Οικονομικό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής
Μάθημα: Τεχνητή Νοημοσύνη, 2024–25
Διδάσκων: Ι. Ανδρουτσόπουλος

Ασκήσεις μελέτης της 12ης διάλεξης

12.1. Δείξτε χρησιμοποιώντας τον αλγόριθμο εξαγωγής συμπερασμάτων προς τα εμπρός *fol-fc-ask* ότι ο τύπος $Path(A, NextOf(NextOf(A)))$ προκύπτει ως συμπέρασμα από την ακόλουθη βάση γνώσεων προτάσεων Horn. Τα x, y, z, u, v, w είναι μεταβλητές, ενώ το A είναι σταθερά.

$$Link(x, NextOf(x))$$
$$(Link(y, z) \Rightarrow Path(y, z))$$
$$\big((Path(u, v) \wedge Link(v, w)) \Rightarrow Path(u, w)\big)$$

Απάντηση:

Υπάρχει στη ΒΓ ο κανόνας $(Link(y, z) \Rightarrow Path(y, z))$, ο οποίος με νέες μεταβλητές γίνεται:

$$(Link(y_1, z_1) \Rightarrow Path(y_1, z_1))$$

Για $\{y_1/x_2, z_1/NextOf(x_2)\}$, η μοναδική συνθήκη του παραπάνω κανόνα ενοποιείται με το ακόλουθο γεγονός της ΒΓ, στο οποίο έχουμε χρησιμοποιήσει νέες μεταβλητές:

$$Link(x_2, NextOf(x_2))$$

και προστίθεται στη ΒΓ το συμπέρασμα:

$$Path(x_2, NextOf(x_2))$$

Υπάρχει επίσης στη ΒΓ ο ακόλουθος κανόνας, στον οποίο έχουμε χρησιμοποιήσει νέες μεταβλητές:

$$\big((Path(u_3, v_3) \wedge Link(v_3, w_3)) \Rightarrow Path(u_3, w_3)\big)$$

Για $\{u_3/x_5, v_3/NextOf(x_5), x_4/v_3, w_3/NextOf(v_3)\}$, οι δύο συνθήκες του παραπάνω κανόνα ενοποιούνται με τα ακόλουθα γεγονότα της ΒΓ, στα οποία έχουμε χρησιμοποιήσει νέες μεταβλητές:

$$Path(x_5, NextOf(x_5))$$
$$Link(x_4, NextOf(x_4))$$

και προστίθεται στο στη ΒΓ το συμπέρασμα:

$$Path(x_5, NextOf(NextOf(x_5)))$$

το οποίο για $\{x_5/A\}$ ενοποιείται με το στόχο $Path(A, NextOf(NextOf(A)))$ και ο αλγόριθμος αποκρίνεται ότι ο στόχος προκύπτει ως συμπέρασμα από την αρχική ΒΓ.

12.2. Γράψτε ένα πρόγραμμα Prolog το οποίο να αντιστρέφει λίστες, όπως στο παρακάτω παράδειγμα (με πλάγια γράμματα φαίνονται οι αποκρίσεις της Prolog, με έντονα ό,τι γράφει ο χρήστης). Μπορείτε να χρησιμοποιήσετε το κατηγόρημα append των διαφανειών (προσοχή: τα πρώτα δύο ορίσματα του append είναι λίστες). Χρησιμοποιήστε π.χ. την SWI-Prolog (βλ.

ιστοσελίδα «σύνδεσμοι» του μαθήματος), για να βεβαιωθείτε ότι το πρόγραμμά σας δουλεύει σωστά.

?- **myReverse([a, b, c, d], R).**
R = [d, c, b, a] **;**
no
?- **myReverse([a, b, c, d], [d, c, b, a]).**
yes

Απάντηση: Το ζητούμενο πρόγραμμα είναι το εξής:

```
myReverse([], []).

myReverse([FirstIn | RestIn], ListOut) :-

  myReverse(RestIn,RestOut), append(RestOut, [FirstIn], ListOut).
```

Ο πρώτος κανόνας δηλώνει ότι το αντίστροφο της κενής λίστας είναι ο εαυτός της. Χρησιμεύει για την περίπτωση που δώσουμε ως είσοδο την κενή λίστα αλλά και ως τερματική συνθήκη για την αναδρομή του δεύτερου κανόνα.

