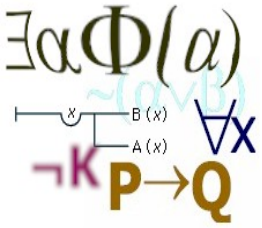
Πανεπιστήμιο Μακεδονίας
Τμ. Εφαρμοσμένης Πληροφορικής

Λογικός Προγραμματισμός με Περιορισμούς
Constraint Logic Programming



Ηλίας Σακελλαρίου

Πληροφορίες Σχετικές με το Μάθημα

- Σελίδα στο compus.
- Γραπτές Εξετάσεις στο τέλος του Εξαμήνου (70% της τελικής βαθμολογίας),
- Παράδοση εβδομαδιαίων εργαστηριακών ασκήσεων (10%)
- Δύο εργασίες (20%).

Όραμα

- Δηλωτικός Προγραμματισμός

Ο προγραμματιστής διατυπώνει το πρόβλημα και ο Η/Υ το επιλύει.

(holly grail of programming)

Θεωρητικά Θεμέλια

- Κλασική εξίσωση του Robert Kowalski (1974):

Program = Logic + Control

(πρόγραμμα= λογική + έλεγχος)

- Ο R. Kowalski, απέδειξε ότι ένα **υποσύνολο** της λογική πρώτης τάξης (προτάσεις **Horn**) μπορούν να χρησιμοποιηθεί για μια νέα γενιά γλωσσών προγραμματισμού.
- Βασίστηκε στο γεγονός της ύπαρξης μιας **αποδεικτικής διαδικασίας** για λογική Α τάξης (Αρχή της ανάλυσης – Robinson μέσα δεκαετίας 60).

Δηλωτικός vs Διαδικαστικός Προγραμματισμός.

- Στις συμβατικές γλώσσες προγραμματισμού, (C, Pascal, Fortran), το τμήμα της λογικής και το τμήμα του ελέγχου είναι αλληλένδετα.
  - Ο προγραμματιστής είναι επιφορτισμένος τις με τον καθορισμό της ροής ελέγχου του προγράμματος.
- Στις δηλωτικές γλώσσες γίνεται σαφής διαχωρισμός της λογικής και του ελέγχου.
  - Απαιτείται να περιγραφεί μόνο η λογική του προς επίλυση προβλήματος.
  - Ο έλεγχος αφήνεται στο σύστημα.

Λογικό Πρόγραμμα

- Σε ένα πρόγραμμα Λογικού Προγραμματισμού:
  - Ορίζουμε τις σχέσεις μεταξύ των οντοτήτων, χρησιμοποιώντας λογικές προτάσεις Horn.
  - Ζητάμε από το σύστημα να αποδείξει την αλήθεια άλλων προτάσεων (ερωτήσεις-queries).
- Ο υπολογιστής βάσει των προτάσεων που έχουμε περιγράψει προσπαθεί να αποδείξει την αλήθεια της ερώτησης χρησιμοποιώντας μια αποδεικτική διαδικασία της λογικής βασισμένη στον κανόνα της αρχής της ανάλυσης.

Πλεονεκτήματα

- Μερικά Πλεονεκτήματα:
  - Ευκολία στην ανάπτυξη προγραμμάτων.
  - Μικρός χρόνος ανάπτυξης και συντήρησης.
  - Βασίζεται σε ισχυρά θεωρητικά θεμέλια.
  - Διαχωρισμός εκτέλεσης προγράμματος από λογική επίλυσης.
- Κύριο Μειονέκτημα (στο παρελθόν):
  - Μειωμένη απόδοση σε σχέση με μη-δηλωτικά προγραμματιστικά παραδείγματα.

Περιορισμοί

- Είναι **λογικές σχέσεις** μεταξύ **μεταβλητών**, όπου κάθε μεταβλητή μπορεί να πάρει τιμές από ένα **συγκεκριμένο πεδίο**.
  - Περιορίζει τις πιθανές τιμές που μπορούν να πάρουν οι μεταβλητές, δηλ. εκφράζει **μερική πληροφορία** για το πρόβλημα.
  - Για παράδειγμα σε μια εφαρμογή χρονοπρογραμματισμού, αν S_A και S_B είναι οι χρόνοι έναρξης των εργασιών A και B, και D_A η διάρκεια της A, τότε ο περιορισμός

    $$S_A + D_A < S_B$$

    δηλώνει ότι η εργασία B πρέπει να γίνει μετά την A.

Χαρακτηριστικά Περιορισμών

- Οι περιορισμοί είναι:
  - **δηλωτικοί**: ορίζουν μια σχέση μεταξύ των οντοτήτων του προβλήματος χωρίς να ορίζουν μια συγκεκριμένη υπολογιστική διαδικασία.
  - **προσθετικοί**: ενδιαφέρει συνήθως η σύζευξη των περιορισμών και όχι η σειρά με την οποία τέθηκαν.
  - **σπανίως ανεξάρτητοι**: στη συνηθέστερη περίπτωση οι περιορισμοί έχουν κοινές μεταβλητές.
- Είναι ένας φυσικός τρόπος έκφρασης προβλημάτων σε ένα εξαιρετικό φάσμα πεδίων.

Προγραμματισμός με Υποστήριξη Περιορισμών

- CP (constraint programming): Μελέτη συστημάτων βασισμένων στους περιορισμούς.
- Δηλωτικό Παράδειγμα Προγραμματισμού:

"Ο προγραμματιστής δηλώνει ποιοι είναι οι περιορισμοί του προβλήματος και η πλατφόρμα προσφέρει την υποδομή για την επίλυση τους".

- Συνδυάζει αποτελέσματα από διάφορα πεδία: *τεχνητή νοημοσύνη, επιχειρησιακή έρευνα, λογική, νευρωνικά δίκτυα*, κλπ.
- Έχει αναγνωριστεί από την ACM σαν μια από τις στρατηγικές κατευθύνσεις στην έρευνα στο πεδίο των υπολογιστών.

Λογικός Προγραμματισμός με Περιορισμούς (CLP)

- Υποστήριξη Περιορισμών
  - Αντικατάσταση κλασικής διαδικασίας **ενοποίησης** με **επίλυση περιορισμών.**
  - **Δραματική Αύξηση** της απόδοσης σε συνδυαστικά προβλήματα.
- Λογικός Προγραμματισμός κατάλληλο όχημα για CP.
  - Δηλωτικότητα
  - Υποστήριξη Λογικών Σχέσεων (relations)
  - Υποστήριξη διαδικασιών αναζήτησης
  - Ισχυρά Θεωρητικά Θεμέλια
  - + όλα τα προηγούμενα πλεονεκτήματα....

Λογικός Προγραμματισμός = Prolog?

- Η **Prolog** (**PRO**gramming in **LOG**ic) είναι μια συμβολική γλώσσα προγραμματισμού.
- Κύριοτερος αντιπρόσωπος της σχολής λογικού Προγραμματισμού
- Βασίζεται στην κατηγορηματική λογική Α' Τάξης.
- Σχεδόν όλα τα σύγχρονα συστήματα Prolog, υποστηρίζουν περιορισμούς.

Ιστορική Αναδρομή

- Ξεκίνησε στη δεκαετία του '70 στην Ευρώπη (A. Colmerauer και R. Kowalski).
- Πρώτες εφαρμογές
  - Απόδειξη θεωρημάτων.
  - Επεξεργασία φυσικής γλώσσας.
- Υλοποίηση ενός μεταφραστή - διερμηνευτή της γλώσσας από τον D.H.D. Warren (1977).
  - Με automatic garbage collection, virtual machine execution, typeless variables, etc.

