ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών

Τίτλος Μαθήματος

Ενότητα 4: Επεκτάσεις, υλοποίηση, παραλληλία

Παναγιώτης Σταματόπουλος
Σχολή Θετικών Επιστημών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

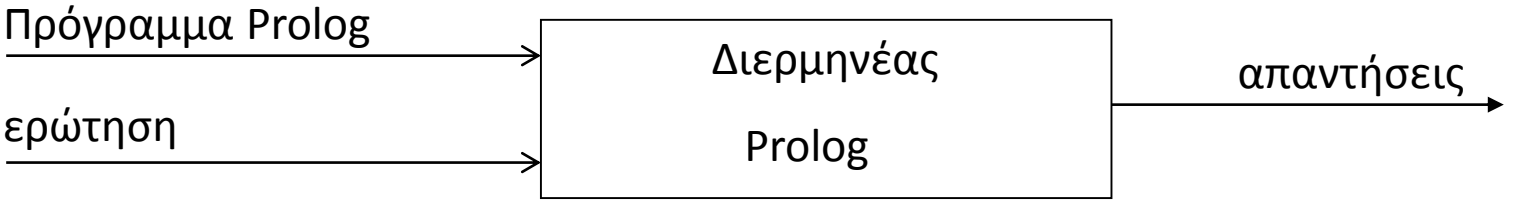
Περιγραφή ενότητας

Αναλυτική παρουσίαση μίας επέκτασης του λογικού προγραμματισμού, αυτής του λογικού προγραμματισμού με περιορισμούς. Θέματα υλοποίησης συστημάτων λογικού προγραμματισμού και παράλληλου λογικού προγραμματισμού.

Επεκτάσεις, υλοποίηση, παραλληλία

Υλοποίηση συστημάτων λογικού προγραμματισμού (1/7)

- Οι προσεγγίσεις για την υλοποίηση ενός συστήματος λογικού προγραμματισμού, ουσιαστικά κάποιας γλώσσας Prolog, είναι οι συνήθεις δύο, είτε μέσω ενός διερμηνέα (interpreter) είτε μέσω ενός μεταγλωττιστή (compiler).
- Ένας διερμηνέας Prolog είναι ένα πρόγραμμα που δέχεται στην είσοδο του ένα πρόγραμμα Prolog και μία ερώτηση που απευθύνεται σ' αυτό και υπολογίζει, εφαρμόζοντας την ανάλυση και την ενοποίηση, τις ζητούμενες απαντήσεις στην ερώτηση. Σχηματικά:



Υλοποίηση συστημάτων λογικού προγραμματισμού (2/7)

- Ο διερμηνέας κωδικοποιεί το πρόγραμμα σαν μία δομή δεδομένων στην περιοχή προγράμματος (code area) και χρησιμοποιεί μία στοίβα (stack) για να διαχειριστεί το δέντρο ανάλυσης που κατασκευάζεται κατά την προσπάθεια απάντησης της ερώτησης. Στη στοίβα φυλάσσονται ό,τι πληροφορίες απαιτούνται για την υποστήριξη, εκτός των άλλων, και της δυνατότητας οπισθοδρόμησης.
- Σε κάθε κόμβο του δέντρου ανάλυσης, που αντιστοιχεί σε μία εγγραφή της στοίβας, κωδικοποιείται η ενοποίηση ενός στόχου με την κεφαλή μίας πρότασης και δημιουργείται ένα περιβάλλον (environment), στο οποίο για κάθε μεταβλητή της επιλεγμένης πρότασης υπάρχει διαθέσιμος χώρος για να καταχωρηθεί η διεύθυνση της τιμής της.

Υλοποίηση συστημάτων λογικού προγραμματισμού (3/7)

- Επειδή μία μεταβλητή κάποιου περιβάλλοντος μπορεί να πάρει τιμή σε μεταγενέστερο κόμβο από αυτόν της δημιουργίας της, πρέπει να υπάρχει η δυνατότητα αναίρεσης της απόδοσης της τιμής αν γίνει οπισθοδρόμηση πριν το σημείο της απόδοσης. Αυτό επιτυγχάνεται μέσω των ιχνών (trails) που είτε καταχωρούνται μέσα σε περιβάλλοντα είτε συνιστούν χωριστή στοίβα.
- Τέλος, όσον αφορά την τιμή μίας μεταβλητής, αυτή μπορεί να παρασταθεί είτε από κάποιο δείκτη σε μία δομή στην περιοχή προγράμματος και ένα περιβάλλον από το οποίο θα βρεθούν οι τιμές των μεταβλητών της δομής είτε από ένα δείκτη σε μία περιοχή που λέγεται σωρός (heap) και στην οποία έχει κτισθεί η τιμή της μεταβλητής. Η πρώτη προσέγγιση χαρακτηρίζεται σαν “structure sharing” και η δεύτερη σαν “structure copying”.