Στον δεύτερο κανόνα, [FirstIn | RestIn] είναι η λίστα προς αντιστροφή. Αν RestOut είναι η αντεστραμμένη RestIn, τότε προσθέτοντας το FirstIn στο τέλος της RestOut, δηλαδή συνενώνοντας τις λίστες RestOut και [FirstIn], προκύπτει η ζητούμενη λίστα ListOut.

12.3. (α) Παραστήστε τις προτάσεις (i), (ii), (iii) και (iv) σε κατάλληλη μορφή πρωτοβάθμιας κατηγορηματικής λογικής, ώστε να είναι δυνατόν να αποδειχθεί με τους αλγορίθμους fol-fc-ask και fol-bc-ask ότι τα (i), (ii) και (iii) συνεπάγονται το (iv).

(i) Η Μαρία συμπαθεί τον Γιάννη.
(ii) Τον Νίκο τον έκοψε ο Γιάννης ή ο Γιώργος.
(iii) Αν κάποιος συμπαθεί τον x, τότε ο x δεν έκοψε κανέναν.
(iv) Τον Νίκο τον έκοψε ο Γιώργος.

Δείξτε σύντομα τα δέντρα απόδειξης (όπως στις διαφάνειες) που θα κατασκεύαζαν οι δύο αλγόριθμοι για να αποδείξουν το (iv). Υπόδειξη: Χρησιμοποιήστε στην παράσταση του (ii) ένα κατηγόρημα της μορφής OrCut(x, y, z) και στην παράσταση του (iii) ένα κατηγόρημα της μορφής NotCut(x,y). Προσθέστε κανόνες που να συνδέουν τα OrCut(x,y,z), Cut(x,y) και NotCut(x,y).

Απάντηση: Παριστάνουμε τις προτάσεις (i), (ii), (iii) και (iv) με τις αντίστοιχες προτάσεις Horn ΠΚΛ παρακάτω. Με τον τύπο (ii) παριστάνουμε ότι τον Νίκο τον έκοψε ο Γιάννης ή ο Γιώργος.

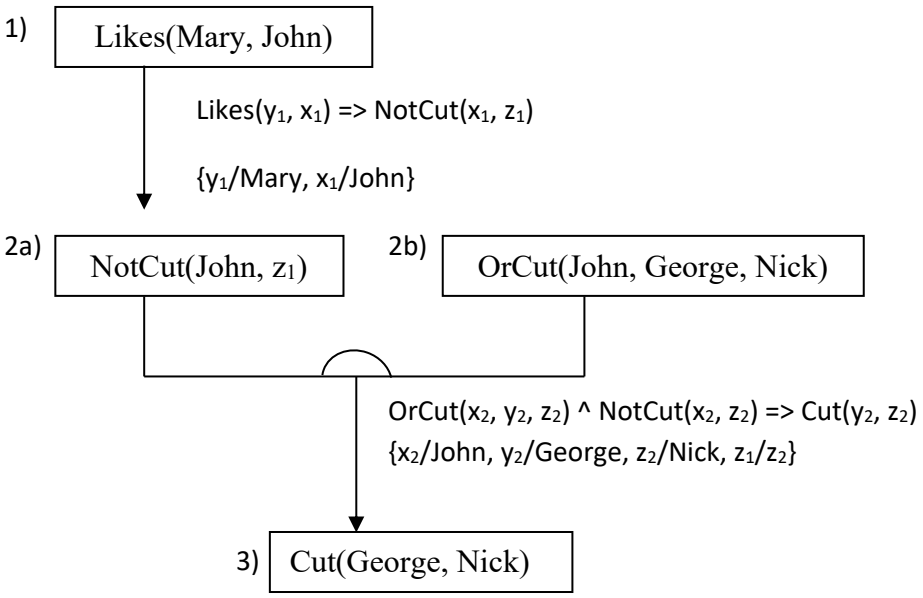
(i) Likes(Mary, John)
(ii) OrCut(John, George, Nick)
(iii) Likes(y, x) => NotCut(x, z)
(iv) Cut(George, Nick)

Προσθέτουμε τους κανόνες:

