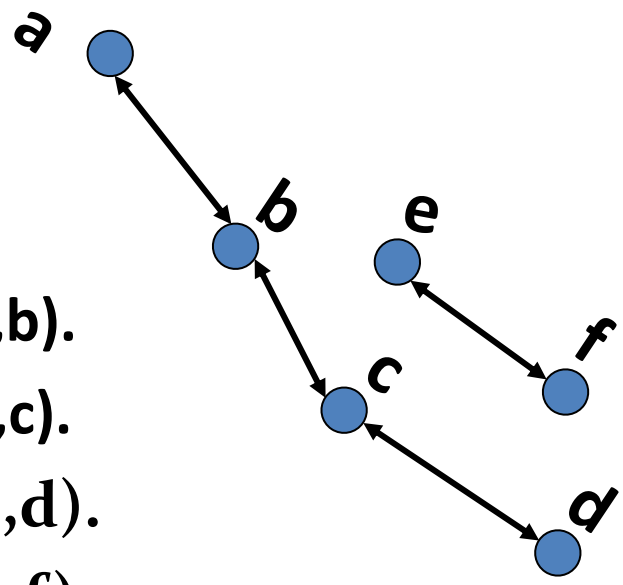
↓

?- road2(b,c).

↓

?- road1(b,c). □

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (4/12)



road1(a,b).

road1(b,c).

road1(c,d).

road1(e,f).

road2(X,Y) :- road1(X,Y).

road2(X,Y) :- road1(Y,X).

connected(X,Y):- road2(X,Y).

connected(X,Y):- road2(X,Z), connected(Z,Y).

?- connected(a,e).

?- road2(a,Z),connected(Z,e).

?- road1(a,Z),connected(Z,e).

{Z=b}

?- connected(b,e).

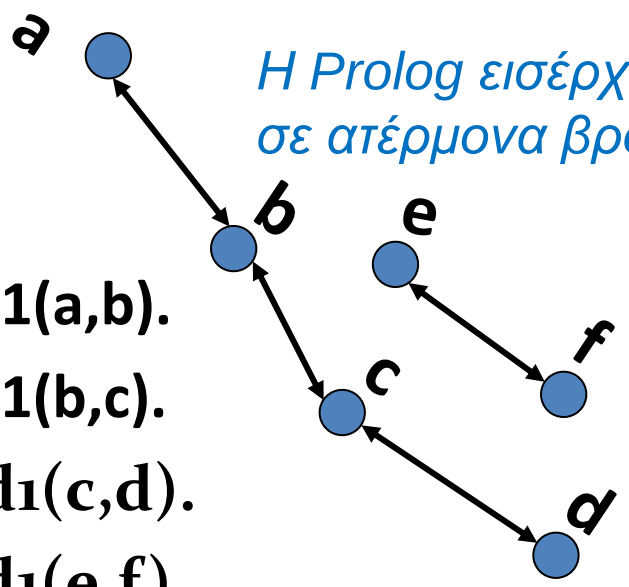
?- road2(b,Z),connected(Z,e).

?- road1(b,Z),connected(Z,e).

{Z=c}

?- connected(c,e).

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (5/12)



Η Prolog εισέρχεται σε ατέρμονα βρόχο!

road1(a,b).

road1(b,c).

road1(c,d).

road1(e,f).

road2(X,Y) :- road1(X,Y).

road2(X,Y) :- road1(Y,X).

connected(X,Y):- road2(X,Y).

connected(X,Y):- road2(X,Z), connected(Z,Y).

?- connected(c,e).

?- road2(c,Z),connected(Z,e).

?- road1(c,Z),connected(Z,e).

{Z=d}

?- connected(d,e).

?- road2(d,Z),connected(Z,e).