Εφαρμογές Prolog/CLP

- Εξαιρετική για
  - Natural Language Processing
  - Έμπειρα Συστήματα (Expert Systems), Σχεδιασμό Ενεργειών (Planning), Μαθηματικά (Mathematical Reasoning) και άλλες εφαρμογές ΤΝ.
  - Εφαρμογές χρονοπρογραμματισμού (CLP)
  - Σημασιολογικό Διαδίκτυο
  - Ότι έχει να κάνει με επεξεργασία συμβόλων.

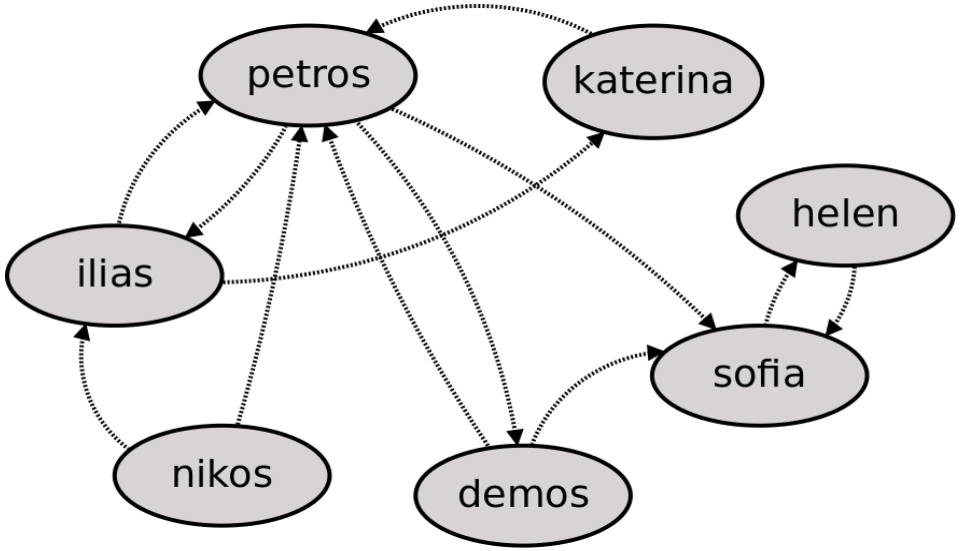
Περιεχόμενο Μαθήματος

- Θεωρητικές Βάσεις-Μαθηματική Λογική
- Προγραμματισμός σε Prolog (σύνταξη, δομές, extra-logical predicates, κλπ)
- Θεωρία Επίλυσης Περιορισμών
- Υποστήριξη περιορισμών σε συστήματα Λογικού Προγραμματισμού.

Πρόβλημα

Motivation

- Κοινωνικό Δίκτυο
- Ένας χρήστης ακολουθεί κάποιον άλλο.
  - Σχέση follows/2
- Ένας χρήστης είναι **φίλος** με καποιον άλλο, αν ο ένας ακολουθεί τον άλλο.
  - Σχέση friends/2
- Οι χρήστες έχουν φύλο.
  - male/1, female/1



Μαθηματική Λογική (σύντομη παρουσίαση)

Μαθηματική Λογική

- Συστηματική μελέτη έγκυρων ισχυρισμών (valid arguments)
- Είδη Λογικής
  - **Προτασιακή Λογική (Propositional Logic)**
  - **Κατηγορηματική Λογική (Predicate Logic)**
  - Τροπικές Λογικές (Modal Logic)
  - Λογικές Ανώτερης Τάξης (Higher Order Logics)
  - ..κλπ

Προτασιακή Λογική

- Απλούστερη μορφη της Λογικής
- Προτάσεις: αληθείς (true) / ψευδείς (false)
- Συνδετικά
  - AND ($\wedge$) **Σύξευξη**
  - OR ($\vee$) **Διάζευξη**
  - NOT ($\neg$) **Άρνηση**
  - Implication ($\Rightarrow$) **Συνεπαγωή**
  - Equivalence ($\Leftrightarrow$) **Ισοδυναμία**
- Ορθά δομημένοι τύποι = Προτάσεις + Συνδετικά

Περιγράφοντας το Κόσμο

- Ο κοσμός αποτελείται από
  - Αντικείμενα
  - Έννοιες
- Αντικείμενα και έννοιες έχουν
  - Ιδιότητες
  - Σχέσεις ανάμεσα τους
- Η προτασιακή λογική δεν μπορεί να τα περιγράψει με συμπαγή και αποδοτικό τρόπο.

Κατηγορηματική Λογική Α Τάξης

- Ένας **όρος** (term) μπορεί να είναι
  - Μια **σταθερά**, πχ: small, teacher, john, ...
  - Μια **μεταβλητή**, πχ: X, Y, Z
  - Ένας **συναρτησιακός όρος** N τάξης (N-ary functional term) $f(t_1,t_2,..,t_n)$ όπου $t_1,t_2,..,t_n$ είναι όροι. πχ: time(12,00)
- Ένα κατηγόρημα P (**Predicate)** N-τάξης
  - $P(t_1, t_2,..t_n)$ σημειώνεται με P/N.
  - $t_1..t_n$ είναι τα ορίσματα και είναι όροι,
  - Περιγράφει **μια σχέση** ανάμεσα σε αντικείμενα του κόσμου.
- Οι μεταβλητές μπορούν να πάρουν σαν τιμή οποιοδήποτε όρο.

Παράδειγμα

- Ιδιότητες Αντικειμένων

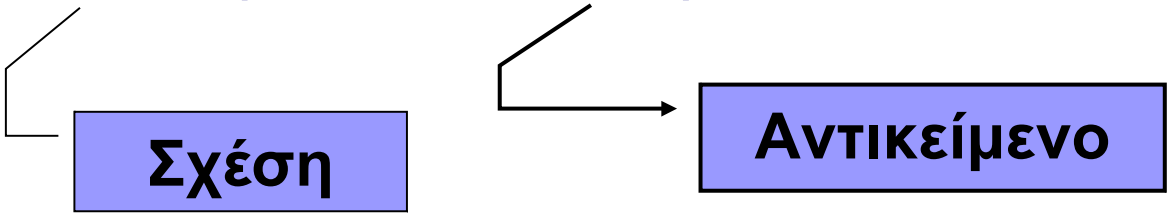
Ο petros είναι άρρεν

male(petros)←

- Σχέσεις μεταξύ Αντικειμένων

Ο petros ακολουθεί τον ilias

follows (petros, ilias)←



Μεταβλητές

- Αυξάνουν την εκφραστικότητα της Λογικής
  - ο petros ακολουθεί κάποιον

    follows(petros, X) $\quad \exists$ ***X:follows(petros,X)***
  - ο πετρος ακολουθει τους πάντες

    follows (petros, X) $\quad \forall$ ***X:follows(petros,X)***

- **Ποσοδείκτες**
  - Υπαρξιακός Ποσοδείκτης: ∃
  - Καθολικός Ποσοδείκτης: ∀

Σύνθετες Σχέσεις

- Ισχύουν τα κλασικά συνδετικά (σύζευξη, διάζευξη, άρνηση, κλπ).
- Ορθά σχηματισμένοι τύποι
  - Κατηγορήματα + Συνδετικά + Ποσοδείκτες
- Παράδειγμα

  Οι x και y είναι φίλοι, αν ο ένας ακολουθεί τον άλλο.

