University of Cyprus

Department of Computer Science

EPL434 - Logic Programming and Artificial Intelligence

Recursion *(extra notes)*

18.09.2008

Παράδειγμα χωρίς αναδρομή

Υποθέστε ότι έχουμε την πιο κάτω λογική βάση που αφορά τη σχέση παιδιού:

```
child(kostas,marios).
child(marios,alexantros).
```

Ερμηνεύοντας το κατηγόρημα, ο Μάριος είναι παιδί του Κώστα και ο Αλέξανδρος είναι γιος του Μάριου. Υποθέστε τώρα ότι θέλουμε να ορίσουμε τη σχέση απογόνου, δηλαδή τη σχέση να είσαι παιδί, ή να είσαι παιδί του παιδιού, ή παιδί του παιδιού του παιδιού… Μια πρώτη λύση στο πρόβλημα μας θα ήταν να προσθέσουμε δύο ***μη***-αναδρομικούς κανόνες στη λογική μας βάση:

```
descend(X,Y) :- child(X,Y).
descend(X,Y) :- child(X,Z),
                child(Z,Y).
```

Με αυτούς τους δύο κανόνες το πρόγραμμα μας δουλεύει μέχρι ένα σημείο, όμως είναι εμφανές ότι το πρόγραμμα μας είναι πολύ περιορισμένο. Μας αφήνει δηλαδή να ορίσουμε τη σχέση απογόνου για δύο γενεές ή λιγότερο. Για τα γεγονότα που έχουμε ως στιγμής, αυτοί οι δύο κανόνες ικανοποιούν το πρόβλημα μας. Τι θα γίνει όμως όταν στη λογική μας βάση θέλουμε να προσθέσουμε κι άλλους απογόνους; Για παράδειγμα σε περίπτωση που η λογική μας βάση έχει τα ακόλουθα γεγονότα:

```
child(kostas,marios).
child(marios,alexantros).
child(alexantros,homer).
child(homer,bart).
```

Τώρα οι δύο μας κανόνες είναι ανεπαρκείς. Για παράδειγμα αν κάνουμε τις ακόλουθες δύο ερωτήσεις στην Prolog:

```
?- descend(kostas,homer).
```

ή

```
?- descend(marios,bart).
```

παίρνουμε την απάντηση 'No', πράγμα που *δεν* θέλουμε να συμβεί. Θα μπορούσαμε να 'διορθώσουμε' το πρόγραμμα μας προσθέτοντας τους ακόλουθους δύο κανόνες.

```
descend(X,Y) :- child(X,Z_1),
                child(Z_1,Z_2),
                child(Z_2,Y).
descend(X,Y) :- child(X,Z_1),
                child(Z_1,Z_2),
                child(Z_2,Z_3),
                child(Z_3,Y).
```

Αυτός όμως δεν είναι σωστός τρόπος να λύσεις το πρόβλημα επειδή μας περιορίζει πολύ και ο κώδικας γίνεται δυσκολοδιάβαστος όσο προσθέτουμε κανόνες για να μεγαλώσουμε τον αριθμό γενεών στη σχέση απογόνου:

```
descend(X,Y) :- child(X,Z_1),
                child(Z_1,Z_2),
                child(Z_2,Z_3),
                    .
                    .
                    .
                child(Z_17,Z_18).
                child(Z_18,Z_19).
                child(Z_19,Y).
```

Παράδειγμα με αναδρομή

Αυτό το πρόβλημα μπορεί να λυθεί εύκολα χρησιμοποιώντας αναδρομή (Recursion). Οι πιο κάτω κανόνες λύνουν το πρόβλημα μας:

```
descend(X,Y) :- child(X,Y).   % Βασικός Κανόνας
descend(X,Y) :- child(X,Z),   % Αναδρομικός Κανόνας
                descend(Z,Y).
```

Η σημασιολογία του βασικού κανόνα είναι: αν ο Y είναι παιδί του X, τότε ο Y είναι απόγονος του X. Η σημασιολογία του αναδρομικού κανόνα είναι: αν ο Z είναι παιδί του

Χ και ο Υ είναι απόγονος του Ζ, τότε ο Υ είναι απόγονος του Χ. Ας πάρουμε ένα παράδειγμα για να δούμε τα βήματα που θα εκτελεστούν όταν κάνουμε ερώτηση στο πρόγραμμα μας. Τι θα συμβεί όταν κάνουμε την εξής ερώτηση:

```
descend(kostas,homer).
```

Η Prolog αρχικά θα δοκιμάσει τον πρώτο κανόνα. Η μεταβλητή Χ θα ταυτοποιηθεί με τη σταθερά kostas και η μετβλητή Υ με τη σταθερά homer και μετά θα προσπαθήσει να αποδείξει ότι:

```
child(kostas,homer).
```

Αυτή η προσπάθεια θα αποτύχει αφού η Prolog δεν θα βρει κανένα γεγονός child(kostas,homer), ούτε κανόνα που να συμπεραίνει αυτό. Το επόμενο βήμα της Prolog είναι να κάνει backtracking, για να βρει εναλλακτική στην ερώτηση descend(kostas,homer). Χρησιμοποιώντας τον δεύτερο κανόνα, έχουμε δύο καινούργια δεδομένα:

```
child(kostas,_633),
descend(_633,homer).
```

Η Prolog παίρνει το πρώτο κατηγόρημα και προσπαθεί να το ταυτοποιήσει με ένα γεγονός στη λογική βάση. Η Prolog βρίσκει το γεγονός child(kostas,marios), και η μεταβλητή _633 παίρνει την τιμή marios. Αφού η Prolog απόδειξε το πρώτο κατηγόρημα, θα προσπαθήσει να αποδείξει το δεύτερο:

```
descend(marios,homer)
```

Εδώ είναι που συμβαίνει η αναδρομή. Όπως προηγουμένως, η Prolog θα ελέγξει τον πρώτο βασικό κανόνα του προγράμματος και θα αποτύχει αφού δεν βρίσκει κανένα γεγονός που να ικανοποιεί το:

```
child(marios,homer)
```

Πάλι η Prolog κάνει backtracking και βρίσκει ότι υπάρχει δεύτερος κανόνας (αναδρομικός κανόνας) για να ικανοποιήσει την ερώτηση μας. Χρησιμοποιώντας πάλι τον δεύτερο κανόνα, έχουμε δύο καινούργια δεδομένα:

```
child(marios,_1785),
descend(_1785,homer).
```

Το πρώτο κατηγόρημα ταυτοποιείται με το γεγονός child(marios,alexantros), οπότε η μεταβλητή _1785 παίρνει την τιμή alexantros. Επόμενο βήμα είναι να αποδείξει το:

```
descend(alexantros,homer)
```

Εδώ η Prolog καλεί για δεύτερη φορά αναδρομικά το descend(alexantros,homer). Όπως προηγουμένως, η Prolog προσπαθεί να αποδείξει τον πρώτο κανόνα:

```
child(alexantros,homer)
```

Αυτή τη φορά ο πρώτος κανόνας δεν αποτυγχάνει αφού υπάρχει γεγονός child(alexantros,homer) στη λογική βάση. Η Prolog έχει αποδείξει το descent(alexantros,homer) (δεύτερη αναδρομή). Αυτό όμως επίσης σημαίνει ότι το child(marios, homer) (πρώτη αναδρομή) είναι επίσης αληθές, πράγμα που αποδεικνύει την αρχική μας ερώτηση descend(kostas,homer). Πιο κάτω απεικονίζεται το σχήμα με τα βήματα που ακολούθησε η Prolog για να απαντήσει στην ερώτηση descend(kostas,homer):