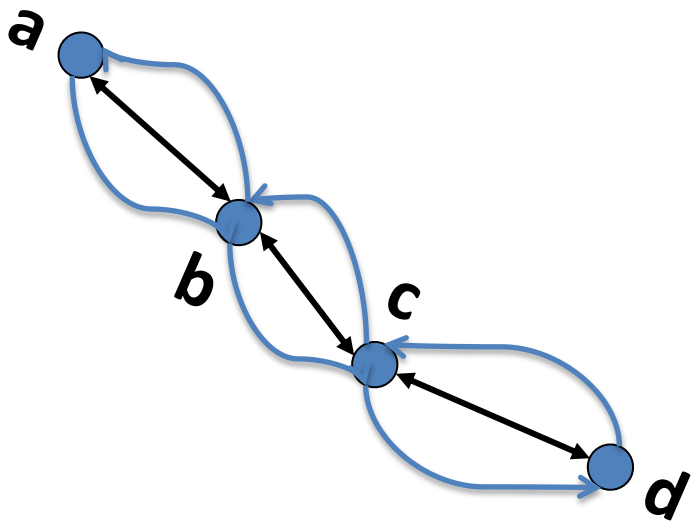
?- road1(Z,d),connected(Z,e).

{Z=c}

?- connected(c,e).

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (6/12)

- Το πρόβλημα προκαλείται από την αντιμεταθετική ιδιότητα, η οποία δημιουργεί κυκλικά δεδομένα.



Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (7/12)

- Λύση: Ο διαχωρισμός των κατηγορημάτων για την μεταβατική και αντιμεταθετική ιδιότητα.

```
road(a,b).
road(b,c).
road(c,d).
road(e,f).
```

Γεγονότα ακμών απλής κατεύθυνσης.

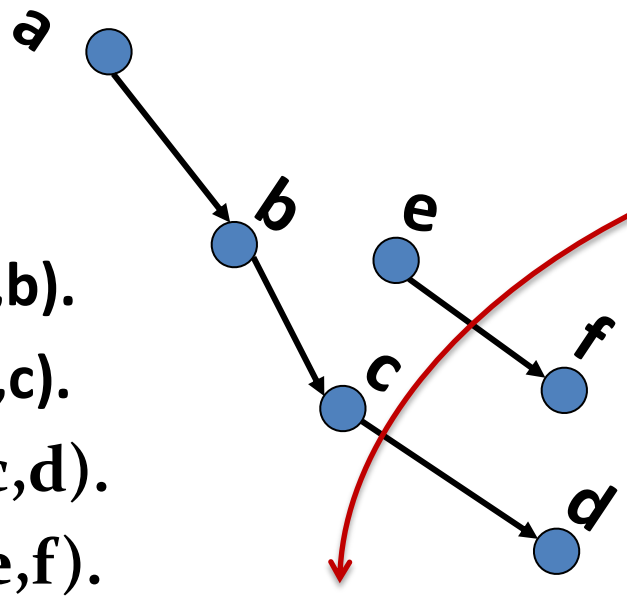
Αναδρομική μεταβατική σχέση «έμμεσων» ακμών «μονής κατεύθυνσης».

```
connected(X,Y):- road(X,Y).
connected(X,Y):- road(X,Z), connected(Z,Y).
```

```
connected2(X,Y):- connected(X,Y).
connected2(X,Y):- connected(Y,X).
```

Αντιμεταθετική σχέση «έμμεσων» ακμών.

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (8/12)



road(a,b).

road(b,c).

road(c,d).

road(e,f).

connected2(X,Y):- connected(X,Y).

connected2(X,Y):- connected(Y,X).

connected(X,Y):- road(X,Y).

connected(X,Y):- road(X,Z), connected(Z,Y).

?- connected2(a,c).

↓

?- connected(a,c).

↓

?- road(a,Z),connected(Z,c).

↓ {Z=b}

?- connected(b,c).