  $\forall x,y: friends(x,y) \leftarrow follows(x,y) \wedge follows(y,x)$

Αποδεικτικές Διαδικασίες

- Διαπίστωση αν **ένας ορθά δομημένος τύπος**, **συνεπάγεται λογικά** από ένα σύνολο τύπων.
- **Απόδειξη**: ακολουθία βημάτων καθένα από τα οποία είναι η εφαρμογή ενός κανόνα συμπερασμού στους τύπους, που δημιουργεί νέους τύπους.
- **Σημασία της αρχής της ανάλυσης**: Ένας και μοναδικός κανόνας είναι απαραίτητος για να αποδείξει τα πάντα.

Prolog

μια σύντομη παρουσίαση...

Prolog

- Βασίζεται στην κατηγορηματική λογική πρώτης τάξης.
  - Σύνταξη της γλώσσας ακολουθεί εκείνη της λογικής.
- Αποδεικτική διαδικασία: αρχή της ανάλυσης.
- Διερμηνευόμενη γλώσσα (αν και υπάρχει πλήθος compiled εκδόσεων).
- Απλή στη σύνταξη, συμπαγής κώδικας, εύκολη συντήρηση, κλπ.

Σύνταξη Prolog (1/2)

- Όλες οι **σταθερές** και τα **κατηγορήματα** ξεκινούν **με πεζό γράμμα.**
- Όλες οι **μεταβλητές** ξεκινούν με **κεφαλαίο γράμμα**.
- Κάθε **πρόταση** τελειώνει με τελεία ‘.’
- Το σύμβολο ← αντικαθίσταται από άνω κάτω τελεία και παύλα ‘**:-**’ (**colon** and a **dash)**.

Σύνταξη Prolog (2/2)

- Το συνδετικό ΚΑΙ (*σύξευξη*) (**AND**) αντικαθίσταται από το κόμμα ‘,’.
- Το συνδετικό Η (*διάζευξη*) (**OR)** αντικαθίσταται από το ελλ. ερωτηματικό ';'.
- Δεν υπάρχουν ποσοδείκτες!
  - Οι μεταβλητές στο αριστερό μέλος του κανόνα θεωρούνται καθολικά ποσοτικοποιημένες, ενώ στο δεξιό υπαρξιακά ποσοτικοποιημένες.

Από την λογική στη Prolog

- Σύνταξη προτάσεων **Horn** στη Λογική

  male(petros)←

  follows(petros, ilias)←

  $\forall x,y$: *friend(x,y)* ← *follows(x,y)* $\wedge$ *follows(y,x)*

- Σύνταξη Prolog

  male(petros).

  follows(petros, ilias).

  friend(X,Y) :- follows(X,Y), follows(Y,X).

Τυπικοί Ορισμοί

- Prolog πρόγραμμα = σύνολο από προτάσεις **Horn**:
  - H:-B_1,B_2,..B_n.
  - όπου H, B_1,B_2,..B_n είναι κατηγορήματα.
- Γενονός (fact) ή μοναδιαία πρόταση (unit clause) H.
- Κανόνας (rule) H:-B_1,B_2,..B_n.
- Ερώτηση ή πρόταση στόχος (Query, Goal Clause) :-B_1,B_2,..B_n

Εκτελώντας ένα Prolog Πρόγραμμα

- Εκτέλεση Οδηγούμενη από ερωτήματα
- Παράδειγμα

?-male(petros).

yes

?-follows (petros, ilias).

yes

Πρόγραμμα

```
male( petros ).
follows(ilias, petros).
follows(petros,ilias).
friends(X,Y):-
        follows(X,Y),
        follows(Y,X).
```

Άρνηση σαν Αποτυχία

- Όχι από ότι γνωρίζω (“Not as far as I know”)
  - Υπόθεση Κλειστού Κόσμου

?- male(mary).

no

?-male(gregory).

no

?- follows(nick, mary).

no

Πρόγραμμα

```
male( petros ).
follows(ilias, petros).
follows(petros,ilias).
friends(X,Y):-
        follows(X,Y),
        follows(Y,X).
```

Πολλαπλές (Εναλλακτικές) Απαντήσεις

- Η Prolog επιστρέφει όλες τις διαθέσιμες απαντήσεις
  - Νο σημαίνει “no more answers”

?-male(X).

X = petros ;

X = ilias ;

X = demos ;

no

Πρόγραμμα

male(petros).

male(ilias).

male(demos).

Ερμηνεία των Prolog Προγραμμάτων

$$H{:-}B_1,B_2,..B_n.$$

- Δηλωτική Ερμηνεία
  - Το Η είναι αληθές αν B_1 είναι αληθές και B_2 είναι αληθές και B_n είναι αληθές.
- Διαδικαστική Ερμηνεία
  - Για να αποδειχθεί η αλήθεια του Η πρέπει να απόδείξουμε το B_1 και έπειτα το B_2 και έπειτα το B_n.

Πλατφόρμες Λογικού Προγραμματισμού

- Πλατφόρμες Λογικού Προγραμματισμού με περιορισμούς:
  - ECLiPSe Prolog
  - SWI Prolog
  - YAP Prolog
  - SICStus Prolog (εμπορική)
- Πλέον τα περισσότερα “παιδικά” προβλήματα των υλοποιήσεων έχουν επιλυθεί.

Σύνοψη

- Μέχρι τώρα
  - Δηλωτικός Προγραμματισμός
  - Μαθηματική Λογική σαν αφαίρεση για την δημιουργία νέας σχολής προγραμματισμού
  - Prolog (σύντομη εισαγωγή).
- Συνέχεια με
  - Σύνταξη της Prolog

Σύνταξη

- Prolog όροι
- Γεγονότα και Κανόνες
- Σύνθετοι όροι και Ενοποίηση

Prolog Όροι

Όροι της Prolog (Terms)

- Prolog Terms
  - **Σταθερές - άτομα ή αριθμοί** (Atoms or Numbers)
  - **Μεταβλητές** (Variables)
  - **Σύνθετοι όροι** (Compound Terms)

Άτομα (Atoms)

- Τα **άτομα** (atoms) είναι συμβολοσειρές γραμμάτων και ψηφίων που ξεκινούν από **πεζό γράμμα** και μπορεί να περιλαμβάνουν το underscore:

  πχ. sunday_morning, theSun, more_Days12

- Οποιαδήποτε συμβολοσειρά σε μονά εισαγωγικά

  πχ. ‘Sunday Morning’, ‘SUN’, ‘s o s’

- Οι χαρακτήρες **[], {}, ; , !**
- Οποιαδήποτε συμβολοσειρά που περιλαμβάνει αποκλειστικά τα: **+, -, *, /, \, ^, <, >, =, : , . ,?,@,#,$,&**

  πχ. ==>, \#=, >>>, --->, &=>

Έλεγχος Ατόμων

- Ενσωματωμένο κατηγόρημα **atom/1.**

?- atom(foo).

yes

?- atom('Foo').

yes

?- atom(==>)

yes

?- atom(_asd).

no

?- atom(12xa)

<error>

Αριθμοί

- Υποστηρίζονται οι συνήθεις μορφές αριθμών
  - Οι υποστήριξη αριθμών εξαρτάται και από την υλοποίηση της γλώσσας.
- Ακέραιοι

  7, 23, 45, 19
- Μεταβλητής υποδιαστολής

  7.2, 3.4, 89.9

Μεταβλητές

- Μια συμβολοσειρά (χαρακτήρες + ψηφία) που ξεκινά είτε:
  - Με ένα κεφαλαίο χαρακτήρα

    X, Father, FileName, File_name, File_Name
  - Το underscore “_” (Ανώνυμες μεταβλητές)

    _, _Mother, _File
- Μπορούν να πάρουν σαν τιμή **οποιονδήποτε** όρο.