Υλοποίηση συστημάτων λογικού προγραμματισμού (4/7)

- Ποια γλώσσα προγραμματισμού θα επιλέγατε (Prolog ή C) για να υλοποιήσετε ένα διερμηνέα της Prolog; Αν το κάνετε, τι απάντηση θα δώσει στην ερώτηση `?- a(X).` αν το πρόγραμμα που "γνωρίζει" είναι το παρακάτω;

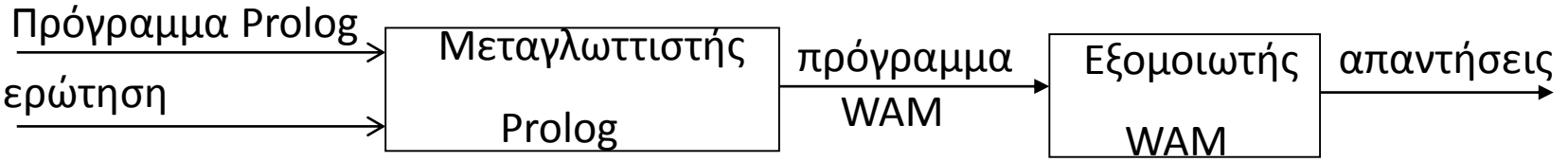
```
a(X) :- b(X), c(X), d(X).
b(X) :- e(X), f(X).
b(X) :- e(X).
c(2).    c(3).    e(1).    e(2).    e(3).
d(1).
d(X) :- g(X).
f(1).    f(3).    g(2).
```

Υλοποίηση συστημάτων λογικού προγραμματισμού (5/7)

- Ένας μεταγλωττιστής Prolog είναι ένα πρόγραμμα που δέχεται στην είσοδο του ένα πρόγραμμα Prolog το οποίο το μεταφράζει σε μία ειδική γλώσσα χαμηλού επιπέδου (τύπου “assembly”), ονόματι WAM, της λεγόμενης αφηρημένης μηχανής του Warren (Warren Abstract Machine).
- Μια ερώτηση που απευθύνεται στο πρόγραμμα Prolog δεν είναι δύσκολο να ενσωματωθεί μέσα σ’ αυτό, με την έννοια ότι μπορεί να θεωρηθεί ότι συνιστάτο σώμα ενός κανόνα που έχει προκαθορισμένη κεφαλή (π.χ. main)και το ζητούμενο είναι να απαντηθεί η ερώτηση που αντιστοιχεί στην κεφαλή (`?- main`).

Υλοποίηση συστημάτων λογικού προγραμματισμού (6/7)

- Το πρόγραμμα WAM που προκύπτει από τη μετάφραση του προγράμματος Prolog και της αρχικής ερώτησης δίνεται στην είσοδο ενός άλλου προγράμματος, που λέγεται εξομοιωτής (emulator) WAM, το οποίο μετά την απαραίτητη “εκτέλεση” υπολογίζει τις ζητούμενες απαντήσεις. Σχηματικά:



Υλοποίηση συστημάτων λογικού προγραμματισμού (7/7)

- Η μηχανή WAM μπορεί να περιγραφεί από την αρχιτεκτονική της και το σύνολο εντολών (instruction set) της. Οι εντολές διαχειρίζονται τη μνήμη της μηχανής και τους καταχωρητές (registers) της. Η μνήμη είναι διαμερισμένη σε χώρους αντίστοιχους με αυτούς που συναντάμε στους διερμηνείς (περιοχή προγράμματος, στοίβα, σωρός, ίχνη), αν και με σημαντικά διαφορετικό τρόπο καταχώρησης των σχετικών πληροφοριών

Αναπαράσταση στο σωρό του όρου

0	STR	1
1	h/2	
2	REF	2
3	REF	3
4	STR	5
5	f/1	
6	REF	3
7	STR	8
8	g/3	
9	REF	2
10	STR	1
11	STR	5

Αναπαράσταση στο σωρό του όρου:

```
g(Z, h(Z, W), f(W))
```

Αναπαράσταση στην περιοχή προγράμματος των προτάσεων (1/2)

```
p(a, X).
p(X, b).
p(X, Y) :- p(X, a), p(b, Y).

p/2: try_me_else L1              %  p
     get_structure a/0, A1       %    (a,
     get_variable X3, A2         %        X)
     proceed                     %
L1 : retry_me_else L2            %  p
     get_variable X3, A1         %    (X,
     get_structure b/0, A2       %        b)
     proceed                     %
```