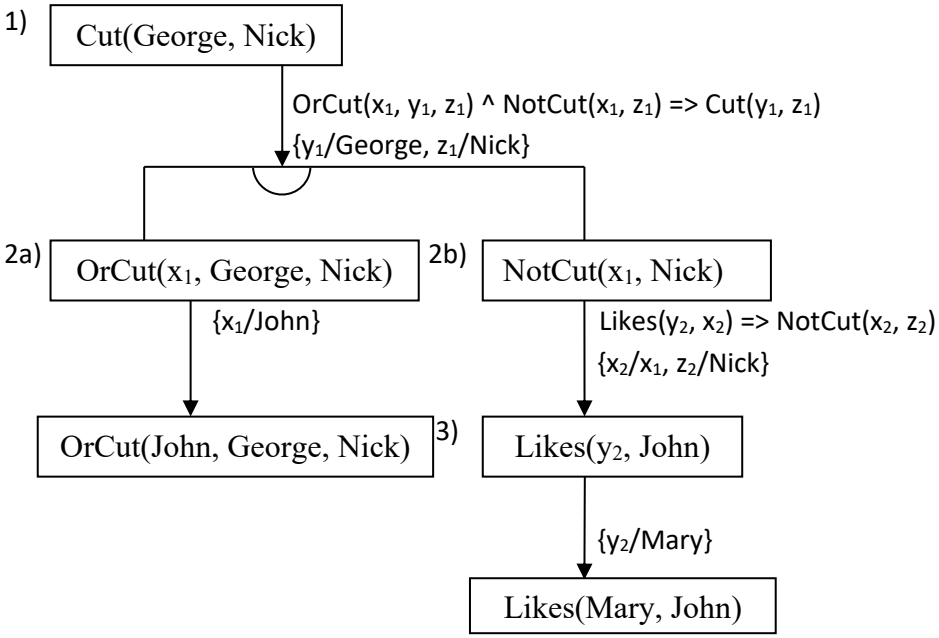
(v) OrCut(x, y, z) ^ NotCut(y, z) => Cut(x, z)
(vi) OrCut(x, y, z) ^ NotCut(x, z) => Cut(y, z)

Ο κανόνας (v) λέει ότι αν αληθεύει πως τον z τον έκοψε ο x ή ο y και αληθεύει επίσης ότι τον z δεν τον έκοψε ο y, τότε αληθεύει ότι τον z τον έκοψε ο x. Αντίστοιχα για τον κανόνα (vi). Τα ζητούμενα δέντρα φαίνονται παρακάτω.

Δέντρο FOL-FC-ASK:



Δέντρο FOL-BC-ASK:



(β) Γράψτε ως πρόγραμμα Prolog τους τύπους του σκέλους (α) για τα (i), (ii), (iii), (iv) και τους επιπλέον κανόνες για τις σχέσεις μεταξύ των OrCut(x,y,z), Cut(x,y) και NotCut(x,y). Χρησιμοποιήστε το πρόγραμμά σας και την SWI-Prolog για να αποδείξετε ότι τον Νίκο τον έκοψε ο Γιώργος, όπως φαίνεται παρακάτω:

?- **cut(george, nick).**
yes
?- **cut(X, nick).**

X = george **;**
no

Απάντηση: Το αντίστοιχο πρόγραμμα σε Prolog είναι το εξής :

```
likes(mary, john).
orCut(john, george, nick).
notCut(X, _) :- likes(_, X).
cut(X, Z) :- orCut(X, Y, Z), notCut(Y, Z).
cut(Y, Z) :- orCut(X, Y, Z), notCut(X, Z).
```

12.4. Συμπληρώστε τα υπογραμμισμένα κενά στο παρακάτω **πρόγραμμα Prolog**, ώστε να **συνδυάζει δύο λίστες ως εξής**:

?- combine([a, b, c], [d, e, f], L).

L = [a, d, b, e, c, f]

?- combine([a, b, c], [d, e], L).

L = [a, d, b, e, c]

?- combine([a, b], [d, e, f, g], L).

L = [a, d, b, e, f, g]

?- combine([a, b], [], L).

L = [a, b]

?- combine([], [d, e, f], L).

L = [d, e, f].

?- combine([], [], []).

Yes

?- combine([a, b, c], [d, e], [a, d, b, e, c]).

Yes

Πρόγραμμα Prolog (συμπληρώστε τα υπογραμμισμένα κενά):

combine(_____, _____, _____).

combine(_____, _____, _____).

combine(_____, _____, _____) :- combine(_____, _____, _____).

Απάντηση:

```
combine([], B, B).
combine(A, [], A).
combine([F1|R1], [F2|R2], [F1, F2|R]):- combine(R1, R2, R).
```