Ανώνυμες Μεταβλητές

- Θέσεις σε μια πρόταση για τις τιμές των οποίων δεν ενδιαφερόμαστε.

  city(athens, south, 6).

  city(thessaloniki, north, 2).

  city(kavala, north, 0.5).

  - **Ποια πόλη είναι στο Βορά:** *?- city(City, north, _).*
  - **Δώστε το όνομα μιας πόλης:** *?- city(City, _ , _).*
- Και γιατί δεν βάζουμε απλώς ένα όνομα;
  - Όλες οι υλοποιήσεις της Prolog προσφέρουν ένα μηχανισμό εντοπισμού **singleton** μεταβλητών για αποφυγή λαθών.

Διαφορές από κλασσικές Γλώσσες

- **ΔΕΝ έχουν τύπο** (typeless language) ΔΕΝ απαιτούν δηλώσεις.
- **Μεταβλητές Μοναδικής Ανάθεσης** (single assignment)
  - Κλασσικές γλώσσες → καταστροφική ανάθεση (destructive assignment).
  - Prolog → σε μια μεταβλητή που έχει πάρει τιμή δεν μπορεί να δοθεί νέα (non destructive assignment / single assignment).
- **Τοπικές μεταβλητές**
  - Εμβέλεια μέσα στον κανόνα που εμφανίζονται.

Γεγονότα και Κανόνες

Λογικό Πρόγραμμα

- Γεγονότα
  - Ορισμοί κατηγορημάτων που είναι πάντα αληθείς.

    male(petros).

    follows (ilias, petros).
- Κανόνες
  - Κατηγορήματα τα οποία είναι αληθή υπό συνθήκες.

    follows_male(X,Y):- follows(X,Y), male(Y).

Στοιχεία ενός Prolog Προγράμματος

`A.` *Γεγονός* (*fact*)

`A :- B`$_1$`, B`$_2$`, . . ., B`$_\kappa$`.` *Κανόνας* (*rule*) (`κ > 0`)

Κεφαλή (head) **Σώμα (body)**

- Τα `A` και `B`$_i$ ονομάζονται ατομικοί τύποι και είναι παραστάσεις της μορφής: $p(t_1, t_2, \ldots, t_n)$
  - p ονομάζεται **κατηγόρημα** (predicate), και τα
  - t_i ονομάζονται **ορίσματα** (arguments).

Ορισμός ενός Κατηγόρηματος

- Ένα κατηγόρημα χαρακτηρίζεται από
  - το **όνομά** του (predicate name)
  - την **τάξη** του (arity), δηλ. τον αριθμό των ορισμάτων του (arguments).
  - Χρησιμοποιείται το **name/arity** για να αναφερθούμε στο κατηγόρημα.
  - Ένα κατηγόρημα μπορεί να αποτελείται από **συνδυασμό** γεγονότων και κανόνων.

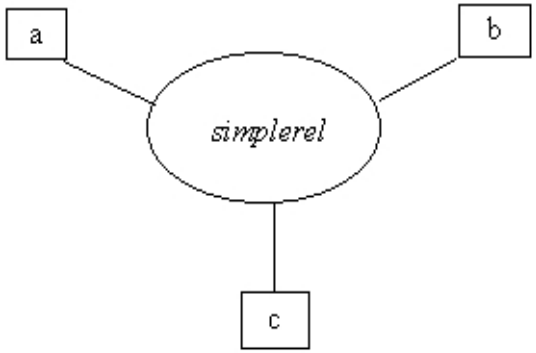
father(john,mary).

father(nick,mary).

father(Father,Child):- parent(Father,Child),male(Father).

Γεγονότα

simplerel(a,b,c).



- Απλές ιδιότητες
  - black(table).
  - red(chair).
- Απλές σχέσεις ανάμεσα σε αντικείμενα
  - room(dinning, first_floor).
  - room(bedroom, second_floor).

Ορίσματα

- Η σειρά των ορισμάτων είναι θέμα σχεδίασης

  color(chair, blue).

  color(blue, chair).

- Δεν υπάρχει η κλασική έννοια ορισμάτων **εισόδου** και **εξόδου** (**INPUT & OUPUT).**

  ?- color(chair, X). *X = blue*

  ?- color(X, blue). *X = chair*

  ?-color(X, Y). *X = chair, Y = blue*

Εναλλακτικές Προτάσεις

- Διαφορετικές προτάσεις του **ιδίου κατηγορήματος** αποτελούν εναλλακτικές.

vehicle(car).

vehicle(truck).

vehicle(bike).

vehicle(forklifter).

- Αρκεί μόνο μια για να δοθεί μια λύση.

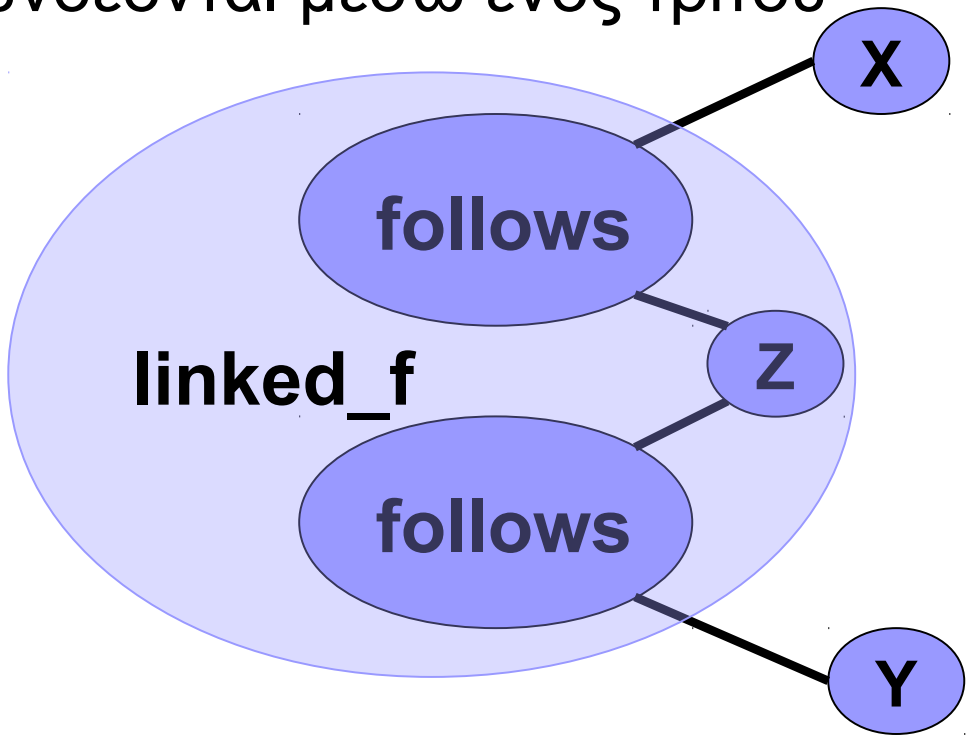
Κανόνες

- Αναπαριστούν πολύπλοκες σχέσεις που προκύπτουν από την σύνθεση άλλων σχέσεων.
  - linked_f/w: X,Y συνδέονται μέσω ενός τρίτου χρήστη.

linked_f(X, Y):-

follows(X, Z),

follows(Z, Y).