Αναπαράσταση στην περιοχή προγράμματος των προτάσεων (2/2)

```
L2 : trust_me                    %
     allocate                    % p
     get_variable X3, A1         %   (X,
     get_variable Y1, A2         %       Y) :-
     put_value X3, A1            %             p(X,
     put_structure a/0, A2       %                 a
     call p/2                    %                  ),
     put_structure b/0, A1       %               p(b,
     put_value Y1, A2            %                 Y
     call p/2                    %                  )
     deallocate                  %                   .
```

Παράλληλος λογικός προγραμματισμός (parallel logic programming)

- Ο όρος “παράλληλος λογικός προγραμματισμός” αναφέρεται στην προσπάθεια εκμετάλλευσης των παράλληλων υπολογιστών μέσα από γλώσσες λογικού προγραμματισμού, με σκοπό την αύξηση της απόδοσης των λογικών προγραμμάτων. Έχουν ορισθεί, υλοποιηθεί και χρησιμοποιηθεί πολλές γλώσσες παράλληλου λογικού προγραμματισμού, άλλες λιγότερο άλλες περισσότερο επιτυχημένες, οι πιο πολλές από τις οποίες κατατάσσονται σε μία από τις εξής δύο κατηγορίες:
  - Γλώσσες δεσμευμένης επιλογής (Committed choice languages)
  - Γλώσσες καθαρής παραλληλίας (Pure parallelism languages)

Γλώσσες δεσμευμένης επιλογής (Committed choice languages)

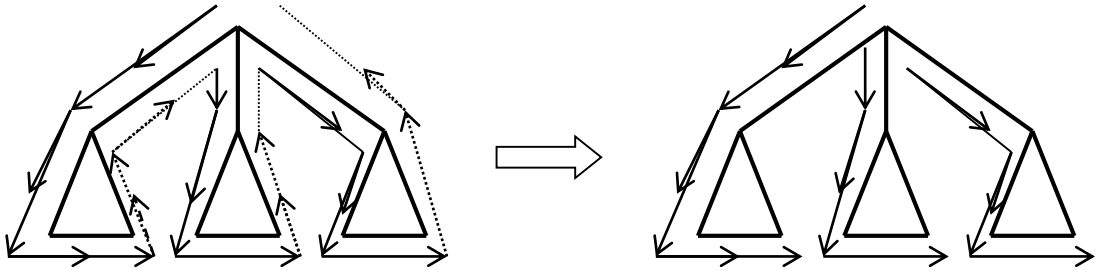
- Οι γλώσσες αυτές είναι τύπου Prolog αλλά σημασιολογικά πολύ διαφορετικές από την Prolog. Δεν υποστηρίζουν μη-ντετερμινισμό, με την έννοια ότι ένας στόχος μπορεί να ικανοποιηθεί το πολύ με έναν τρόπο. Στις γλώσσες αυτές, οι στόχοι του σώματος ενός κανόνα αντιπροσωπεύουν διεργασίες που μπορούν να εκτελούνται παράλληλα και οι οποίες χρησιμοποιούν τις κοινές τους μεταβλητές σαν κανάλια επικοινωνίας. Χαρακτηριστικές γλώσσες στην κατηγορία αυτή είναι η Concurrent Prolog, η Parlog, η Strand κ.λ.π.

Γλώσσες καθαρής παραλληλίας (Pure parallelism languages)

- Οι γλώσσες αυτές αποσκοπούν σε μία πραγματικά καθαρή επέκταση της Prolog προς την κατεύθυνση της εκμετάλλευσης της παραλληλίας που, κατά κάποιο τρόπο, είναι έμφυτη στο λογικό προγραμματισμό.

Ποια είναι η έμφυτη παραλληλία στο λογικό προγραμματισμό; (1/3)

- Ή – παραλληλία (OR-parallelism)
  - Όταν ένας στόχος μπορεί να ικανοποιηθεί από περισσότερες της μίας προτάσεις, επειδή είναι ενοποιήσιμος με τις κεφαλές τους, αντί να δοκιμάζονται αυτές οι προτάσεις η μία μετά την άλλη μέσω οπισθοδρόμησης, θα ήταν δυνατόν να δοκιμασθούν παράλληλα, εφόσον υπάρχουν διαθέσιμοι υπολογιστικοί πόροι (επεξεργαστές).