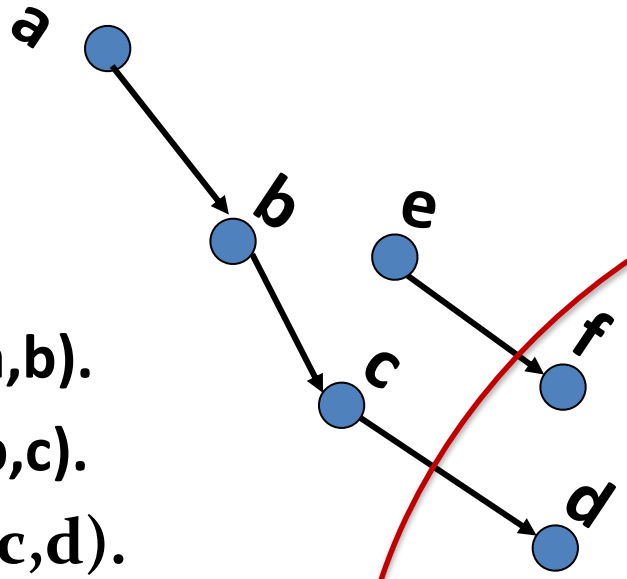
↓

?- road(b,c).

↓

□

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (9/12)



road(a,b).

road(b,c).

road(c,d).

road(e,f).

connected2(X,Y):- connected(X,Y).

connected2(X,Y):- connected(Y,X).

connected(X,Y):- road(X,Y).

connected(X,Y):- road(X,Z), connected(Z,Y).

?- connected2(c,a).

↓

?- connected(a,c).

↓

?- road(a,Z),connected(Z,c).

↓ {Z=b}

?- connected(b,c).

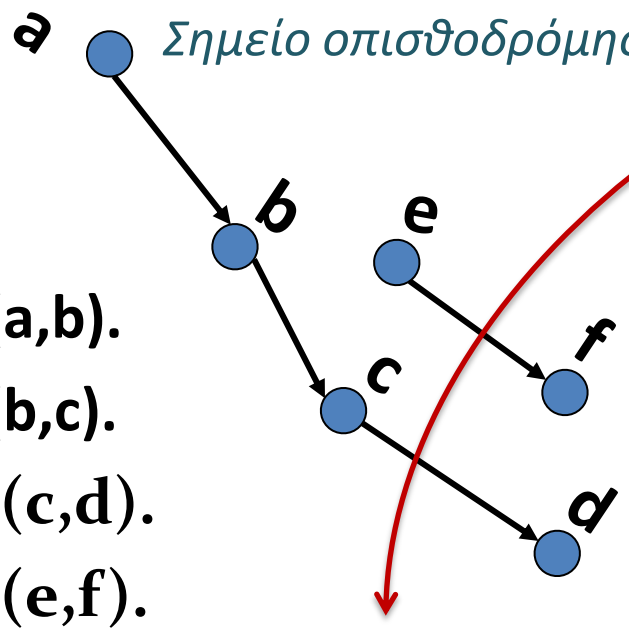
↓

?- road(b,c).

↓

□

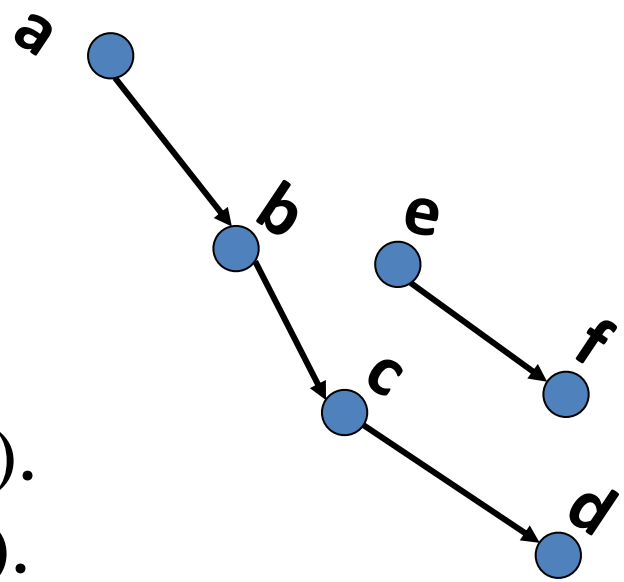
Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (10/12)




```
road(a,b).
road(b,c).
road(c,d).
road(e,f).
connected2(X,Y):- connected(X,Y).
connected2(X,Y):- connected(Y,X).
connected(X,Y):- road(X,Y).
connected(X,Y):- road(X,Z), connected(Z,Y).
```

```
?- connected2(a,e).
*
?- connected(a,e).
?- road(a,Z),connected(Z,e).
{Z=b}
?- connected(b,e).
?- road(b,Z),connected(Z,e).
{Z=c}
?- connected(c,e).
```