Μεταβλητές σε μια πρόταση (1)

- Εμβέλεια των μεταβλητών είναι μέσα στη **πρόταση** που εμφανίζονται.

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

- Οι μεταβλητές X,Y,Z δεν είναι οι ίδιες!

Μεταβλητές σε μια πρόταση (1)

- Εμβέλεια των μεταβλητών είναι μέσα στη **πρόταση** που εμφανίζονται.

```
grandfather( X,Y ):-
    father( X, Z ), parent( Z,Y ).
grandfather( GF, GC ):-
    parent( GF,P), male( GF ), parent( P,GC).
```

- Οι μεταβλητές X,Y,Z δεν είναι οι ίδιες!

Μεταβλητές σε μια πρόταση (2)

- Μέσα στην ίδια πρόταση αναφέρονται στο ίδιο αντικείμενο.
- Όπως είπαμε, ανατίθεται σε αυτές τιμή ΜΟΝΟ μια φορά.

X = X + 1.

Not in Prolog!

Προφανώς δεν ισχύει στη Λογική!

Διαζεύξεις

- Ο τελεστής “;” δηλώνει διάζευξη.

  has_some_link(X) :- follows(X, _) ;

  follows(_,X).

- Προφανώς το παραπάνω μπορεί να γραφεί και σαν:

  has_some_link(X) :- follows(X, _).

  has_some_link(X) :- follows(_, X).

Προτεραιότητα Τελεστών AND/OR

- Ο τελεστής σύζευξης έχει μεγαλύτερη προτεραιότητα από εκείνον της διάζευξης.

  P:-Q, R; S, T, U.

  είναι ισοδύναμο με *P:- (Q,R);(S, T, U).*

  και προφανώς το ίδιο με το ακόλουθο:

  P:- Q,R.

  P:- S,T,U.

Programming Tips

Σχόλια

- Ξεκινούν με το σύμβολο **%**
- Περιλαμβάνονται στα */* ... **/*
- Τυπικός τρόπος σχολιασμού κατηγορημάτων.

```
%%% follows_male/2
%%% follows_male(X,Y)
%%% Succeeds  if X follows a user Y, that is male.
follows_male(X,Y):-
    follows(X,Y),
    male(Y).
```

Κανόνες καλής πρακτικής

- ΠΡΟΣΟΧΗ ΣΤΑ ΟΝΟΜΑΤΑ
- Ονόματα κατηγορημάτων
  - employed(ibm, john).
- Ονόματα μεταβλητών
  - mother(Mother, Child).
- Ονόματα αρχείων
  - *.pl ή *.ecl (για την δική σας ευκολία).

Programming Style

- Προσέγγιση Top Down
  - Ξεκινούμε με την κωδικοποίηση των “ανώτερων” σχέσεων.
- Προσέγγιση Bottom Up
  - Ξεκινούμε με την κωδικοποίηση απλών σχέσεων.
- Συνήθως η τεχνική που χρησιμοποιούμε περιλαμβάνει και τις δύο προσεγγίσεις
- ΑΛΛΑ...

ΑΝΑΠΑΡΑΣΤΑΣΗ!

- Η επιλογή της κατάλληλης αναπαράστασης είναι ζωτικής σημασίας.
- Έστω ότι οι χρήστες ανήκουν σε κάποιες ομάδες.

a(ilias).

a(petros).

b(ilias).

c(petros).

ΑΝΑΠΑΡΑΣΤΑΣΗ!

- Ποιοι χρήστες ανήκουν στην ομάδα α?

  ?-a(User).

- Σε ποιες ομάδες ανήκει ο χρήστης ilias

  ?- ...

Σωστή Αναπαράσταση

- belongs_to(Group,User).

  belongs_to(a,ilias).

  belongs_to(a,petros).

  belongs_to(b,ilias).

  belongs_to(c,petros).

- Σε ποιες ομάδες ανήκει ο χρήστης ilias?

  ?-belongs_to(Group,ilias).

... και ακόμη

- Ποιοι δύο χρήστες ανήκουν στην ίδια ομάδα?

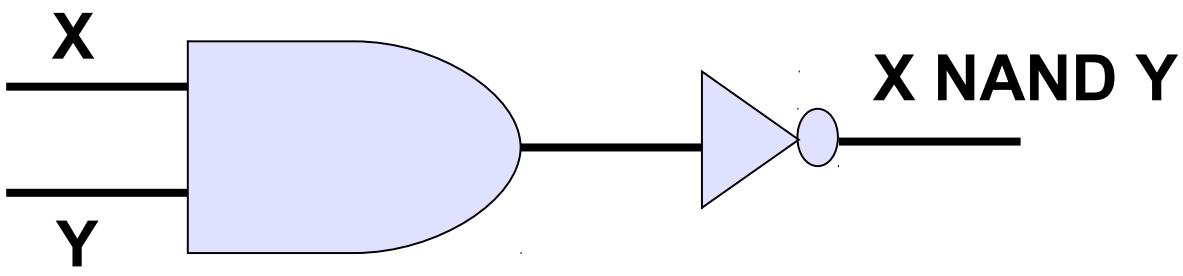
```
same_group(X,Y):-
    belongs_to(Group,X),
    belongs_to(Group,Y).
```

Hands on!

- Κατεβάστε από το compus το αρχείο **friends.ecl** .
- Δείτε τα κατηγορήματα που υπάρχουν.
- Δημιουργήστε ένα κατηγόρημα **in_same_group(Group,User1,User2)** το οποίο πετυχαίνει όταν ο User1 και ο User 2 ανήκουν και οι δύο στο Group.
- Δώστε την ερώτηση
  ?- in_same_group(Group,ilias,X).
  - Τι παρατηρείτε?

Ψηφιακά κυκλώματα

- Πύλη NAND
  - Συνδυασμός πυλών AND και NOT



Πίνακες Αλήθειας

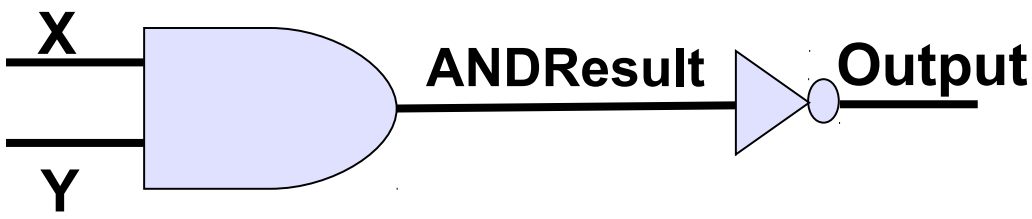
X	Y	NOT X	X AND Y
1	1	0	1
1	0	0	0
0	1	1	0
0	0	1	0

Κώδικας

```
%%% and/3
%%% Definition of the and operation
    and(0,1,0).
    and(0,0,0).
    and(1,0,0).
    and(1,1,1).

%%% not/2
%%% definition of the not operation
    not(1,0).
    not(0,1).

%%% nand/3
nand(X,Y,Output):-
    and(X,Y,ANDResult),
    not(ANDResult,Output).
```



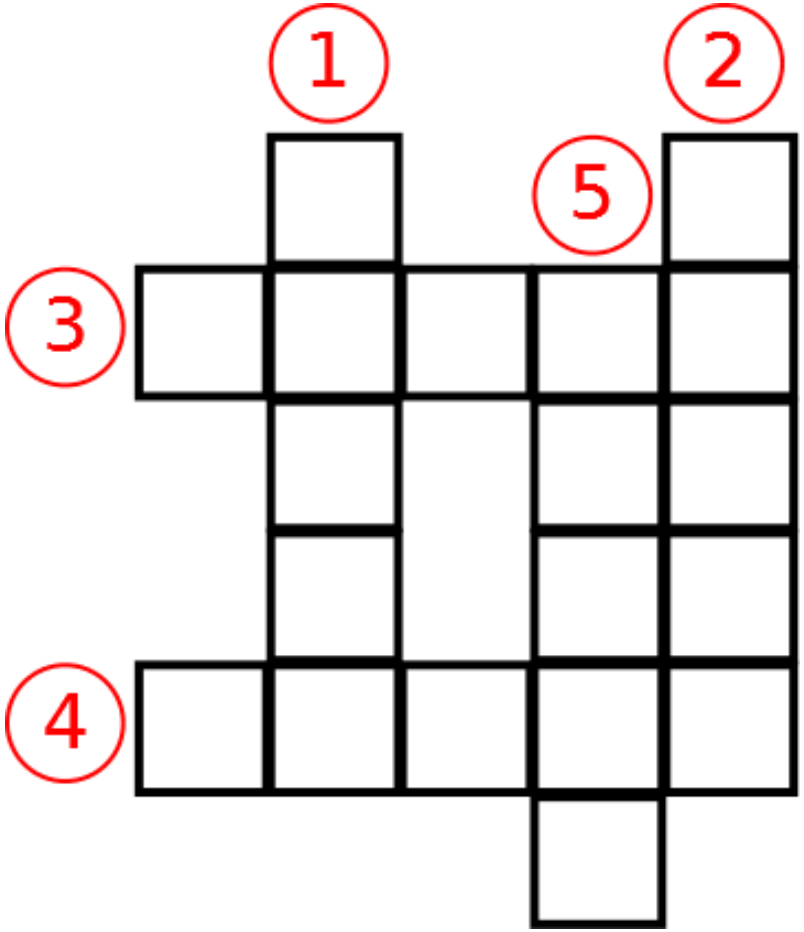
Παράδειγμα: Crossword Puzzle

- Έξι λέξεις της Αγγλικής:

  jumps, *jumbo*,

  jocks, *isles*, *babel*

- Βάλτε τις λέξεις στο διπλανό σταυρόλεξο.



Αναπαράσταση Λέξεων

word(jumbo,j,u,m,b,o).

word(jocks,j,o,c,k,s).

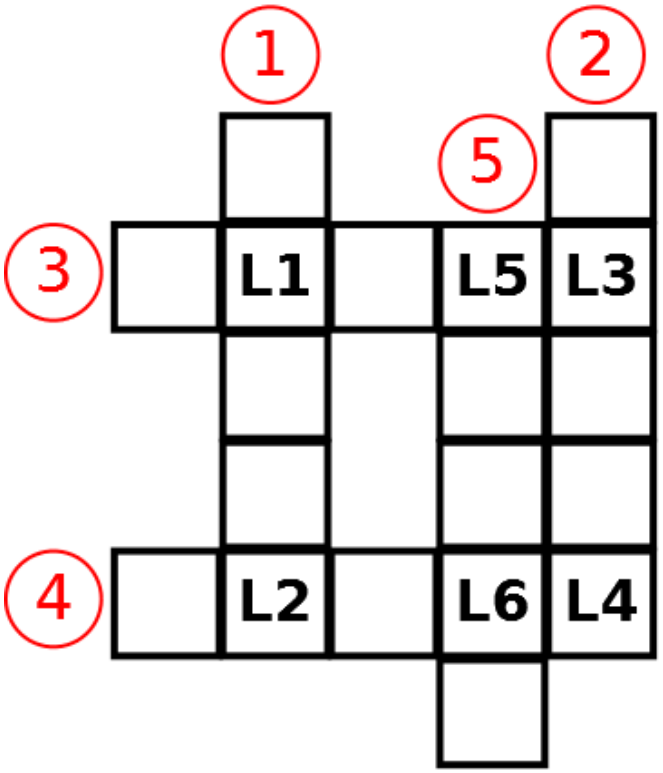
word(isles,i,s,l,e,s).

word(jumps,j,u,m,p,s).

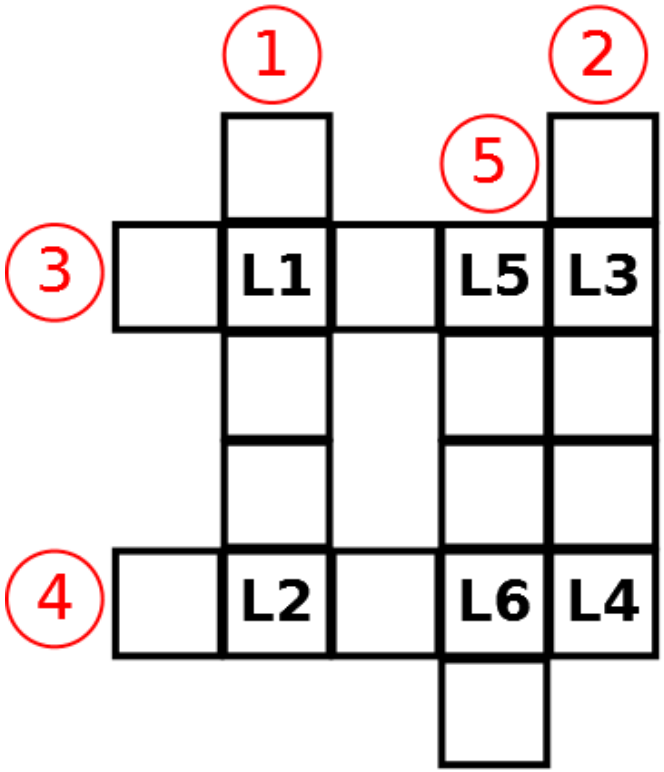
word(babel,b,a,b,e,l).

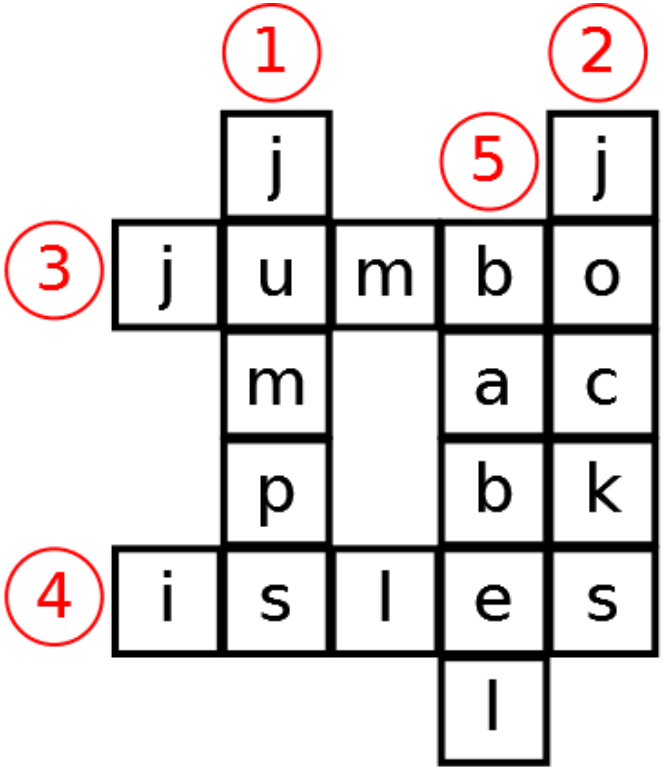
Περιορισμοί του Σταυρόλεξου

```
crossword(W1,W2,W3,W4,W5):-
  word(W1,_,L1,_,_,L2),
  word(W2,_,L3,_,_,L4),
  word(W3,_,L1,_,L5,L3),
  word(W4,_,L2,_,L6,L4),
  word(W5,L5,_,_,L6,_).
```