Ποια είναι η έμφυτη παραλληλία στο λογικό προγραμματισμό; (2/3)

- ΚΑΙ – παραλληλία (AND-parallelism)
  - Στην κλασική Prolog, οι στόχοι του σώματος ενός κανόνα που ενεργοποιήθηκε ικανοποιούνται από αριστερά προς τα δεξιά. Θα ήταν δυνατόν όμως να επιχειρηθεί οι στόχοι αυτοί να ικανοποιηθούν παράλληλα, εφόσον υπάρχουν διαθέσιμοι υπολογιστικοί πόροι, εξασφαλίζοντας όμως ότι οι κοινές τους μεταβλητές θα έχουν τελικά ίδιες τιμές.

Ποια είναι η έμφυτη παραλληλία στο λογικό προγραμματισμό; (3/3)

- Διάφορες γλώσσες λογικού προγραμματισμού υποστηρίζουν μορφές Ή – παραλληλίας ή ΚΑΙ – παραλληλίας ή συνδυασμό των δύο. Η γλώσσα ECLiPSe υποστηρίζει Ή – παραλληλία και μία μορφή ΚΑΙ – παραλληλίας, την ανεξάρτητη ΚΑΙ-παραλληλία (independent AND – parallelism), μέσω της έννοιας των εργατών (workers) που αντιπροσωπεύουν υπολογιστικές μονάδες (όχι, κατ' ανάγκη, επεξεργαστές)

ΚΑΙ – παραλληλία στην ECL^{i}PS^{e} (1/2)

- Η χρήση του τελεστή & μεταξύ δύο στόχων υπονοεί την πρόθεση του προγραμματιστή να ικανοποιηθούν οι στόχοι αυτοί παράλληλα.
- Παράδειγμα:

```
compute(Z) :- (expensive_computation_1(X) &

expensive_computation_2(Y)),
                    combine_results(X, Y, Z).
    expensive_computation_1(X) :- .........
    expensive_computation_2(Y) :- .........
    combine_results(X, Y, Z) :- ..........
```

ΚΑΙ – παραλληλία στην ECLiPSe (2/2)

- Παράδειγμα:

```
q(a).       q(b).       q(c).
r(c).       r(d).       r(e).
p1(X) :- q(X), r(X).
p2(X) :- q(X) & r(X).
```

Σε τι διαφέρουν τα `p1/1` και `p2/1`;

- Παράδειγμα:

Θυμηθείτε την `quicksort/2`. Δεν θα ήταν χρήσιμο οι δύο αναδρομικοί στόχοι στο σώμα της βασική πρότασης να ικανοποιηθούν παράλληλα.

Ή – παραλληλία στην ECLiPSe (1/2)

- Η χρήση της οδηγίας `parallel` για ένα κατηγόρημα υπονοεί την πρόθεση του προγραμματιστή να εξετασθούν οι προτάσεις / ορισμοί για το κατηγόρημα αυτό παράλληλα, αντί μέσω οπισθοδρόμησης, στην περίπτωση απόπειρας ικανοποίησης στόχου με το κατηγόρημα αυτό.
- Παράδειγμα:

```
s(X) :- u(X), p(X), v(X).

u(X) :- .........

v(X) :- .........

:- parallel p/1.

p(X) :- p1(X).p(X) :- p2(X).
```

- Το κατηγόρημα `par member/2` είναι ορισμένο όπως και τογνωστό `member/2`, αλλά είναι δηλωμένο και `parallel`.

Ή – παραλληλία στην ECL^{i}PS^e (2/2)

- Παράδειγμα:

```
process_value(L, Y) :-
    par_member(X, L),
    process(X, Y).
process(X, Y) :- .........
process_all(L1, L2) :-
    findall(Y, process_value(L1, Y), L2).
```

- Το κατηγόρημα `par_indomain/1` έχει την ίδια σημασιολογία με το `indomain/1`, αλλά αντί να δίνει στη μεταβλητή πεδίου όλες τις δυνατές τιμές της μέσω οπισθοδρόμησης, προκαλεί την παράλληλη εξέταση όλων των διαφορετικών κλάδων στο δέντρο ανάλυσης που είναι συνυφασμένοι με τις τιμές αυτές.
- Παράδειγμα: Δοκιμάστε να αντικαταστήσετε στο πρόγραμμα που επιλύει το πρόβλημα των Ν βασιλισσών με περιορισμούς το `indomain/1` σε `par_indomain/1` και ελέγξτε την απόδοση.

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» 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.

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

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.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες