Εργαστήριο Λογικού Προγραμματισμού

Μανόλης Μαρακάκης, Καθηγητής
mmarak@cs.hmu.gr

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Σχολή Μηχανικών
Ελληνικό Μεσογειακό Πανεπιστήμιο

Ενότητα 4: Μάθημα 8

Δέντρο Αναζήτησης Οπισθοδρόμηση και Άρνηση

4. Δέντρο Αναζήτησης, Οπισθοδρόμηση και Άρνηση

- 4.1. Δέντρο Αναζήτησης
- 4.2. Οπισθοδρόμηση και Αποκοπή (!)
  - 4.2.1. Παραδείγματα με Αποκοπή
- 4.3. Το κατηγόρημα fail
- 4.4. Άρνηση σε Prolog

4. Δέντρο Αναζήτησης, Οπισθοδρόμηση και Άρνηση

4.3. Το κατηγόρημα fail

- Η Prolog περιέχει ως προδεδηλωμένο κατηγόρημα το **fail/0** το οποίο
  - **αποτυγχάνει αυτόματα με αποτέλεσμα να δημιουργεί οπισθοδρόμηση**.
- Το fail είναι χρήσιμο στις εξής δύο περιπτώσεις:
- Η **πρώτη χρήση** του fail είναι στις περιπτώσεις που επιθυμούμε **κάποιος στόχος** να **οπισθοδρομήσει μέσω όλων των δυνατών λύσεων**.
  - Αυτό γίνεται όταν θέλουμε να πάρουμε όλες τις δυνατές λύσεις.

4.3. Το κατηγόρημα fail

- Η **δεύτερη χρήση του fail είναι ο συνδυασμός του με την αποκοπή**, όπως στην επόμενη πρόταση.

  p(..) :- .. !, fail.

- Αυτός ο συνδυασμός των !, και fail εκφράζει το εξής:
  - ***Εάν η εκτέλεση έφτασε μέχρι αυτό το σημείο, τότε σταμάτα τις προσπάθειες για ικανοποίηση αυτού του στόχου.***
- Η **σύζευξη των στόχων** αποτυγχάνει λόγω του ***fail***.
- Ενώ ο πατρικός στόχος **«?- p(..),…»** αποτυγχάνει λόγω της ***αποκοπής (!)*** η οποία δεν επιτρέπει οπισθοδρόμηση. Δηλαδή, δεν επιτρέπει να προσπαθήσει την ικανοποίηση του με άλλες προτάσεις που έχουν κεφαλή το ίδιο κατηγόρημα, «**p(..) :- …**».

4. Δέντρο Αναζήτησης, Οπισθοδρόμηση και Άρνηση

4.3. Το κατηγόρημα fail

- Ο συνδυασμός ! και fail είναι ένας περιορισμένος τρόπος υλοποίησης άρνησης σε Prolog η οποία ονομάζεται ***άρνηση ως αποτυχία*** *(negation as failure)*.

4.3. Το κατηγόρημα fail

- ***Παράδειγμα 1:*** ***Η πρώτη χρήση του fail***. Το **Πρόγραμμα 4.11** θα δώσει όλες τις λύσεις εάν καλέσουμε τον στόχο «**?-ektiposeOlous**». Το κατηγόρημα **ektiposeOlous** **οπισθοδρομεί συνεχώς λόγω του fail** και **θα εκτυπώσει όλες τις λύσεις**. Η έξοδος του προγράμματος είναι η εξής:
  - Ο kostas είναι ο πατέρας του/της yannis.
  - Ο kostas είναι ο πατέρας του/της anna.

```
pateras(kostas, yannis).
pateras(kostas, anna).
ektiposeOlous :-
        pateras(X, Y),  write('Ο'),  write(X),
        write('είναι ο πατέρας του/της'), write(Y),
        nl, fail.
```

Πρόγραμμα 4.11: *Έρευνα σε όλο το SLD-δέντρο με χρήση του fail*

4.3. Το κατηγόρημα fail

- ***Παράδειγμα 1***: ***Η δεύτερη χρήση του fail***.
  Χρησιμοποιώντας το **!, fail** μπορούμε να εκφράσουμε σε Prolog ισχυρισμούς της μορφής, ***"στην Μαρία αρέσουν όλα τα ζώα εκτός από τα φίδια"***, **Πρόγραμμα 4.12**.

```
aresei(maria, X) :- zoo(X).
zoo(gata).
zoo(alogo).
zoo(X) :- fidi(X), !, fail.
fidi(ohia).
fidi(voas).
```

Πρόγραμμα 4.12: *Συνδυασμός ! και fail.*

4.4. Άρνηση σε Prolog

- Η εξαγωγή αρνητικών συμπερασμάτων στηρίζεται στην *άρνηση ως αποτυχία* (*negation as failure*) [ClarK 1978], [Apt, Bol 1994]. Ο συμπερασματικός κανόνας της άρνησης ως αποτυχία ορίζεται ως εξής.

- **Ορισμός 4.4**
  - Έστω **Σ** ένας στόχος και **Π** ένα πρόγραμμα. Η άρνηση του **Σ, ¬Σ,** μπορεί να εξαχθεί **εάν** κάθε δυνατή προσπάθεια για απόδειξη του στόχου **Σ** στο πρόγραμμα **Π** αποτυγχάνει πεπερασμένα.

4. Δέντρο Αναζήτησης, Οπισθοδρόμηση και Άρνηση
4.4. Άρνηση σε Prolog

- Θα δούμε πως πρέπει να είναι το *SLD-δένδρο* που δημιουργείται από τον στόχο **Σ** και το πρόγραμμα **Π**.
- Το *SLD-δένδρο* του στόχου **Σ** στο πρόγραμμα **Π** αποτυγχάνει πεπερασμένα εάν
  - **δεν υπάρχει** καμιά ***πεπερασμένη* SLD-εξαγωγή** η οποία να είναι ***SLD-απόρριψη*** και
  - **δεν υπάρχουν *μη πεπερασμένες* SLD-εξαγωγές** στο δένδρο αναζήτησης.
- Δηλαδή, στο *SLD-δένδρο* που προκύπτει από την εκτέλεση του στόχου **Σ**
  - ***δεν υπάρχει επιτυχής κλάδος*** και
  - ***δεν υπάρχει κάποιος μη πεπερασμένος κλάδος***.
- Το **κύριο πλεονέκτημα** αυτού του κανόνα απόδειξης της άρνησης είναι **η αποτελεσματική υλοποίηση του**.

4.4. Άρνηση σε Prolog

- Η αποτυχία της Prolog να αποδείξει ένα στόχο **Σ** σε ένα πρόγραμμα **Π**
  - **δεν σημαίνει ότι πράγματι ο στόχος είναι ψευδής**.
  - **Σημαίνει ότι η μηχανή απόδειξης της Prolog δεν μπορεί να αποδείξει τον στόχο Σ με την συγκεκριμένη βάση γνώσης** που έχει, δηλαδή με τα δεδομένα του προγράμματος **Π**.

4.4. Άρνηση σε Prolog

- Αυτός ο κανόνας απόδειξης αρνητικής πληροφορίας στηρίζεται στην ***υπόθεση του κλειστού κόσμου*** (***closed world assumption***).
- Σύμφωνα με αυτή την υπόθεση
  - κάθε πρόγραμμα εμφανίζεται **να έχει την πλήρη περιγραφή του κόσμου του προβλήματος**.
  - Συνεπώς,
    - **ότι περιγράφεται** σε ένα πρόγραμμα **είναι αληθές** για τον κόσμο του προβλήματος και
    - **ότι δεν περιγράφεται είναι ψευδές** για τον κόσμο του προβλήματος.

4.4. Άρνηση σε Prolog

- Ο στόχος **¬Σ** αποτυγχάνει πεπερασμένα εάν ο στόχος **Σ** επιτυγχάνει. Σε αυτή την περίπτωση υπάρχει κάποια ***πεπερασμένη* SLD-εξαγωγή** η οποία είναι ***SLD-απόρριψη***.
  - Δηλαδή, στο ***SLD-δένδρο*** που προκύπτει από την εκτέλεση του στόχου **Σ** ***υπάρχει επιτυχής κλάδος***.
- Αυτός ο τρόπος ορισμού της άρνησης διαφέρει από τον τρόπο που η άρνηση ορίζεται στη λογική. ***Σύμφωνα με την άρνηση της λογικής θα έπρεπε να περιγράψουμε και την αρνητική πληροφορία***.

4.4. Άρνηση σε Prolog

- Επειδή **η άρνηση της Prolog δεν αντιστοιχεί** στην **άρνηση της λογικής** γι' αυτό χρησιμοποιείται για σύμβολο άρνησης στη **Prolog** το σύμβολο «**\+**» αντί του συμβόλου «¬» ή «**not**» της **λογικής**.
- Με αυτό τον τρόπο δεν δημιουργείται σύγχυση στον τρόπο έκφρασης της άρνησης σε **Prolog** και σε **λογική** αλλά ούτε και στην σημασιολογία τους.

4.4. Άρνηση σε Prolog

- **Παράδειγμα 1 με άρνηση**: Έστω το, Πρόγραμμα 4.13 το οποίο έχει τους προπτυχιακούς φοιτητές του κόσμου του προβλήματός μας.
- Εάν τρέξουμε το στόχο, «**\+ undergrad_student(yannis).**» η Prolog θα απαντήσει **yes**. Αυτός ο στόχος αντιστοιχεί στην ερώτηση: «***Δεν είναι ο Γιάννης προπτυχιακός φοιτητής***;».
- Η Prolog εφαρμόζοντας τον συμπερασματικό κανόνα της «**άρνησης ως αποτυχία**» που στηρίζεται στην ***υπόθεση του κλειστού κόσμου*** αποδεικνύει την αλήθεια της ερώτησης.
-
  - undergrad_student(maria).
  - undergrad_student(nikos).
  - **Πρόγραμμα 4.13: Πρόγραμμα επίδειξης της άρνησης 1.**

4.4. Άρνηση σε Prolog

- **Παράδειγμα 2 με άρνηση**: Έστω το πρόγραμμα, Πρόγραμμα 4.14, το οποίο έχει άρνηση στην πρώτη του πρόταση.
-
  - p(a,Z) :- \+ q(a,Z).
  - q(b,a).
  - q(b,b).

  - **Πρόγραμμα 4.14: Πρόγραμμα επίδειξης της άρνησης 2.**

- Θα τρέξουμε αυτό το πρόγραμμα με τους παρακάτω στόχους.
- | ?- **p(a,b).**
  - yes
- | ?- **p(a,Y).**
  - true ?
  - yes
- | ?- **p(X,Y).**
  - X = a ? ;
  - no
- | ?- **\+q(a,Z).**
  - true ?
  - yes

4.4. Άρνηση σε Prolog

- Ο στόχος "**?- aresei(maria,X).**" δίνει τα ζώα τα οποία αρέσουν στη Μαρία.
- Ο στόχος "**?- \+ aresei(maria,X).**" επιτρέφει **yes** εαν **υπάρχει ζώο το οποίο δεν αρέσει στην Μαρία** διαφορετικά επιστρέφει **no**.
- Για να πάρουμε τα ζώα τα οποία δεν αρέσουν στη Μαρία θα πρέπει να κάνουμε ένα νέο πρόγραμμα επειδή **αρνητικοί στόχοι στην Prolog δεν επιστρέφουν δεσμευμένες τις μεταβλητές τους**.

4. Δέντρο Αναζήτησης, Οπισθοδρόμηση και Άρνηση
4.4. Άρνηση σε Prolog

- Η άρνηση η οποία έχει υλοποιηθεί στην Prolog είναι «**άρνηση με αποτυχία**». Δηλαδή, για να ικανοποιήσει τον στόχο **"?- \+aresei(maria,X)."** η Prolog εκτελεί τον στόχο **"?- aresei(maria,X)."** ο οποίος
  - **εαν αποτύχει θεωρείται ότι η άρνηση του**, δηλαδή ο στόχος **"?- \+ aresei(maria,X).", επιτυγχάνει (είναι αληθής).**
  - **Εαν επιτύχει θεωρείται ότι η άρνηση του**, ο στόχος **"?- \+ aresei(maria,X).", αποτυγχάνει (είναι ψευδής).**
- Η υλοποίηση της άρνησης σε Prolog **δεν επιτρέπει την δέσμευση των μη δεσμευμένων μεταβλητών** που πιθανόν να υπάρχουν στο στόχο.
- **Αρνητικοί στόχοι στην Prolog** πρέπει **να έχουν πλήρως δεσμευμένες όλες τις μεταβλητές τους πριν εκτελεστούν για να είναι σωστοί αρνητικοί στόχοι**.

4.4. Άρνηση σε Prolog

- Το **Πρόγραμμα 4.15** υλοποιεί το κατηγόρημα **den_aresei(maria, X)** το οποίο είναι αληθές εαν το **X** είναι **ένα ζώο που δεν αρέσει στην Μαρία**.

```
den_aresei(maria, X) :- zoo(X).
zoo(X) :- !, fidi(X).
zoo(gata).
zoo(alogo).
fidi(ohia).
fidi(voas).
```

Πρόγραμμα 4.15: *Κατηγόρημα* den_aresei(maria, X).

- ***Παράδειγμα 2: Υλοποίηση Άρνησης***. Το **not X είναι αληθές εάν** το **X δεν είναι αποδεικτέο**. Το **Πρόγραμμα 4.16** υλοποιεί την άρνηση σε Prolog με συνδυασμό ! και fail.

```
:- op(900, fy, not).
not X :- X, !, fail.
not X.
```

Πρόγραμμα 4.16: *Υλοποίηση άρνησης με ! και fail*

- Εάν το ! απομακρυνθεί από αυτό το πρόγραμμα τότε η συμπεριφορά του δεν θα είναι η επιθυμητή.
- Η έννοια του **not** όπως υλοποιήται εδώ αλλά και όπως έχει υλοποιηθεί στην **Prolog** διαφέρει από την αυστηρά λογική έννοια της άρνησης.

4. Δέντρο Αναζήτησης, Οπισθοδρόμηση και Άρνηση
4.4. Άρνηση σε Prolog

- Η υλοποίηση του **not** στη Prolog βασίζεται στην ***άρνηση ως αποτυχία*** και όχι στην αυστηρά λογική έννοια της άρνησης
- Η Prolog εκφράζει αυτή την σημαντική διαφορά, διαθέτοντας το κατηγόρημα «***μη αποδεικτέο***» το οποίον συμβολίζεται με «**\+**».

Τέλος Διάλεξης

Ευχαριστώ!

Ερωτήσεις;