Περιορισμοί και Τελική Θέση





Σύνθετοι Όροι και Ενοποίηση

Σύνθετοι Όροι (Compound Terms)

- Μορφή $f(t_1, t_2, \ldots t_n)$.
  - f : Συναρτησιακό σύμβολο (**Functor)**
  - Αριθμός ορισμάτων: Τάξη (**Arity)**
- Τα ορίσματα είναι επίσης όροι.

  date(monday, 13, october)

  car(bmw, model(316), year(1981))
- Αναφέρονται και ως δομές **(structured objects).**

Κατηγορήματα & Σύνθετοι Όροι

- Οι σύνθετοι όροι και τα κατηγορήματα έχουν ίδια μορφή, ΑΛΛΑ:
  - Τα κατηγορήματα παίρνουν μια τιμή αλήθειας.
  - ΟΙ σύνθετοι όροι αναπαριστούν αντικείμενα (αποτελούν δηλαδή data).
  - Οι σύνθετοι όροι εμφανίζονται σαν ορίσματα στα κατηγορήματα.
- Αυτή η “ομοιομορφία” θα είναι ιδιαίτερα χρήσιμη κατά την **μεταβλητή κλήση**.

Παράδειγμα

- Κατηγόρημα *employee/2*

  ...

  employee(ibm, name(john)).

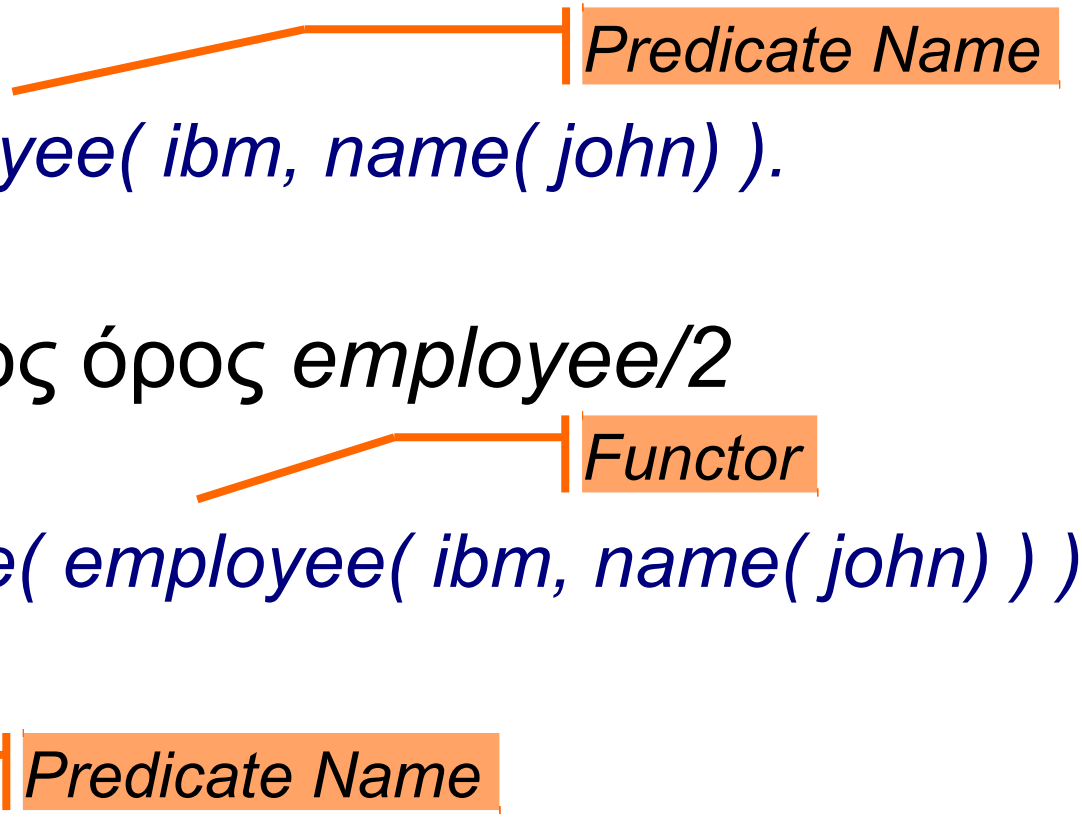
  ...

- Σύνθετος όρος *employee/2*

  ...

  people(employee(ibm, name(john))).

  ...



Συναρτησιακά Σύμβολα

- Κάθε άτομο μπορεί να αποτελέσει συναρτησιακό σύμβολο.

  ***date**(12,3)* ***month**(july, 2002)*

  ***+**(3, 4)* *3 **+** 4* *(infix notation, τελεστές!)*

  (3,+**(2,4))* *3*(2+4)*

- Οι σύνθετοι όροι δεν παίρνουν τιμή!

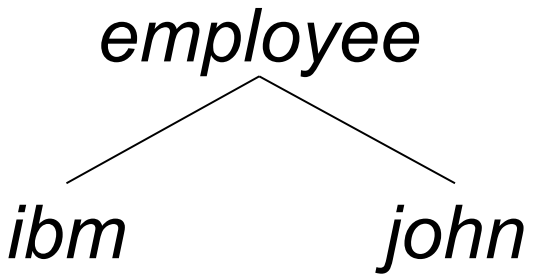
  total_work(3+5).

  total_work(+(3,5)).