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (11/12)



road(a,b).

road(b,c).

road(c,d).

road(e,f).

connected2(X,Y):- connected(X,Y).

connected2(X,Y):- connected(Y,X).

connected(X,Y):- road(X,Y).

connected(X,Y):- road(X,Z), connected(Z,Y).

?- connected(c,e).

?- road(c,Z),connected(Z,e).

{Z=d}

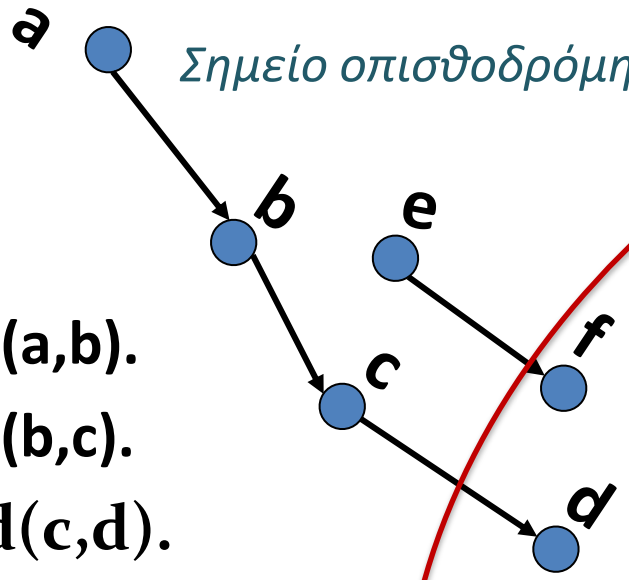
?- connected(d,e).

?- road(d,Z),connected(Z,e).

FAIL

Οπισθοδρόμηση σε προηγούμενο σημείο της εκτέλεσης με εναλλακτική επιλογή.

Συνδυασμός αντιμεταθετικής-μεταβατικής ιδιότητας (12/12)



Σημείο οπισθοδρόμησης

```
road(a,b).
road(b,c).
road(c,d).
road(e,f).
connected2(X,Y):- connected(X,Y).
connected2(X,Y):- connected(Y,X).
connected(X,Y):- road(X,Y).
connected(X,Y):- road(X,Z), connected(Z,Y).
```

?- connected2(a,e).

*

?- connected(e,a).

?- road(e,Z),connected(Z,a).

{Z=f}

?- connected(f,a).

?- road(f,Z),connected(Z,a).

FAIL

Δεν υπάρχει άλλο σημείο οπισθοδρόμησης.

no

Γράφοι και Προβλήματα Αναδρομής

- Λύση στα προβλήματα της αναδρομής σε κυκλικούς γράφους μπορεί να δοθεί μόνο αν μπορέσουμε να «θυμόμαστε» με κάποιον τρόπο από ποια σημεία έχουμε «περάσει».
- Αυτό θα γίνει με τη βοήθεια κάποιων αναδρομικών δομών δεδομένων, όπως είναι οι λίστες.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2013-2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.