Σύνθετοι όροι ως δένδρα

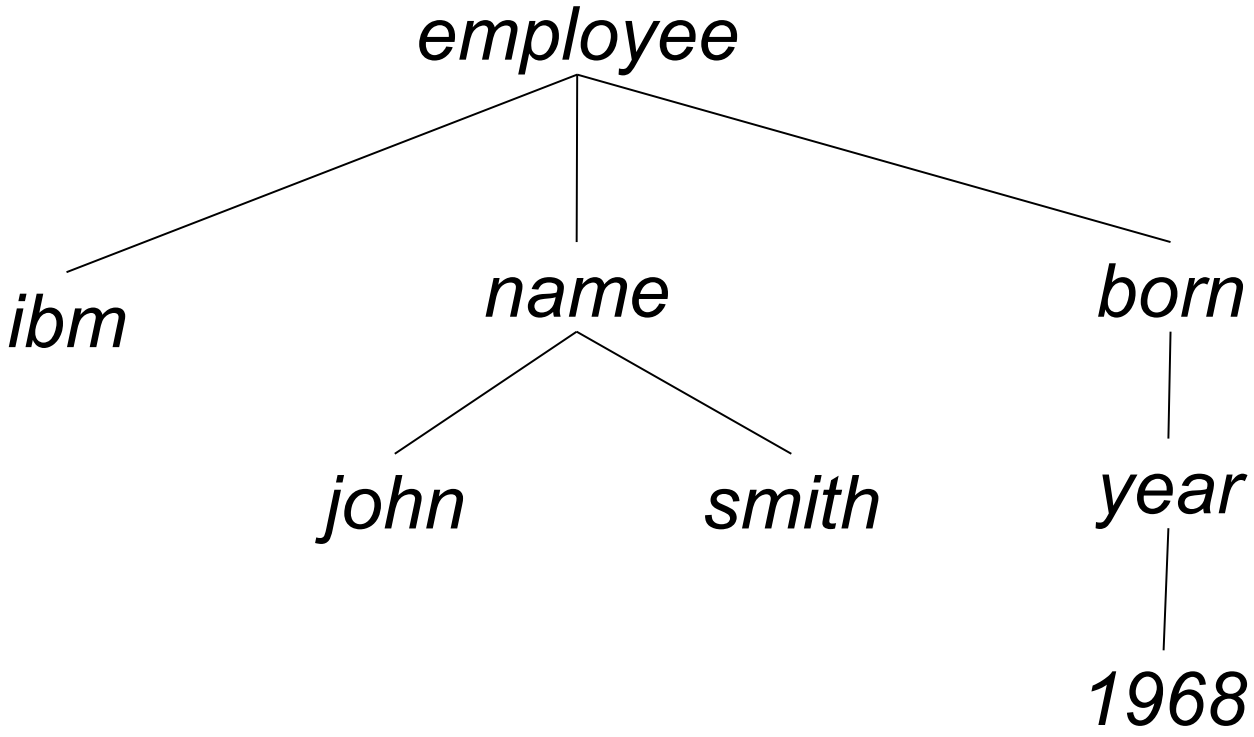
- Το συναρτησιακό σύμβολο είναι η ρίζα του δένδρου.
- Τα ορίσματα είναι τα παιδιά του.

employee(ibm, john)



Παράδειγμα

employee(ibm, name(john, smith), born(year(1968))



Παράδειγμα με μεταβλητές

- Μεταβλητές σαν ορίσματα σε σύνθετους όρους.

employee(Company, Name, Date)

employee(ibm, Name, Date)

employee(ibm, name(john, smith), Date)

employee(ibm, name(john, smith), born(1971))

Ενοποίηση (unification)

- Η πλέον σημαντική διαδικασία για όρους.
- Μέσω της ενοποίησης δύο όροι γίνονται συντακτικά όμοιοι.
  - Με κατάλληλες αναθέσεις μεταβλητών.
- Μέρος του μηχανισμού εκτέλεσης της Prolog
- Μπορεί να κληθεί ρητά από τον “=“ operator.

?- p(1,2) = p(X, Y)

X =1, Y=2

yes

Αλγόριθμος Ενοποίησης

Δοθέντων δύο εκφράσεων T1 και T2

1. Εάν T1 ή T2 είναι μεταβλητή τότε επέστρεψε επιτυχία.
2. Εάν T1 και T2 είναι η ίδια σταθερά τότε επέστρεψε επιτυχία.
3. Εάν T1 και T2 είναι σύνθετοι όροι, επέστρεψε επιτυχία:
   - Εάν έχουν το ίδιο συναρτησιακό σύμβολο και τάξη, ΚΑΙ
   - κάθε όρισμα του T1 ενοποιείται με το αντίστοιχο όρισμα του T2.
4. Εάν T1 και T2 είναι ατομικοί τύποι, επέστρεψε επιτυχία:
   - Εάν έχουν το ίδιο κατηγόρημα και τάξη, ΚΑΙ
   - κάθε όρισμα του T1 ενοποιείται με το αντίστοιχο όρισμα του T2.

Παράδειγμα ενοποίησης

employee(ibm, Name) = employee(ibm, john)

employee(ibm, Name) = employee(ibm, john)

employee(ibm, Name) = employee(ibm, john)

employee(ibm, Name) = employee(ibm, john)

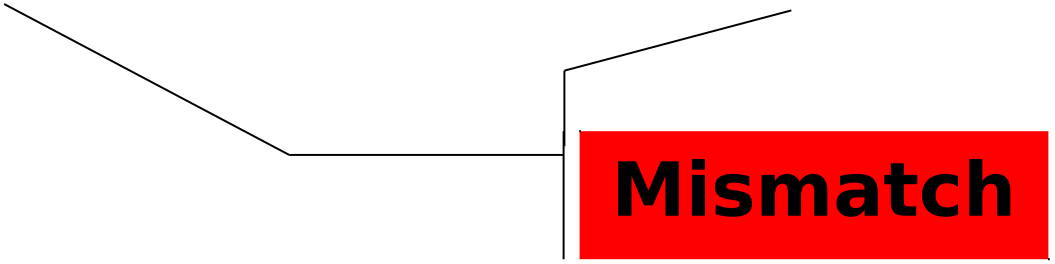
yes, {Name = john}

Παράδειγμα μη ενοποίησης

employee(ibm, Name) = employee(dell, john)

employee(ibm, Name) = employee(dell, john)

employee(ibm, Name) = employee(dell, john)



no

Τυπικοί Ορισμοί

- Αντικατάσταση (substitution) θ
  - $\theta = \{v_1=t_1, v_2=t_2 \ldots v_n=t_n\}$
- Ανάθεση (binding)
  - $v_1=t_1$
- Στιγμιότυπο ενός όρου $T\theta$
  - *T = name(Surname) and θ = {Surname=smith}*
    Tθ = name(smith)

Τυπικοί Ορισμοί (συν)

- **Ενοποιητής** (Unifier) των T_1, T_2 είναι μια **αντικατάσταση** θ τέτοια ώστε τα $T_1\ \theta$ *και* $T_2\ \theta$ *είναι συντακτικά όμοια.*

  T1 : building(College, stein)

  T2 : building(city, Name)

  θ = *{ College = city, Name=stein}*

- Ο πιο γενικός ενοποιητής θ
  - Για κάθε αλλό ενοποιητή σ ισχύει $\sigma = \theta\gamma$

Τελεστής Μη-Ενοποίησης \=

- Τελεστής μη-ενοποίησης. Τα δύο ορίσματα ΔΕΝ είναι ενοποιήσιμα.
  - ?- john \= nick.

    Yes
  - ?- john \= john.

    No
  - ?- john \= X.

    No

Παράδειγμα με σύνθετους όρους

- Για κάθε χρήστη έχουμε το profile του:
- user(petros,
  info(greek,
  birthday(12,1,1980),
  job(director)
)
).

Κοινωνικό Δίκτυο

- Δύο χρήστες με ίδια ηλικία (ίδια χρονιά γέννησης).
- same_age(User1,User2):-

  user(User1,info(_,BDay1,_)),

  user(User2,info(_,BDay2,_)),

  same_birth_year(BDay1,BDay2),

  User1 \= User2.

Ενοποίηση όρων και Μεταβλητές

- Ίδια χρονιά.
- same_birth_year(birthday(_,_,**Year**),
  birthday(_,_,**Year**)).

Hands on!

- Υλοποιείστε το κατηγόρημα **birthday_same_day(Date,User1,User2)**, το οποίο επιτυγχάνει όταν οι User1 και User2 έχουν γενέθλια την **ίδια ημερομηνία**.
- ?- birthday_same_day(Date, U1, U2).

  Date = on(1, 12)

  U1 = nikos

  U2 = sofia

  Yes

Σύνοψη

- Μέχρι τώρα
  - Σύνταξη της Prolog
    - Ατομικοί όροι και μεταβλητές
    - Γενόνοτα και κανόνες
    - Σύνθετοι όροι και ενοποίηση
- Επόμενα
  - **Εκτέλεση Prolog προγραμμάτων**